Parag Havaldar and Gérard Medioni

Multimedia Systems

Algorithms, Standards, and Industry Practices

MULTIMEDIA SYSTEMS: ALGORITHMS, STANDARDS, AND INDUSTRY PRACTICES

Parag Havaldar
and Gérard Medioni

COURSE TECHNOLOGY
CENGAGE Learning™

Australia • Brazil • Japan • Korea • Mexico • Singapore • Spain • United Kingdom • United States

COURSE TECHNOLOGY
CENGAGE Learning™

Multimedia Systems: Algorithms, Standards, and Industry Practices
Parag Havaldar and Gérard Medioni

Executive Editor: Marie Lee

Acquisitions Editor: Amy Jollymore

Senior Product Manager: Alyssa Pratt

Editorial Assistant: Zina Kresin

Marketing Manager: Bryant Chrzan

Content Project Manager: Jennifer Feltri

Technical Editing: Green Pen Quality Assurance

Art Director: Faith Brosnan

Compositor: Integra

Cover Designer: Wing-ip Ngan, Ink design, inc.

Cover Image Credit (left): Digital Vision/Getty Images (Royalty Free) Image description (left): Video Motif

Image credit (right): Digital Vision/Getty Images (Royalty Free) Image description (right): Speaker

Image credit: iStockphoto Image descriptions: Urban Teenagers, Wireless

ISBN-13: 978-1-4188-3594-1

ISBN-10: 1-4188-3594-3

Course Technology
20 Channel Center Street
Boston, MA 02210
USA

Printed in Canada
1 2 3 4 5 6 7 13 12 11 10 09

The successful completion of any large project needs devotion, discipline, and sacrifice.

To my parents for their love and the values they instilled in me.

To my family, teachers, friends, and well wishers for their support.

To my students for their feedback and invaluable discussions.

But little did I know whose sacrifice it really was:

To my children Veebha and Shreya—

for the weekends I could not take you hiking; for the afternoons that you waited for me to play; for the evenings I did not take you swimming, bicycling, or skating; and for the nights when I couldn't be beside you when you went to bed.

To my wife Chandrani—

without whose understanding, support, and love, this book was just not possible.

Parag Havaldar

This page intentionally left blank

CONTENTS

CHAPTER 7

CHAPTER 8

CHAPTER 15

Multimedia Databases and Querying 485

CHAPTER 16

Multimedia Frameworks 509

PREFACE

SCOPE AND RATIONAL FOR ANOTHER BOOK IN MULTIMEDIA

Multimedia is now a broad ''umbrella'' that innovatively combines different fields of research and industry to produce practical solutions that are used on a wide scale today. Some of these fields are signal processing, imaging and color science, video and audio analysis, 2D/3D graphics, information theory, compression, networking, databases, watermarking, encryption, mobile terminals, and user interfaces. Research in each field is progressing, and our need to consume digital information has been forever changing. This has resulted in novel multimedia applications and faster dissemination of information that is constantly making our life more convenient when it comes to communication, entertainment, learning, interacting, and so on.

There are many books that address the progress of each of these above-mentioned fields individually. And although there exist books that deal with multimedia systems, most of them have been rather weighted and biased toward explaining only one or a few aspects of multimedia as a whole. For instance, many multimedia books target only the networking and distributing aspects, or only the compression and storage aspects. There is no comprehensive textbook that puts all these concepts coherently together, explaining each area sufficiently enough to understand the problems, solutions, technologies, and standards that can ultimately be used to create broad end-to-end applications in this ever-evolving field.

This book intends to serve that purpose by bringing together the different aspects of a modern multimedia pipeline from content creation, compression, distribution, and consumption on different end terminals. This book is borne out of teachings that the author has been carrying out at the University of Southern California,

feedback from students, and, more important, the author's perspectives gained from working in the industry. We discuss the issues involved in architecting an end-to-end multimedia pipeline and give plenty of examples from the industry, including digital television, IPTV, mobile deployments, Digital Rights Management solutions, digital cinema pipelines, and so on. We also provide lots of practical questions and programming assignments, which are really projects, to augment the student's understanding of the text.

TARGET AUDIENCE AND PREREQUISITES

The content, explanations, and exercises in this book have been designed for senior-level undergraduate students or graduate students in engineering and technical art disciplines.

We do not expect that you have taken courses in all of the engineering fields mentioned and most of the explanations do not assume this. However, it will be helpful to your overall understanding of the multimedia field if you are familiar with one or more of these fields. With regard to exercises, all of them have a rating between 1 and 10, with 1 being very easy, needing only a few moments to answer, whereas a 10 might turn out to be a weekend project. The programming exercises at the end of every chapter should give you hands-on experience with relevant aspects of multimedia. Although we do not assume that you are an expert programmer, you will need to know basic programming or scripting to attempt these. We also provide starter code in C++ and Java, with many sample data sets for the programming exercises. These can be found on the publisher's Web site, *www.cengage.com*, under the Student Downloads section of the book's catalog page.

ORGANIZATION

We start with an introductory chapter that takes the reader from a naïve perspective of multimedia and provides a more concrete definition of multimedia explaining the history, evolution, and current status of what multimedia is today. It also explains the four-part organization of the chapters, each with many visual figures, exercises, programming assignments (programming starter code available at *www.cengage.com*), along with solutions to selected exercises. A complete solution set is available to instructors via the Instructor Downloads section of *www.cengage.com*.

The first part of the book deals with authoring, where we explain relevant issues involved in capturing, representing, and creating content. We start with the digitization process for all media types, explaining the theoretical and practical details, issues in rendering on various display/sound devices, working of cameras, and formats of different media types. Also described are paradigms used by commercial authoring tools in the creation of rich multimedia content for different consumers on high-bandwidth digital networks to low-bandwidth mobile networks. This part also explains each media type (text, images, video, audio, graphics) from its simplistic individual

aspects to more complex content formed by the combinations, such as surround sound, spatial audio, THX, composite, and component video.

The second part is devoted to the data economics of storage and transmission of multimedia content. It first gives an overview of compression, which discusses theoretical and practical limits of information compression as well as explains a taxonomy of algorithms/standards in lossless and lossy coding. The succeeding chapters in this section discuss how these generic compression ideas are purposefully used to minimize perceptual distortion in compressing each media type (images, video, audio, and graphics) with good illustration of examples through every stage of the algorithms. Also discussed are the prominent ISO and ITU compression standards, which include JPEG, JPEG2000, MPEG-1, MPEG-2, H.264, CELP, MP3, MIDI, Topological Surgery used in MPEG-4, X3D, Java3D, and so on. In many cases, we also discuss the bit stream organization and syntax in these relevant standards. We also give examples of user interfaces and parameters exposed by industry-grade compressors for video and graphics.

The third part of the book is devoted to the distribution of compressed content over wired networks and wireless mobile networks. This includes the fundamentals of digital communication using packet-based networks over wired and wireless mediums. In wireless medium access, the main principle behind medium access (FDMA, TDMA, direct sequencing, and CDMA) has been discussed. An important issue for end clients is the steady and synchronized consumption of multimedia information in the presence of varying network throughput, jitter, and errors. We show how such fluid throughput can be achieved using Quality of Service in both wired and wireless cases. Also discussed are the different industry standards from the IP family of protocols to the 1G, 2G, and 3G deployments and 4G developments. One significant chapter in this section is devoted to securing digital content prior to and during distribution, where we discuss how important a role Digital Rights Management plays today. This includes algorithms for encryption and watermarking for all the media types. Also discussed are industry standards and sample deployments of DRM in the industry, which include Apple's iTunes framework, Digital Cinema, DVD encryption standards, MPEG-4's Intellectual Property Management and Protection (IPMP), HDCP, and DTCP.

The last part of the book pays attention to more recent trends and applications in multimedia. Here, we show the paradigm shift in content description/distribution using the MPEG-4 standard compared with the earlier MPEG standards. It shows examples of how MPEG-4 works, where it is currently used, and what might be possible with the standard. One chapter is also devoted to multimedia databases, where we explain the role of semantic queries and the complication involved in formulating and processing semantic queries when compared with queries in standard text databases. We show how solutions in this area have proposed the use of metadata and depict standards that use metadata such as MPEG-7, TV-Anytime, Dublin Core, MXF, DMS-1, and so on. One future requirement of media consumption will be the creation of frameworks that can seamlessly exchange multimedia data with different networks. With many different kinds and brands of commercial networks becoming commonplace today—cell phone networks, Internet, digital cable networks—we discuss the

current progress in MPEG-21 to create such a framework where seamless and commercial exchange can be possible.

Finally, the last chapter concludes with a summary of the content covered in the book as well as content that was left out deliberately or because of the changing progress in the field. We also illustrate the impact that multimedia information consumption and dissemination has had in our industry and society and provide a perspective on where the multimedia industry is likely to move in future.

TEACHING

The book contains more than enough material for a one-semester course on multimedia and can also be used for a two-semester course depending on the depth that an instructor might want to go into each topic. The book has been organized into four parts that progress sequentially. However, depending on the student comfort level and exposure to prerequisites, each part could be taught individually. In the Computer Science Department at the University of Southern California, the authors have been teaching a one-semester course to graduate students majoring in computer science, electrical engineering, industrial engineering, arts, and sciences. The first half of the course has normally covered material in varying detail from the first two parts of the book. The second half of the course has covered selected chapters from the third and fourth parts, depending on overall student interests and the new technologies that had then been in the news.

ACKNOWLEDGEMENTS

A project succeeds because of the contribution of many people. I'd like to thank students of CSCI 576 at the University of Southern California. There are very many who have gone through the course and are now part of the multimedia industry. Every semester has brought forth engaging discussions, novel ideas, and invaluable feedback that have played a pivotal role in the structure of this textbook. I would like to thank the reviewers who helped shape the contents of the manuscript with their numerous suggestions and corrections: Michael Hennessy, University of Oregon; Chung-wei Lee, Auburn University; Roberto Manduchi, University of California, Santa Cruz; Gloria Melara, California State University, Northridge; Refaat Mohamed, Western Kentucky University; Jane Ritter, University of Oregon; and Matthew Turk, University of California, Santa Barbara. Finally, Green Pen Quality Assurance provided technical editing for each chapter. This text will be more useful to students because of their combined efforts.

I would like to thank all the professionals at Cengage Learning, especially Amy Jollymore, Alyssa Pratt, and Jennifer Feltri for their efforts and organization throughout the project. The quality of the textbook would not have been possible without their meticulous oversight and timely management during the production process.

I want to thank friends and family who made it possible for me to undertake and complete this book. In life, the balance of work, family time, parental responsibilities, health, and fun is very critical. This is especially so when you are simultaneously engaged in a large project such as writing a book. My parents instilled values in me that have helped me keep that balance. They have long waited for the completion of the textbook. My daughters Shreya and Veebha have missed me many times when I disappeared in their presence with my laptop and thoughts. And last, but the very most, I could not have done this without the understanding, the support, and the smiles of my wife Chandrani.

We are happy to answer any questions about the book, receive corrections, engage in discussions regarding this evolving field, and provide additional features to help readers have a fruitful experience with the textbook. Please feel free to contact Parag Havaldar directly at havaldar@usc.edu.

This page intentionally left blank

CHAPTER 1

Introduction to Multimedia—Past, Present, and Future

The definition of the word *multimedia* has gone through a large number of evolutionary steps from the time the concept emerged, to what it signifies today, and it will definitely evolve into something new tomorrow. Ask a layman or a computer professional about the definition of multimedia and you get answers such as playing computer games, videoconferencing, listening to MP3s, or watching a movie on the Internet. Most of these multimedia scenarios are tightly bound to the use of a computer. The truth is, most answers are only partially correct but it is not easy to give the term multimedia a concrete and accurate definition. You could naively say that, as the name *multi*media suggests, it consists of all applications that involve a combined use of different kinds of media, such as text, audio, video, graphics, and animation. A presentation that involves all these different media types can be termed a multimedia presentation. Software that involves animations, sound, and text is called multimedia software. Also, any system that incorporates different flavors of media can be termed as a multimedia system. Based on this initial and informal understanding, let us try to categorize the following ten scenarios or examples as being "multimedia" or "not multimedia." Give some thought to each and write down your answers (yes or no). Later in the chapter, we will revisit and analyze these scenarios.

- Watching a Microsoft PowerPoint presentation
- Playing a video game
- Drawing and describing a picture to your friend
- Reading the newspaper in the morning
- Videoconferencing

- Watching television or listening to radio
- Going to a movie theater
- Assembling a car in a garage
- Browsing/searching using the Internet
- Having a telephone conversation

In this introductory chapter, we aim to motivate the reader with what multimedia is and how it has, or currently is, changing the world in terms of the way we communicate, access information, entertain ourselves, and so on. We start by discussing multimedia from a historical perspective in Section 1. Then, in Section 2, we define multimedia information by explaining its inherent properties and its relevant media types. In Section 3, we depict what a multimedia system looks like today and categorize various components that are used. Next, in Section 4 we speak about the technological aspects of multimedia systems, the forces that are feeding its revolution, and we convey the importance of industry-established standards. Section 5 portrays a few thoughts on how multimedia might continue to shape our future. Finally, in Section 6, we set the stage for the book, how it is organized, how it should be read, and how to approach the exercises.

1 MULTIMEDIA: HISTORICAL PERSPECTIVE

The word *multimedia* was coined in the beginning of the 1990s. After the success of the digital audio recording industry, and the distribution of digital audio in the form of compact discs (CDs), the next anticipated step was to create digital content involving images, text, and video along with audio and distribute it in a similar fashion. Outcomes of this were multimedia CD-ROMs, which included informational content as well as games. Examples of these include, Encyclopedia Britannica and interactive CD-ROM games with simple graphics, animations, and audio. These experiences were then only limited to a single person interacting with the content on a PC computer. But this single person-to-PC experience changed dramatically with the advances in digital networks and digital distribution technologies. In fact, the whole multimedia world started to deeply alter our ways of communication with the (1) availability of low-cost capture devices, rendering devices, and smarter software to create content; (2) larger, less expensive storage devices along with research in better compression of media content; and (3) technological advances in digital networks and standardization of distribution protocols.

The preceding three points directly map to three processes that are now inherent to multimedia systems:

- *Multimedia content creation or multimedia authoring*—This process involves digitizing media (audio, images, video) using capture devices and assembling/processing them using smart software and hardware.
- *Storage and compression*—Multimedia content created today has significant memory requirements and has to be engineered so as to minimize necessities for

storage and distribution. The process mostly involves state-of-the-art compression algorithms and standards for audio, video, images, and graphics.

- *Distribution*—Distribution involves how multimedia content is distributed via various media, such as wired cables, optical networks, satellite, wireless networks, or any combination thereof, to specific platforms ranging from television, computers, personal digital assistants (PDAs), and so on.

This threefold view is not new and has been used for information in general—creating or gathering information, storing or recording it, and distributing it to the end user. The table shown in Figure 1-1 gives an evolutionary perspective on the type of information that people have grown accustomed to through the ages, the

Age	Time and era	Type of information	Storage medium	Mode of distribution
Prehistoric	15,000 BC	Sounds to communicate, gestures, painting	Rock surfaces, cave walls	–
Ancient	500 BC	Alphabets, drawing	Invention of paper	People delivering messages, horseback
Middle Ages	400–1000 AD	Letters, writing	Books	Beginning of a postal system
Renaissance	1300–1800 AD	News, paintings, magazine	Books, libraries	Printing press, steam engines, automobiles
Modern world	1900 AD	Morse code, radio, photographs, movies	Film, magnetic tapes, phonograph	Telegram service, wireless radio waves
Electronic	1950–1980	Telephone, television, fax, computers	Electronic memory, cassette tapes, LP records	Radio and TV broadcasting, satellite communication
Digital	1980 to present day	Computers, digital video, surround sound	Hard disks, CD-ROMs, DVDs	Ethernet, wireless networks, optical networks

Figure 1-1 A brief evolution of information

various ways in which information was captured or stored, and the means used to distribute it.

As you go down the table from olden times to recent times, the column showing the type of information suggests that the informational variety that people have grown accustomed to has significantly increased. In the olden days, we had handwritten messages and letters or just word of mouth being propagated, but today people are habituated to information that contains video, audio, images, and text. Simultaneously, the speed at which information has been distributed to the end user has increased. For example, in the event of a war, in olden days it would suffice for a king to know that the battle ended when an emissary reached him carrying a note to that effect, which could often take two or three days. However, in today's digital world, you want to see video/audio/text coverage of a war taking place in foreign areas. People expect to see this coverage in real time on all kinds of devices, including computers, television, PDAs, and so on. From this generic trend involving increasing quantity of information content, its growing medium of storage, and its accelerated distribution, you might wonder what a multimedia system would correspond to at a certain time. For example, if you were to learn about multimedia systems in the 1700s, it would entail learning how the printing press worked, how you would use it to print newspapers, and how efficiently you would distribute that printed information to people. Today, however, if you learn about multimedia systems, it entails dealing with digital data, where all the information is digital and distributed using digital networks to end terminals and rendering devices that are also digital in nature.

2 MULTIMEDIA DATA AND MULTIMEDIA SYSTEMS

Multimedia information can be defined as information that consists of one or more different media types. This definition, however, is a changing one because media types themselves are constantly changing. Today, multimedia information consists of text, audio, video, 2D graphics, and 3D graphics. These are the media types that are used extensively today because of the availability of devices to capture them, as well as capabilities of authoring software applications to combine them to produce a variety of informational and entertaining content. Other more "futuristic" media types are being researched today (and more will be invented tomorrow), but have not yet made it into mainstream multimedia, such as holographs and haptics. Whereas holography deals with the creation of experiences in 3D, haptics deals with providing feedback and interactivity using a sense of touch. Thus, this definition of multimedia information is a changing one.

2.1 Inherent Qualities of Multimedia Data

Before we delve into each media type in detail and the way they can be combined to produce multimedia content, it should be noted that there are certain inherent qualities

generic to all media, which, in turn, define its multimedia nature. These qualities are as follows:

- *Digital*—Multimedia information is always digital.[1] In fact, it is the digital nature of the information that allows it to be combined together (or to keep its own identity) to produce rich content. Whether it is digital images, video, audio, or text, the underlying representation of the information is always bits and bytes.
- *Voluminous*—The size of the data resulting from combining video, audio, and images together is understandably large and voluminous. This causes problems when such high volume data has to be stored, searched, and, worse, when it has to be transmitted over bandwidths, which might be narrow, wide, and even varying. The storage and transmission bandwidth limitations require that the data be compressed.
- *Interactive*—Multimedia content can be interacted with from a high-level application point of view, such as choosing a video to watch or a set of images to browse down to a low level, where you can click on areas of an image causing an action to be taken. For example, on a Web site consisting of hyperlinked text, images, or video, you can read, jump to different Web sites, or browse video in any order you want. Another practical example of interactivity is the navigational capability to jump to chapters as well as browse additional related content in a DVD experience.
- *Real-time and synchronization*—When transmitting content involving different media types, real-time requirements and resulting synchronization issues play a crucial role in the system's architecture and design. Real-time needs imply that there can be only a very small and bounded delay while transmitting information to the end client. Synchronization imposes time-respected rendering of the media, which might be self-contained or interdependent. For instance, video has to play at a certain rate (intramedia) while the accompanying sound must match the video playback rate (intermedia).

Understanding these properties is core to designing good working multimedia applications. For example, suppose you want to capture a live football game and transmit it over the Internet. There are signal-processing issues that stem from capturing (or converting to) digital media, which directly relate to the quality and quantity of data recorded. The quantity of data dictates what type of compression you impose on it to transmit it over a fixed bandwidth network. The real-time transmission requirements need to take into account the available bandwidth, and network traffic. Further more the design and architecture of such a real time, end-to-end system will need buffering, caching, monitoring data throughput, maintaining synchronization and so

1 It is also possible to talk about multimedia information in an analog form. However, we believe that the digital nature of multimedia makes it possible to easily combine the different media types and interact with it to create purposeful content. Hence, for the purpose of this text, we assume multimedia information to be digital.

on at the sender and receiver ends. Moreover, this system architecture should be scalable and able to serve multiple viewers that can connect over varying bandwidths.

2.2 Different Media Types Used Today

As mentioned earlier, the different types of media used to create information is changing. The following sections describing these different types of media are, then, an incomplete taxonomy. These definitions and descriptions for media types are brief and introductory explanations. Detailed explanations of these media types are the subject matter of Chapters 2 and 3.

2.2.1 Text

"This is a line of text to explain that text does convey information!" Text has been commonly used to express information not just today but from the early days. Literature, news, and information being archived today and accessed by browsing on the Internet include a large amount of text. The representation and writing of text information has evolved from simple text to more meaningful and easy-to-read *formatted* text, using a variety of fonts. Today, hypertext is commonly used in digital documents, allowing nonlinear access to information.

One aspect that needs mention is the role text plays in multimedia. It is very natural to downplay the role textual information plays in the context of multimedia, perhaps because of its simplicity, especially when compression, display, and distribution technologies all concentrate on serving the video, audio, and graphical media types. Text has been—and still is—the single most widely used media type to store information, and it has been attributed with aspects of revolutionizing society similar to what multimedia is doing today. You might draw an analogy between the beginning of the digital era, where computer experience was limited to a single person-to-PC situation, and the time when text was contained in handwritten notes and books were kept at specific places, such as libraries and monasteries. The invention of the printing press revolutionized this limited access by making it possible to easily duplicate or print text and send it to various people in different regions, similar to digital duplication and distribution via digital networks today. The printing press opened the way for smaller and more portable texts, lowered the cost of books, and encouraged a great surge in literacy. The resulting ease of access to information globalized Europe in the 1700s, changed people's social and political ways, and ultimately led to the industrial revolution. In addition, the text printing and typography industry invented automated ways to efficiently duplicate and distribute printed information. It was the first automated process that spawned methodical step-by-step processes, which became a blueprint of all automation that followed—from the creation of the assembly line for product manufacturing to the digital world of disseminating information.

2.2.2 Images

Images consist of a set of units called pixels organized in the form of a two-dimensional array. The two dimensions specify the width and height of the images. Each pixel has bit depth, which defines how many bits are used to represent an image.

There are various kinds of images, which can be characterized into groups depending on the following:

- *Bit depth*—Bit depth represents the number of bits assigned to each pixel. Accordingly, images are categorized by the bit depth as binary images where every pixel is represented by one bit *or* gray-level images where every pixel is represented by a number of bits (typically 8) or color images, where each pixel is represented by three color channels.
- *Formats*—Formats are application-specific, for example, faxes are also images that have a format different from digital photographs.
- *Dimensionality*—Images can be enjoyed singularly or combined in a variety of ways. Stereo images are commonly used for depth-perception effects. Images can also be stitched together to form mosaics and panoramas.

2.2.3 Video

Video is represented as a sequence of images. Each image in the sequence typically has the same properties of width, height, and pixel depth. All of these parameters can be termed as spatial parameters. Additionally, there is one more temporal parameter known as frames per second or fps. This parameter describes how fast the images need to be shown per second for the user to perceive continuous motion. Apart from this basic definition, video can be classified depending on the following:

- *Aspect ratio*—A common aspect ratio for video is 4:3, which defines the ratio of the width to height. This has been the adopted standard for the major part of the last century. Today, however, we have a variety of different aspect ratios for high definition, cinemascope, and so on.
- *Scanning format*—Scanning helps convert the frames of video into a one-dimensional signal for broadcast. The interlaced scanning format was invented to make television work in the middle of the last century. Today, in the digital world, display devices can support progressive scanning and provide better quality for visual information.

2.2.4 Audio

Digital audio is characterized by a sampling rate in hertz, which gives the number of samples per second. A sample can be defined as an individual unit of audio information. Each sample also has a size, the sample size, which typically is anywhere from 8-bits to 16-bits depending on the application. Apart from these properties, audio is also described by:

- *Dimensionality*—The dimensions of an audio signal signify the number of channels that are contained in the signal. These may be mono (one channel), stereo (two channels), which is by far the most common. Recent standards also use surround sound which consists of many channels, for example 5.1 surround sound systems have one low frequency speaker and five spatially-located speakers.
- *Frequency Range*—Audio signals are also described by the frequency range or frequency band that they contain. For example, audio voice signals are referred

to as narrow band because they contain lower frequency content. Music is normally referred to as wide band.

2.2.5 2D Graphics

2D graphical elements have become commonplace in multimedia presentations to enhance the message to be conveyed. A 2D graphic element is represented by 2D vector coordinates and normally has properties such as a fill color, boundary thickness, and so on. Additionally, 2D graphical elements are effectively used to create 2D animations to better illustrate information.

2.2.6 3D Graphics

3D graphics are primarily used today for high-end content in movies, computer games, and advertising. Like 2D graphics, 3D graphics largely make use of vector coordinate spaces. 3D graphics concepts and practices have advanced considerably as a science but, until recently, were not a commonplace media type. This is now changing with affordable consumer-level software, scanning devices, and powerful graphics cards now becoming available.

2.3 Classification of Multimedia Systems

A multimedia system, defined end to end, is a system that takes care of all content creation, storage, and distribution issues to various platforms. Depending on the application, multimedia systems can be classified in a variety of ways, such as interaction style, the number of users interacting, when the content is live, and so on. A few common classifications are discussed in the following list:

- *Static versus dynamic*—This differentiation, although rarely used, refers to cases when the multimedia data remains the same within a certain finite time, for example, one slide of a Microsoft PowerPoint presentation or one HTML Web page. Compare this with the dynamic case when the data is changing, for example watching a video.
- *Real-time versus orchestrated*—This is a more common classification. Orchestrated refers to cases when there is no real-time requirement. For example, compressing content on a DVD and distributing it has no real-time requirement. The most important constraint here is the quality of the compressed data. However, showing a game in a live broadcast over the Internet imposes a whole new set of engineering constraints in addition to compression quality, which relate to on-time data delivery and synchronization.
- *Linear versus nonlinear*—Here, the method of interaction with the multimedia data is used to differentiate the system. In a linear system, you would proceed linearly through the information, for example reading an eBook or watching a video. However, if you want to interact with the data in a nonlinear fashion, you would have to make use of links that map one part of the data to another. A well-known example of this is hypertext. You could extend this analogy from text to other media types—images, video, and audio. The term hypermedia generalizes the concept of accessing media nonlinearly.

- *Person-to-machine versus person-to-person*—In this case, the classification is based on whether the end user is interacting with a machine or with another person. For example, playing a CD-ROM game is a simple person-to-machine experience. However, videoconferencing is a person-to-person experience.
- *Single user, peer-to-peer, peer-to-multipeer, and broadcast*—Here, the manner of information distribution is used as a means to classify a multimedia system. You might have a single-user scenario such as browsing the Web, or it could be a peer-to-peer scenario when the information is exchanged from one person/computer to another, for example two friends instant messaging over the Internet. A peer-to-multipeer scenario extends the paradigm to sending messages to a limited set of intended viewers such as in a chat room. Broadcasting is the most general-purpose scenario, where information is sent not to any specific listener(s) but available to all those who want to listen, such as television and radio broadcasts.

3 A MULTIMEDIA SYSTEM TODAY

Multimedia systems can be logically grouped into three parts whose primary functionalities are (1) content production, (2) compression and storage, and (3) distribution to various end users and platforms. The multimedia experience today has transcended a simplistic one person-to-PC scenario to become a very sophisticated one, which involves a distributed and collaborative medium. This has been made possible because of sophisticated, inexpensive devices for capturing and rendering content, as well as smarter software to create content and the availability of increasing digital bandwidth. A typical end-to-end multimedia system today has been graphically depicted in Figure 1-2. It consists of three logical sections, which as explained earlier, correspond to content creation, compression, and distribution.

The content creation section shows a variety of different instruments, which capture different media types in a digital format. These include digital cameras, camcorders or video cameras, sound recording devices, scanners to scan images, and 3D graphical objects. Once the individual media elements are in their digital representations, they may be further combined to create coherent, interactive presentations using software (S/W) applications and hardware (H/W) elements. This content can be stored to disk, or in the case of real-time applications, the content can be sent directly to the end user via digital networks.

The second section deals with the compression of multimedia content. This entails the use of various compression technologies to compress video, audio, graphics, and so on. Shown in the Figure 1-2 are hardware and software elements, such as media encoders and storage devices.

The last section deals with media distribution across a variety of low-bandwidth and high-bandwidth networks. This ranges from cellular, to wireless networks, to cable, to digital subscriber line (DSL), to satellite networks. Distribution normally follows standards protocols, which are responsible for collating and reliably sending

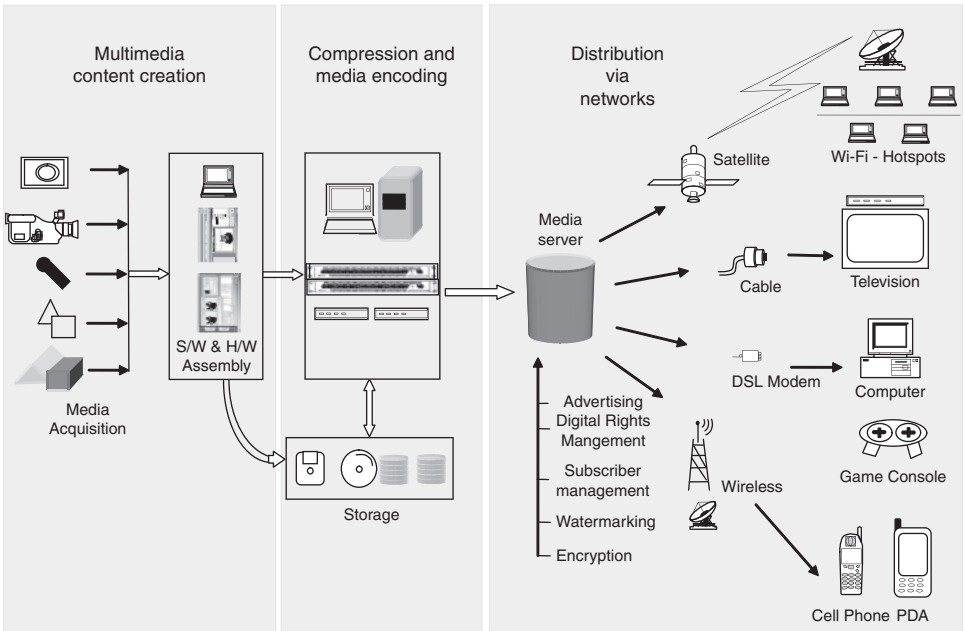


Figure 1-2 Components of a multimedia system today

information to end receivers. The commonly used end receivers are computers, televisions, set-top boxes, cell phones, or even more application- or entertainment-specific items, such as video game consoles.

Now that we know what a multimedia system looks like, and what its general requirements are, let us revisit the ten examples from the beginning of this chapter, and analyze them. In each case, we provide a percentage number that roughly corresponds to the way our students answered on the first day of class. The class included entry level graduate students in computer science and electrical engineering.

1. A PowerPoint presentation. Yes—95%. Of course, PowerPoint presentations involve all types of media and have tools to make it interactive.
2. Playing a video game. Yes—100%. Video games are inherently interactive.
3. Describing a picture to your friend. Yes—10%. What if you are in a chat room where the picture makes up the backdrop and you and your friend talk interactively?
4. Reading the newspaper in the morning. Yes—20%. What if you were reading *www.latimes.com*?
5. Videoconferencing. Yes—100%. Almost all students unanimously agreed that video conferencing is multimedia because video conferencing is considered to be one of first digital multimedia applications.
6. Watching television or listening to radio. Yes—80%. Most said yes because TV comprises audio and video, with channel surfing as interactivity.

However, the multimedia experience becomes clear when you experience digital transmission over cable networks with a DVR (Digital Video Recording; for example, TiVo) that allows you to nonlinearly choose and watch what you want.

7. Going to a movie theater. Yes—90%. Again, most of the students agreed to this being multimedia because movies today entail digital video and audio.
8. Assembling a car in a garage. Yes—0%. Almost all said no. What if the garage is a metaphor for a "3D-room" in an application where designers from different geographic locations get together virtually to assemble 3D car parts?
9. Browsing/searching using the Internet. Yes—100%. Surfing the Internet to read, watch videos, listen to music are applications that involve different media.
10. Having a telephone conversation. Yes—60%. What if you were making use of Voice over IP?

The truth is that all of these scenarios can be defined as multimedia experiences, depending on the devices used, the interactivity involved, and the medium of delivery. One commonality of all these experiences is that they are digital and the end user could always interact with the content or the other end user.

4 THE MULTIMEDIA REVOLUTION

The creation of multimedia information and the ability to exchange it among various devices has clearly created conveniences in our lives. Two decades ago, it was hard to fathom a wireless telephone. Today, whether it is using a cell phone to make a call from any place, or browsing the Internet for information for which you previously had to drive down to a library, or watching live video and audio coverage of an event halfway across the world on your mobile laptop, the rich content of information along with its mobility across various devices has revolutionized our habits of creating, exchanging, browsing, and interacting with information. It is difficult to quantify one definite reason for this revolution to have happened globally but we can definitely attribute a few causes for it to have taken place:

- *Digitization of virtually any and every device*—Today, you have digital cameras, camcorders, sound recorders that make good-quality digital media available for processing and exchange. At the same time, digital displays such as high-performance cathode ray tubes (CRTs), liquid crystal displays, plasma screens, and so on allow us to view information at good resolutions.
- *Digitization of libraries of information*—Virtually all libraries, whether general-purpose or specific, are making their way to be digital.
- *Evolution of communication and data networks*—The research in digital networks and networking protocols have made it possible to exchange huge amounts of data over wired, optical, and wireless mediums. Deployments in this area are making availability of bandwidth on demand.

- *New algorithms for compression*—Because multimedia information is very voluminous, abilities to compress information prior to sending it over networks allow us to engineer applications that perform in real time and with a high fidelity.
- *Better hardware performance*—Microprocessors, along with graphical processing units (GPU) are both getting faster and perform better. Also, large capacity storage devices are becoming common place now, not just with computers but also other hardware devices such as digital cameras, camcorders and so on.
- *Smarter user interface paradigms to view/interact with multimedia information on a variety of terminals*—As personal communication devices get compact and smaller in size, the role of user interfaces becomes important when it is expected for them to have information access capabilities similar to a computers. User interface designs based on touch screens are now playing an increasing role in how we access information on our cell phones, PDAs and so on.

Although this list should not be considered comprehensive, it does show a definite trend, that of increasing our capabilities and capacities to gain ubiquitous access to information. In addition to this, industrial companies along with research/academic communities have formed international bodies to regularly set *standards*. These standards have gone a long way toward allowing us to experience information exchange on a variety of platforms. Standards bodies such as the International Organization for Standardization (ISO) and the International Telecommunication Union (ITU) are primarily responsible for ratifying specifications that the industry uses for interoperability and exchange of information. Within these standards bodies there are groups and committees responsible for each of the different media types. Examples of such committees that have set forth standards for digital media are Joint Pictures Expert Group (JPEG) for images, Motion Pictures Expert Group (MPEG) for video, DVD standards for audio-visual content, Open Systems Interconnection (OSI) for networking, Synchronized Multimedia Integration Language (SMIL), Virtual Reality Modeling Language (VRML) for graphics and so on. There is a reason why you can buy any DVD content created by a movie studio such as Warner Brothers or Universal Pictures; insert it into any DVD player manufactured by Panasonic, SONY, and so on; view the video signal on any HDTV or standard definition TV manufactured by Samsung, RCA, and so on; and enjoy the sound experience on a surround sound system by BOSE or Yamaha. This is possible because everyone adheres to the set standards that allow for interoperability and information exchange.

One aspect previously mentioned that needs more elaboration is the role that user interface paradigms have played in the enabling of our access to and interaction with media information. User interface paradigms have always encouraged users to explore the range of features that a multimedia application offers. The ease and convenience with which any user can access information, manipulate it, and interact with it has always been defined by the interface that allows the user to do so. Graphical user interface paradigms such as buttons, drop-down lists, context-sensitive menus, and other spatial/temporal browsing metaphors on computers have been around for some time now. These metaphors have become more critical today as the devices for access and interaction become more portable, such as cell phones, PDAs, kiosks, and

so on, where the device's capabilities are far less compared with that of a traditional desktop computer. Good examples of current consumer devices today where the role of the user interface has been revisited and rethought are Apple's iPhone and Google's G1 phone. Although both boast of a variety of features, they both have moved toward the use of touch-sensitive screens to have simple but efficient metaphors to browse through and select multimedia information.

5 A POSSIBLE FUTURE

Information has existed in a digital form for more than a decade. The distributed nature of closed and open networks, including the massive Internet, has made this digital information available on a very broad scale. With proper network access protocols, it is now possible to post, access, analyze, search, view, copy, and own digital information from virtually any place without any regard to geopolitical boundaries. Additionally, with the digitization of devices, you do not necessarily need a computer to access information—smaller, smarter devices such as cell phones, PDAs, and so on can do the job just as well. This means that as long as there is network access, virtually anyone—person, group, or organization—located physically anywhere can post information and practically anyone can use it. This digital phase is an ongoing reality in most of the developed nations and will very soon be all over the world.

The digital change has already affected people's everyday life and will do so more effectively in different walks of life. This is seen in the various digital modes in which people primarily communicate, such as cell phones, e-mail, instant messaging, blogs, sharing documents, voice and videoconferencing, chat rooms, social networking sites, and other more interesting avenues yet to come. Aspects of this and improvements there upon will naturally get imbibed into different applications such as distance learning, media entertainment, health care, commerce, defense/military, advertising, and so on. The marriage of Global Positioning Systems (GPS) and communication devices has made it possible to add a new dimension to information analysis. Although the initial uses of GPS were restricted to military applications and commercial aircrafts to improve and automate navigational accuracy, systems are now in place for consumer-level commerce. Novel applications include systems to aid farmers in fertilizer distribution over less-fertile regions; tracking people, vehicles, and cargo; consumer vehicle navigation; and plotting road routes based on traffic, road, closures, and so on.

Although the application areas and communication improvements seem endless, a few common hurdles might need to be solved to make the suggested media and information exchange technologies usable, viable, and commercially practical. The following paragraphs mention a few of these hurdles.

First, there will be much-needed applications that can search quickly through this digital myriad of information. Currently, a number of search engines efficiently search Web pages on the Internet (for example, Google), but all of them are limited to searching text or text-annotated media objects. There will be a need to search, index, and browse through media information such as images, audio, and video. This media information might also be hyperlinked to other media objects, creating hypermedia.

Searching and browsing abilities will not be enough. With the enormous amount of information available for any topic, it is improbable not to be in an "information overload" state. One common example is the amount of e-mail that you need to sift through on a daily basis. Also, more important, when you have a purposeful search about a topic, you get a multitude of information, most of which might not be relevant or even valid, and definitely not easy to efficiently sift through with all the hyperlinks. Current research, which could unfold into a practical future, involves the use of artificial intelligence to create *autonomous agents* and *software robots* whose task is to organize information for the user. The set of tasks or applications an agent can assist with is virtually unlimited: information filtering; information retrieval; mail management; meeting scheduling; selection of books, movies, and music; and so forth.

With the availability of information also comes the need to have specific, limited, and restricted access to it. This is another area, which will need to play an effective role in the future—digital rights management or DRM. DRM refers to protecting the ownership/copyright of digital content by restricting what actions an authorized recipient may take in regard to that content. DRM is a fairly nascent field with implementation limited to specific areas/businesses, most notably with the distribution of movies via DVDs, perhaps because of the large revenue streams and ease of duplication that go with digital movies. But as media and text information become customarily distributed via networking, a variety of businesses in publishing, health, finance, music, and movies will need standard ways to control and authenticate digital information.

6 MAP OF THIS BOOK

This book has been written with one objective in mind: to educate students with the theory and industry practices that are increasingly used to create real applications involving multimedia content. To achieve this goal, we have divided the book into four parts; they are related and should be read sequentially, but depending on the comfort level with each section, you could also read them independently. The chapters in each part provide a description of the fundamental aspects, illustrative examples, and a set of comprehensive exercises, both in theory and programming.

The first part deals with relevant aspects of the *creation* of multimedia content, which includes capturing media into digital form and related signal-processing issues. It discusses representational issues of each media type as well as various formats that are prevalently used to store each media type. It also presents color theory and how it relates to creating display devices from monitors, televisions, and printers. Also, selective techniques and algorithms widely used to combine various media types to create multimedia content are discussed. These include image processing techniques to enhance images, chroma-keying and compositing for video, simple audio filtering, creating graphical animations, and so forth. This section also touches on the importance of user interfaces to interact with multimedia content with a few important user interface paradigms.

The second part of the book analyzes the quantity of multimedia information and discusses issues related to *compression* and *storage*. This section starts with formal analysis of information representation, the theoretical limits of information, leading to

information compression. We give a taxonomy of generic lossless and lossy encoding algorithms. We show how these generic techniques are specifically applied to each media domain—text, images (DCT for JPEG, wavelets for JPEG2000), video (motion compensation for MPEG), audio (MP3, Dolby AC3), and graphics (Topological Surgery). We also discuss a variety of standards established around compression and the different issues related to storage of media.

The third part deals with architectures and protocols used for the *distribution* of multimedia information. It addresses and analyzes distribution-related issues such as medium access protocols, unicast versus multicast, constant bit rate traffic, and media streaming. It also formalizes Quality of Service (QoS) issues for different media and explains how they are controlled using flow control, congestion control, and latency issues. Standards used for non-real-time and real-time media distribution are also discussed—TCP/IP, UDP, HTTP, RTP, RTSP. We also discuss wireless access protocols (WAP) implemented on the Global System for Mobile (GSM) communications as well as the current generation G3 networks. We also discuss issues and solutions that need to be addressed to make the next generation G4 networks a practical reality. This section also addresses the design requirements for end-to-end architectures for commercially deployed applications, such as video on demand, wireless content distribution, GPS with media, and so on. Also explained here are security issues related to distribution, which involves digital watermarking and media encryption.

The last section deals with recent trends in multimedia, including a discussion of real-world *applications* and *standards* for multimedia. Among the topics elucidated here are the latest MPEG-4 standard and multimedia frameworks using the emerging MPEG-21 standard. The section also discusses issues related to multimedia databases and the use of MPEG-7. Finally, this section concludes describing many industry deployments that have been spawned out of this theory and technology. Examples of such deployments include HDTV, DVD, HD-DVD, Blu-ray computer game engines and game content, special effects for movies, Wi-Fi hot spots, and so on.

7 HOW TO APPROACH THE EXERCISES

At the end of each chapter, we provide comprehensive exercises both in theory and programming. Each question is rated by a number from 1 to 10. This number relates to the difficulty level for that question—1 being very easy, requiring only a few moments to solve the question, and 10 being hard enough to involve an entire weekend. Solutions to the exercises are also available in instructional copies. Also, we propose programming exercises, which are really projects, and we do provide a good code base to get started in the form of skeletal frameworks. These are written in C++, and run on both Microsoft Windows and Linux environments. All source code is available under the Student Downloads section of *www.cengage.com*.

This page intentionally left blank

CHAPTER 2

Digital Data Acquisition

Multimedia systems involve three major components: multimedia content creation, compression/storage of multimedia content, and delivery or distribution of multimedia content. Multimedia information is digital, interactive, and voluminous. As depicted in the end-to-end multimedia system diagram Figure 1-1 in Chapter 1, one of the first tasks in creating multimedia content using text, audio, video, and images is to record these individual media types into a digital form, making it is easy to combine and assemble these heterogeneous entities.

This chapter describes the theoretical foundations underpinning the conversion and recording of information into a digital medium. It brings forth issues involved in digitizing one-dimensional (such as audio), two-dimensional (such as images), and three-dimensional (such as video) signals. Section 2 discusses the fundamental digitization process, whereas Sections 4 and 5 present common problems that occur during digitization and solutions to overcome them. Section 3 might seem more involved with the definitions and analysis introduced, but the intuitive understanding of the problems and the solutions should hopefully be clear even without the analysis. This chapter essentially attempts to cover the basics of signal and image processing. However, it is the authors' desire to expose the reader to the deep theory only to the extent necessary from a multimedia point of view, and not follow the rigorous mathematical treatment that generally goes with the subject.

The physical world around us exists in a continuous form. We sense the environment by sensing light, sound energy, pressure, temperature, motion, and so on. All these properties are continuously changing. Recording instruments, such as cameras, camcorders, microphones, gauges, and so forth, attempt to measure information in an electrical and digital form. Let us take the example of a digital camera. In the camera, there could be an image sensor CCD (charge coupled device) array. Each sensor

releases an electric charge that is proportional to the amount of light energy falling on it; the more energy, the higher the charge (within a range). The released charge is then converted into a digital representation in terms of bits, which are ultimately used to display the image information on a rendering device.

It is natural to reason that because multimedia information and systems deal with digital data, we might as well assume that we start with digital data and bypass the understanding of conversions and processes necessary to obtain digital data. However, the conversion process, also known as analog-to-digital conversion, ultimately conditions the quality of digital data, as well as the quantity—both of which are important to the creation and distribution of multimedia. Understanding the conversion process helps in the design of end-to-end systems with the necessary digital data generation for the desired quality, at the same time keeping the generated quantity within the allowed bandwidth.

1 ANALOG AND DIGITAL SIGNALS

Analog signals are captured by a recording device, which attempts to record a physical signal. A signal is analog if it can be represented by a continuous function. For instance, it might encode the changing amplitude with respect to an input dimension(s). Digital signals, on the other hand, are represented by a discrete set of values defined at specific (and most often regular) instances of the input domain, which might be time, space, or both. An example of a one-dimensional digital signal is shown in Figure 2-1, where the analog signal is sensed at regular, fixed time intervals. Although the figure shows an example in one dimension (1D), the theory discussed can easily be extended to multiple dimensions.

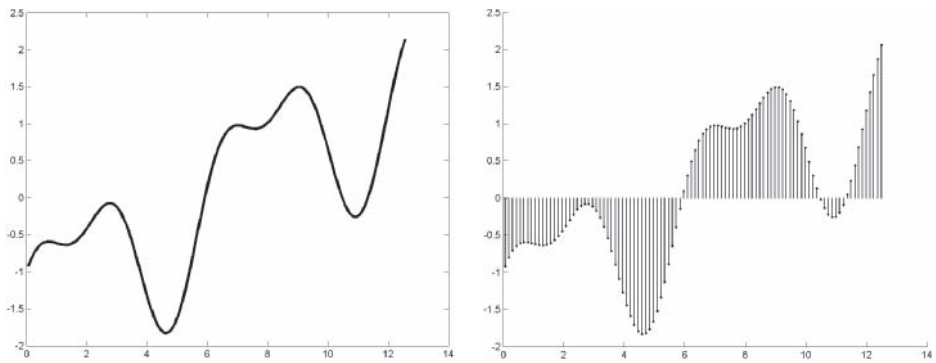


Figure 2-1 Example of an analog signal (left) and a digital signal (right) in one dimension

Before we get into the next section, which addresses the theory of analog-to-digital conversion, it is important to understand the advantages of digital signals over analog ones, some of which are described in the following list:

- When media is represented digitally, it is possible to create complex, interactive content. In the digital medium, we can access each unit of information for a

media type, for example, it is easy to access a pixel in an image, or a group of pixels in a region or even a section of a sound track. Different digital operations can be applied to each, such as to enhance the image quality of a region, or to remove noise in a sound track. Also, different digital media types can be combined or composited to create richer content, which is not easy in the analog medium.

- Stored digital signals do not degrade over time or distance as analog signals do. One of the most common artifacts of broadcast VHS video is ghosting, as stored VHS tapes lose their image quality by repeated usage and degradation of the medium over time. This is not the case with digital broadcasting or digitally stored media types.
- Digital data can be efficiently compressed and transmitted across digital networks. This includes active and live distribution models, such as digital cable, video on demand, and passive distribution schemes, such as video on a DVD.
- It is easy to store digital data on magnetic media such as portable 3.5 inch, hard drives, or solid state memory devices, such as flash drives, memory cards, and so on. This is because the representation of digital data, whether audio, image, or video, is a set of binary values, regardless of data type. As such, digital data from any source can be stored on a common medium. This is to be contrasted with the variety of media for analog signals, which include vinyl records and tapes of various widths.

So, digital data is preferred because it offers better quality and higher fidelity, can be easily used to create compelling content, and can also be compressed, distributed, stored, and retrieved relatively easily.

2 ANALOG-TO-DIGITAL CONVERSION

The conversion of signals from analog to digital occurs via two main processes: *sampling* and *quantization*. The reverse process of converting digital signals to analog is known as *interpolation*. One of the most desirable properties in the analog to digital conversion is to ensure that no artifacts are created in the digital data. That way, when the signal is converted back to the analog domain, it will look the same as the original analog signal. Figure 2-2 illustrates an example of a signal converted from the analog domain to digital domain and back to the analog domain. Multimedia content is digital and distributed in a digital format. The end device onto which the content is rendered might not necessarily be digital, for instance a CRT monitor. It is essential to ensure that the rendered analog signal is very similar to the initial analog signal.

2.1 Sampling

Assume that we start with a one-dimensional analog signal in the time t domain, with an amplitude given by $x(t)$. The sampled signal is given by

$$x_s(n) = x(nT), \text{ where } T \text{ is the sampling period and}$$
$$f = 1/T \text{ is the sampling frequency.}$$

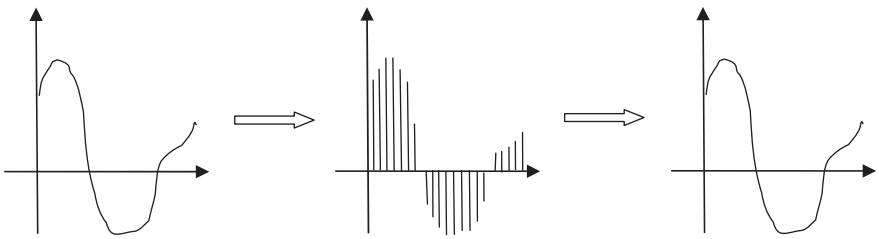


Figure 2-2 Analog-to-digital conversion and the corresponding interpolation from the digital-to-analog domain

Hence, $x_s(1) = x(T)$; $x_s(2) = x(2T)$; $x_s(3) = x(3T)$; and so on. If you reduce T (increase f), the number of samples increases; and correspondingly, so does the storage requirement. Vice versa, if T increases (f decreases), the number of samples collected for the signal decrease and so does the storage requirement. T is clearly a critical parameter. Should it be the same for every signal? If T is too large, the signal might be under sampled, leading to artifacts, and if T is too small, the signal requires large amounts of storage, which might be redundant. This issue is addressed in Section 4. For commonly used signals, sampling is done across one dimension (time, for sound signals), two dimensions (spatial x and y, for images), or three dimensions (x, y, time for video, or x, y, z for sampling three-dimensional ranges). It is important to note that the sampling scheme described here is theoretical. Practical sampling involves averaging, either in time or space. Therefore, sampling is always associated with filtering, and both effects need to be taken into account. Filtering is explained in Section 5.

2.2 Quantization

Quantization deals with encoding the signal value at every sampled location with a predefined precision, defined by a number of levels. In other words, now that you have sampled a continuous signal at specific regular time instances, how many bits do you use to represent the value of signal at each instance? The entire range R of the signal is represented by a finite number of bits b. Formally,

$$x_q(n) = Q[x_s(n)], \text{ where Q is the rounding function.}$$

Q represents a rounding function that maps the continuous value $x_s(n)$ to the nearest digital value using b bits. Utilizing b bits corresponds to $N = 2^b$ levels, thus having a quantization step delta $= R/2^b$. Figure 2-3 shows an analog signal, which is sampled at a common frequency, but quantized using different number of bits, 4 or 2.

Because each sample is represented by a finite number of bits, the quantized value will differ from the actual signal value, thus always introducing an error. The maximum error is limited to half the quantization step. The error decreases as the number of bits used to represent the sample increases. This is an unavoidable and irreversible loss, as the sample would otherwise need to be represented with infinite precision, which requires an infinite number of bits. The question, then, is how many bits should be used to represent each sample? Is this number the same for all signals?

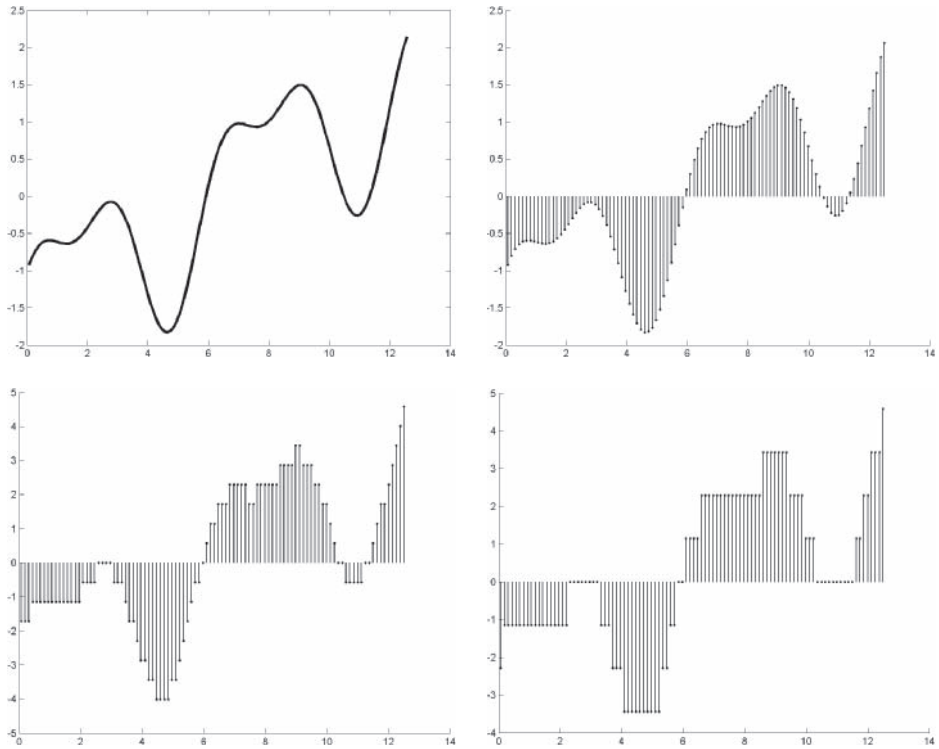


Figure 2-3 Original analog signal (upper left) is shown sampled and quantized at different quantization levels. For quantization, 8 bits (256 levels), 4 bits (16 levels), and 3 bits (8 levels) were used to produce the digital signals on the top right, bottom left, and bottom right, respectively.

This actually depends on the type of signal and what its intended use is. Audio signals, which represent music, must be quantized on 16 bits, whereas speech only requires 8 bits. Figure 2-4 illustrates quantization effects in two dimensions for images. The results show that the error increases as the number of quantization bits used to represent the pixel samples decreases.

Before we conclude this section on quantization, it is worthwhile to explain the different types of quantization schemes used. The discussion so far depicts uniform quantization intervals in which the output range of the signal is divided into fixed and uniformly separated intervals depending on the number of bits used. This works well when all the values in the range of the signal are equally likely and, thus, the quantization error is equally distributed. However, for some signals where the distribution of all output values is nonuniform, it is more correct to distribute the quantization intervals nonuniformly. For instance, the output intensity values of many audio signals such as human speech are more likely to be concentrated at lower intensity levels, rather than at higher intensity levels in the dynamic audio range. Because the distribution of output values in such signals is not uniform over the entire dynamic range, quantization errors should also be distributed nonuniformly.

Figure 2-4 Examples of quantization; initial image had 8 bits per pixel, which is shown quantized from 6 bits down to 1 bit per pixel

An illustration of this is shown in Figure 2-5, where the original signal on the left is shown digitized using eight uniform quantization intervals (center) and eight logarithmic quantization intervals (right). The digitized signal to the right preserves the original signal characteristics better than the digitized signal in the center. We will revisit such nonuniform quantization schemes in the context signal compression described in the compression related chapters 6 through 10 in the book.

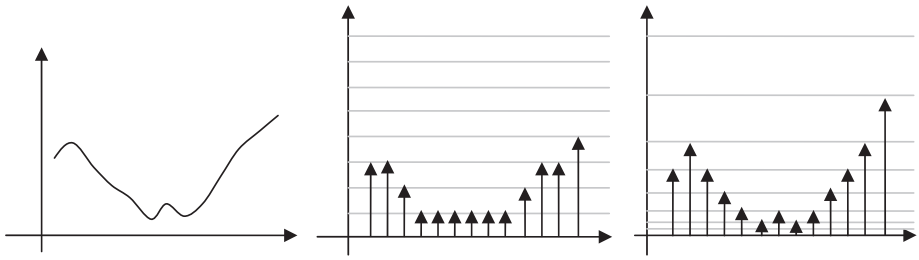


Figure 2-5 Nonlinear quantization scales. The left signal shows the original analog signal. The corresponding digitized signal using linear quantization is shown in the center. The right signal is obtained by a logarithmically quantized interval scale.

2.3 Bit Rate

Understanding the digitization process from the previous two subsections brings us to an important multimedia concept known as the *bit rate*, which describes the number of bits being produced per second. Bit rate is of critical importance when it comes to storing a digital signal, or transmitting it across networks, which might have high, low, or even varying bandwidths. Bit rate, which is measured in terms of bits per second, consists of the following:

$$\begin{aligned} \textit{Bit rate} &= \frac{\textit{Bits}}{\textit{Second}} = \left(\frac{\textit{Samples produced}}{\textit{Second}}\right) \times \left(\frac{\textit{Bits}}{\textit{Sample}}\right) \\ &= \textit{Sampling rate x Quantization bits per sample} \end{aligned}$$

Ideally, the bit rate should be just right to capture or convey the necessary information with minimal perceptual distortion, while also minimizing storage requirements. Typical bit rates produced for a few widely used signals are shown in Figure 2-6.

Signal	Sampling rate	Quantization	Bit rate
Speech	8 KHz	8 bits per sample	64 Kbps
Audio CD	44.1 KHz	16 bits per sample	706 Kbps (mono) 1.4 Mbps (stereo)
Teleconferencing	16 KHz	16 bits per sample	256 Kbps
AM Radio	11 KHz	8 bits per sample	88 Kbps
FM Radio	22 KHz	16 bits per sample	352 Kbps (mono) 704 Kbps (stereo)
NTSC TV image frame	Width – 486 Height – 720	16 bits per sample	5.6 Mbits per frame
HDTV (1080i)	Width – 1920 Height – 1080	12 bits per pixel on average	24.88 Mbits per frame

Figure 2-6 Table giving the sampling rate, quantization factor, and bit rates produced for typical signals

3 SIGNALS AND SYSTEMS

We now present some fundamental elements in the field of digital signal processing to better understand the process of converting analog signals to the digital domain. The goal of the next few sections is to understand signals, how they can be sampled, and the limitations that signals impose on the sampling of signals. A first distinction needs to be made regarding the type of signal under consideration. A practical categorization as described in Figure 2-7 can be viewed as follows:

- Continuous and smooth—Such as a sinusoid.
- Continuous but not smooth—Such as a saw tooth.
- Neither smooth nor continuous—For example, a step edge.
- Symmetric—Which can be further described either as odd ($y = sin(x)$) or even ($y = \cos(x)$). Note that any signal can be decomposed into the sum of an odd and even part.
- Finite support signals—Signals that are defined for a finite interval and zero outside of that interval.
- Periodic signal—A signal that repeats itself over a time period. For a periodic signal f(x), the period is defined to be T if $f(x + T) = f(x)$. For any function $f(t)$, we can define a periodic version of $f(t)$, $g(t) = \Sigma_k f(t - kT)$.

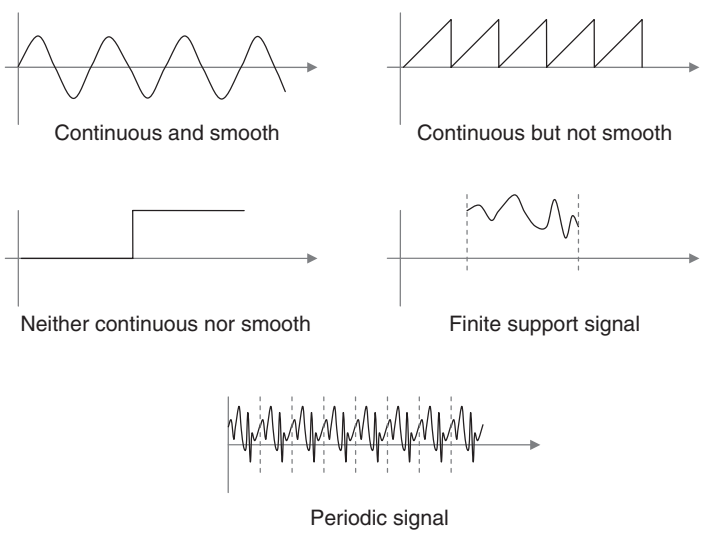


Figure 2-7 Sample signals with different kinds of properties—smooth, unsmooth, continuous, discontinuous, finite support, and periodic. Normally, signals are composed of a combination of one or more of these properties.

3.1 Linear Time Invariant Systems

Any operation that transforms a signal is called a *system*. Understanding linear time invariant systems is necessary to gain insight into the fundamental results that now characterize the process by which any practical system performs sampling and digitization.

Let a system transform an input signal $x(t)$ into an output $y(t)$. We call this system a linear if the output and input obey the following:

$$\begin{aligned} &\textit{If} && x(t) = c_1 x_1(t) + c_2 x_2(t) \\ &\textit{then} && y(t) = c_1 y_1(t) + c_2 y_2(t) \\ &\textit{where } y_k(t) \textit{ is the sole output resulting from } x_k(t) \end{aligned}$$

Time invariance of a system can be defined by the property that the output signal of a system at a given instant in time, depends only on the input signal at that instant in time. Or more formally, if the output of the system at t is $y(t)$, produced by input $x(t)$, then the output of the system at $y(t - T)$ is due to the input $x(t - T)$. Thus, the term time invariance captures the essence of delay. If an input is affected by a time delay, it should produce a corresponding time delay in the output. Both these terms together define a linear time invariant system (LTI system). Their properties are well understood and commonly used in digital system design.

Another important operation that is used to process signals in an LTI system is *convolution*. The convolution of two signals f and g is mathematically represented by f^*g. It is the result of taking the integral of the first signal multiplied with the other signal reversed and shifted.

$$\begin{aligned} (f * g) &= \int_{-\infty}^{\infty} f(\tau) \cdot g(t - \tau) d\tau \\ &= \int_{-\infty}^{\infty} f(t - \tau) \cdot g(\tau) d\tau \end{aligned}$$

3.2 Fundamental Results in Linear Time Invariant Systems

Any LTI system is fully characterized by a specific function, which is called the *impulse response* of the system. The output of the system is the *convolution* of the input with the system's impulse response. This analysis is termed as the *time domain* point of view of the system. Alternatively, we can also express this result in the *frequency domain* by defining the system's *transfer function*. The *transfer function* is the *Fourier transform* of the system's impulse response. This transfer function works in the frequency domain, and expresses the systems operation on the input signal in terms of its frequency representation to produce an output signal in the frequency domain is then the product of the transfer function and the Fourier transform of the input. Thus, as illustrated in Figure 2-8, performing a convolution in the time domain is equivalent to performing multiplication in the frequency domain.

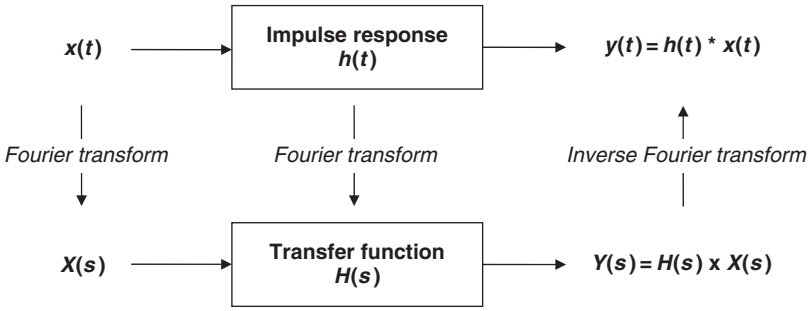


Figure 2-8 Relationship between the impulse response function in the time domain and the transfer function in the frequency domain

3.3 Useful Signals

Specific signals—for example, the Delta function, box function, step function—play an important role in many signaling-processing operations, such as sampling, filtering, and convolution. In this section, we describe some of these functions before delving into how they are used.

The *Delta function* is a mathematical construct introduced by the theoretical physicist Paul Dirac and, hence, is also known as the Dirac Delta function. Informally, it is a sharp peak bounding a unit area: $\delta(x)$ is zero everywhere except at $x = 0$ where it becomes infinite, and its integral is 1. A useful property of the Delta function is *sifting*. The sifting property is used in sampling and can be shown in the following equation:

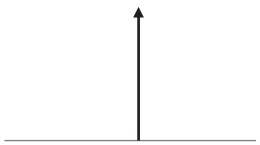
$$\begin{aligned} f(t) * \delta(t - T) &= \int_{-\infty}^{\infty} f(\tau) \cdot \delta(t - T - \tau) d\tau \\ &= \int_{-\infty}^{\infty} f(\tau) \cdot \delta(\tau - (t - T)) d\tau \\ &= f(t - T) \end{aligned}$$

The preceding derivation shows that the signal $f(t)$ convolved with the Delta function at $(t - T)$ yields the same value of the function at $(t - T)$. This is very useful in developing the idea of convolution and sampling. By convolving the input signal with the Delta function and using its sifting property, we can represent an approximation of any system's output.

The comb is another useful function, which is an infinite series of *Dirac Delta* functions with a regular spacing T. During sampling, you could use the comb and convolve it with the input analog function to get sampled digital values. Other useful functions, such as the step function, the box function, and the sinc function, are shown in Figure 2-9.

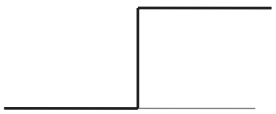
3.4 The Fourier Transform

The Fourier transform is due to Joseph Fourier (1768–1830), who published his initial results in 1807. Fourier proposed to represent any periodic, continuous signal as a sum

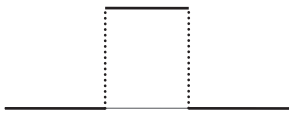


Delta function

Comb function $\Delta_T(t) \stackrel{\text{def}}{=} \sum_{k=-\infty}^{\infty} \delta(t - kT)$

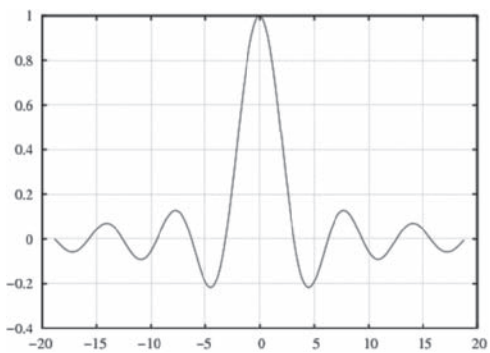


Step function $H(x) = \int_{-\infty}^{x} \delta(t)\mathrm{d}t$



Box function

$$\mathrm{rect}(t) = \sqcap(t) = \begin{cases} 0 & \text{if } |t| > \frac{1}{2} \\ \frac{1}{2} & \text{if } |t| = \frac{1}{2} \\ 1 & \text{if } |t| < \frac{1}{2} \end{cases}$$



Sinc function $\mathrm{Sa}(x) = \mathrm{sinc}(x) = \dfrac{\sin(\pi x)}{\pi x}$

Figure 2-9 Useful functions in signal-processing theory: Delta function (top left), comb function (top right), step function (middle left), box function (middle right), and sinc function (bottom)

of individual complex sinusoids (a Fourier series expansion). In other words every periodic, continuous signal can be expressed as a weighted combination of sinusoid (sine and cosine) waves. The weights that are used to combine the sinusoids are called the Fourier series coefficients, or spectral components. More details are presented in Section 6 at the end of this chapter, and we refer the reader to this section for a more intuitive as well as an involved analysis. The Fourier formulation represents a transformation of the signal into frequency space. The spectral components of a signal define

a discrete set of frequencies that are combined to form the signal. This is also called a harmonic decomposition. Fourier was originally criticized for not proving the existence or the convergence of his series. The formal proof and criteria was proposed by Dirichlet, but this is beyond the scope of our coverage. What is important to know is that most signals fit these criteria. An example of two signals and their Fourier transforms is shown in Figure 2-11.

A remarkable property of the Fourier transform is *duality*. Informally, it means that given a function and its Fourier transform, you can interchange the labels signal and spectrum for them. For instance, the Fourier transform of a *Delta* function is a constant, and the Fourier transform of a constant is a Delta. Also, the Fourier transform of a *box* is a *sinc*, and the Fourier transform of a *sinc* is a *box*. More generally, if a signal $f(t)$ has a Fourier transform $g(\omega) = F\{f(t)\}$, then $h(\omega) = F\{g(t)\} = f(-\omega)$. Therefore, for symmetric signals, $h(\omega) = f(\omega)$.

4 SAMPLING THEOREM AND ALIASING

Here we answer the questions put forth in Section 2.1, which deal with the rate at which sampling should occur. The value of a nonstatic signal keeps changing depending on its frequency content. Some sample signals are shown in Figure 2-10.

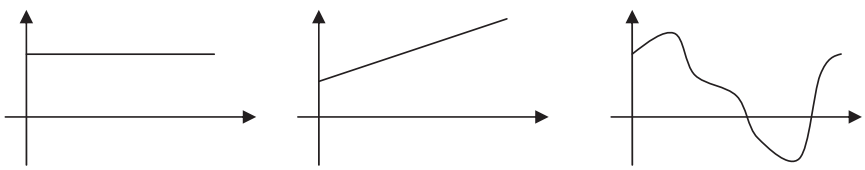


Figure 2-10 Examples of simple one-dimensional functions. Different numbers of samples are required to digitize them

If these signals are to be digitized and reproduced back in the analog domain, the number of samples required to ensure that both starting and ending analog signals are the same is clearly different. For instance, in the first case where the signal is not changing (and, hence, has zero frequency), one single sample will suffice for this purpose. Two samples are required for the second signal and many more for the third one. As we go from left to right, the frequency content in the signals increases, and, therefore, the number of samples needed during the digitization process also goes up.

The relationship between signals and sampling rate was established during the late 1920s by Henry Nyquist and later formalized by Claude Shannon in 1950. This relationship states that the signal has to be sampled using a sampling frequency that is greater than twice the maximal frequency occurring in the signal. Or, it can more formally be stated as follows:

1. A bandlimited signal $f(t)$ with max frequency ωF is fully determined from its samples $f(nT)$ if $2\pi/T > 2\omega F$
2. The continuous signal can then be reconstructed from its samples $f(nT)$ by convolution with the filter

$$r(t) = \text{sinc}(\omega F(t - nT)/2\pi)$$

For example, if the signal has a maximal frequency of 10 kHz, it should be sampled at a frequency greater than 20 kHz. Here, 20 kHz is known as the *Nyquist* sampling frequency for that signal.

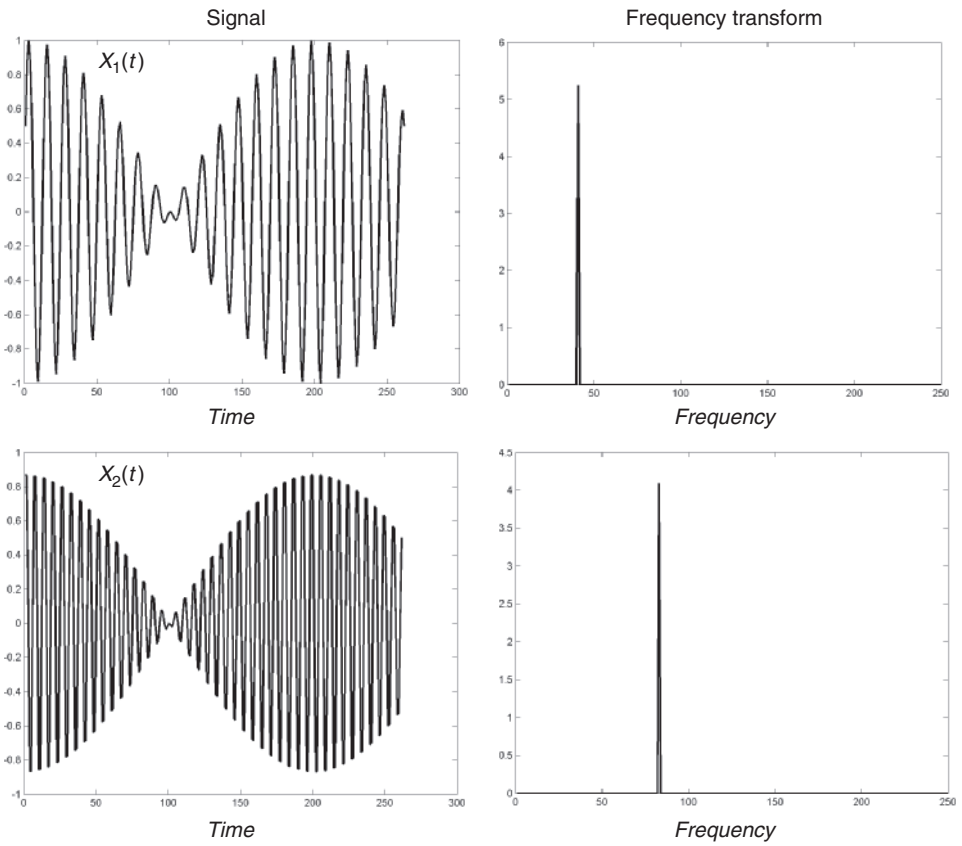


Figure 2-11 Two signals $x_1(t)$ and $x_2(t)$ having different frequencies are shown on the left and their Fourier transforms on the right. The transforms show the amount of energy at all frequencies. Both input signals have just one frequency as their Fourier transform shows, but the frequency in $x_1(t)$ is lower than that in $x_2(t)$.

What happens if your sampling frequency is higher than your Nyquist frequency? The answer is: nothing special. When it comes to reproducing your analog signal, it is guaranteed to have all the necessary frequencies and, hence, the same signal is reproduced. However, by performing such over sampling, you have generated more digital data than necessary, which increases your storage and transmission overhead.

What happens if your sampling frequency is lower than your Nyquist frequency? In this case, you have a problem because all the frequency content is not well captured during the digitization process. As a result, when the digital signal is heard/viewed or

converted back into analog form, it does not correspond to the initial starting analog signal. This results in artifacts, which is termed as *aliasing*. Aliasing is the term used to describe loss of information during digitization. Such undesirable effects are experienced for 1D signals such as sound, 2D signals such as images and graphics, and even in 3D signals such as 3D graphics. Next, we discuss the aliasing effects commonly observed in the spatial or temporal domains.

4.1 Aliasing in Spatial Domains

Aliasing effects in the spatial domain are seen in all dimensions. Figure 2-12 illustrates examples in one and two dimensions. In the one-dimensional case, the original sinusoid was sampled (shown in gray) at a lesser frequency than the Nyquist rate. When the samples are interpolated to reproduce the original, you can see that the two signals do not match, except at the exact sample points.

In the two-dimensional example, the first image shows the original image consists of 750 samples in the x and 620 samples in the y direction. The succeeding figures show what the reconstructed signal looks like by reducing the sampling resolution in both directions. You can see that, as the number of spatial samples decreases, not all of the original frequencies are properly captured. This is especially observable in the high-frequency areas of the image, such as the cloth and hair areas where the pixels are changing more rapidly. Also, the versions with fewer samples display increased effects of blur. This is the result of how the camera image plane's (charge coupled device (CCD)) array works. The pixel intensity value is a representation of the average light energy falling on the CCD element.

4.2 Aliasing in the Temporal Domain

Examples of temporal aliasing can be seen in western movies, by observing the motion of stagecoach wheels. As the stagecoach starts to move, we observe its wheels rotating in the expected forward direction. As the stagecoach speeds up, we see that the wheels appear to rotate in the opposite direction to the initial one. We realize that this phenomenon happens because the movie consists of a number of frames, or *digital samples*. With the wheels' acceleration, the digitized succession of frames will show the spokes appearing to move in the opposite direction to the actual direction of the wheels' rotation. This occurs because the movie camera has under sampled the wheels' motion. Similar effects in real life occur when you watch a helicopter blade as it speeds up. As the blades start moving, our eyes can see each blade but as the blades rotational velocity increases, the blades' apparent motion is sensed or sampled too slowly by our vision and they appear to have a slower rate of rotation than the true rotational speed.

4.3 Moiré Patterns and Aliasing

Another interesting example of aliasing, called the *moiré effect*, can occur when there is a pattern in the image being photographed, and the sampling rate for the digital image is not high enough to capture the frequency of the pattern. The term originates from *moire* (or *moiré* in its French form), a type of textile, traditionally of silk but now also of cotton or synthetic fiber, with a rippled or ''watered'' appearance.

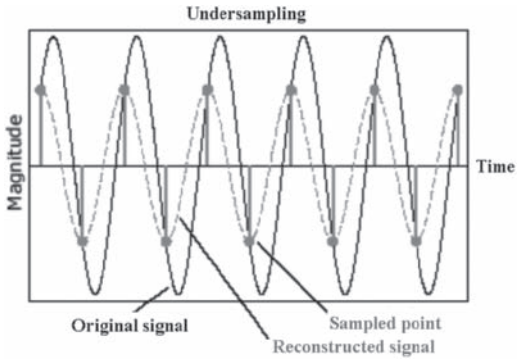


Figure 2-12 Aliasing examples in the spatial domain. The top figure shows an example in one dimension, where original signal is shown along with sampled points and the reconstructed signal. The bottom set of figures show a 2D input image signal. The top left shows the original signal and the remaining three show examples of the signal reconstruction at different sampling resolutions. In all cases, the output does not match the input because the sampling resolution falls below the Nyquist requirement.

If the pattern is not sampled at a rate that is at least twice the rate of repetition of the pattern, a different pattern will result in the reconstructed image. In the image shown in Figure 2-13, a black-and-white pattern is shown changing at a perfectly regular rate. The pattern appears rotated with a regular sampling grid superimposed on

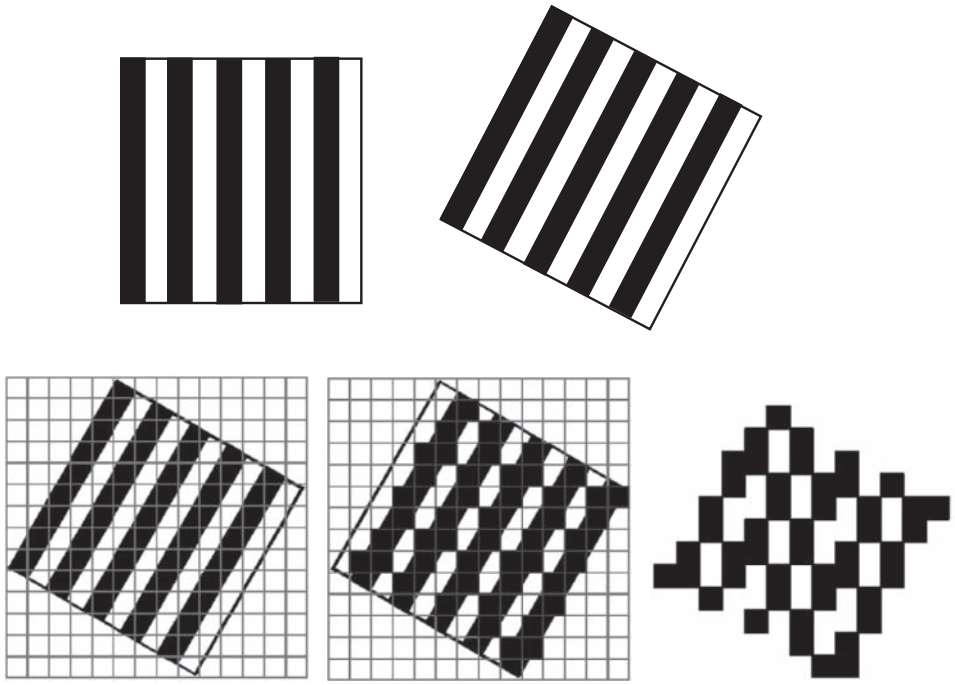


Figure 2-13 Moiré pattern example. The vertical bar pattern shown on the top left is rotated at an angle to form the input signal for sampling. The bottom row illustrates the sampling process. A sampling grid is superimposed to produce a sampled output. The grid resolution corresponds to the sampling resolution. The middle figure in the second row shows the sampled values at the grid locations and the right figure shows just the sampled values. Although the input has a pattern, the output pattern is not like the original.

it. Sampling the input at regular intervals produces the output shown on the right. You can see that the output pattern does not resemble the rotated vertical bars in the input. This is just a simple illustration of the Nyquist theorem. We would be able to reconstruct the image faithfully if we sampled more densely—more than twice the repetition of the pattern.

Moiré patterns occur in digital but not analog photography because digital photography is based on discrete samples. If the samples are taken "off beat" from a detailed pattern in the subject being photographed, aliasing problems are seen in the resulting pattern. An example of a real photograph is shown in Figure 2-14. Sometimes aliasing in digital images manifests itself as small areas of incorrect colors or artificial auras around objects, which can be referred to as *color aliasing*, *moiré fringes*, *false coloration*, or *phantom colors*. To understand what causes this phenomenon, you have to know something about the way color is perceived and recorded in a digital camera, which is explained in Chapter 4.

Figure 2-14 Moiré patterns in images—the left image shows aliasing problems due to texture patterns on the tablecloth and the woman's clothes. These artifacts are minimized when the image is passed through a suitable low-pass filter as shown in the right image.

5 FILTERING

The sampling theorem states sampling requirements to correctly convert an analog signal to a digital signal. Knowing the input analog frequency range can enable this conversion. In practice, however, the range of frequencies needed to faithfully record an analog signal is not always known beforehand. As a result, a universal sampling rate cannot be derived. Nevertheless, engineers can often define the frequency range of interest depending on the signal. For instance, the human voice does not contain frequencies beyond 4 kHz because of the physical or structural limitations of the larynx. However, when voice is recorded, the signal coming into the microphone does contain higher frequencies, which are caused by noise from the environment, noise from the microphone, or even resonance caused by sound waves bouncing off objects in the environment. The maximal frequency content of the signal to be recorded might be considerably higher than 4 kHz. In such cases, adding a *filter* prior to sampling ensures that the frequencies above a certain cutoff limit are eliminated from the signal. For recording human voice, an analog filter with a 4 kHz cutoff frequency is introduced in between the microphone and the recording device. The resulting signal is then sampled at 8 kHz.

It should also be noted that sampling alone is not physically realizable, but is always accompanied by filtering. In images, light falling on a photoreceptor is averaged over the area of the photoreceptor during a small amount of time. Whether filtering is done prior to sampling, or as a postprocess is not important, as these two operations are convolutions and, therefore, commute. Analog filtering techniques are commonly used to capture a variety of commonly used signals such as audio and images. However, in the digital world, digital data manipulations also require filters. Next we discuss digital filters.

5.1 Digital Filters

In signal processing, the function of a filter is to remove unwanted parts of the signal, such as random noise and undesired frequencies, and to extract useful parts of the

signal, such as the components lying within a certain frequency range. There are two main kinds of filters: *analog* and *digital*. They are quite different in their physical makeup and in how they work. An analog filter uses analog electronic circuits made up from components such as resistors, capacitors, and operational amplifiers (op-amps) to produce the required filtering effect. Such filter circuits are widely used in applications such as noise reduction, video signal enhancement, graphic equalizers in hi-fi systems, and many other areas. A digital filter, on the other hand, uses digital numerical computations on sampled, quantized values of the signal. The processing might be done on a general-purpose computer such as a PC, or a specialized digital signal processor (DSP) chip. These calculations typically involve multiplying the input values by constants and adding the products together. Figure 2-15 shows the basic setup of such a system.

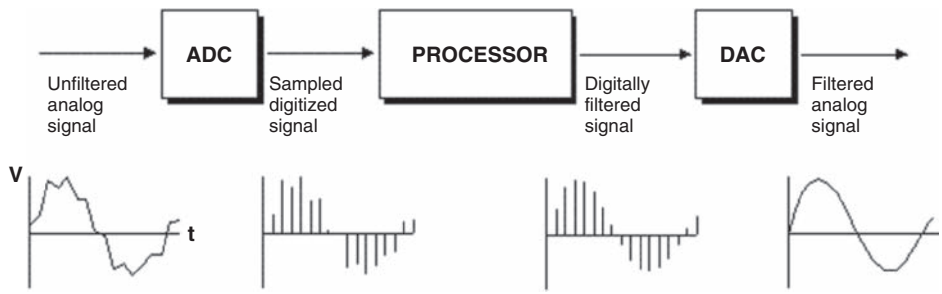


Figure 2-15 Digital filters—A digital filter takes a digital signal as input and produces another signal with certain characteristics removed. In the bottom row, the noisy analog signal is digitized and filtered.

Just as digital signals are preferred, so are digital filters. Digital filters have many advantages over analog filters.

- A digital filter is *programmable*, that is, a program stored in the processor's memory determines its operation. This means the digital filter can easily be changed without affecting the circuitry (hardware). An analog filter can only be changed by redesigning the filter circuit.
- Digital filters are easily *designed*, *tested*, and *implemented* on a general-purpose computer or workstation.
- Digital filters can be combined in parallel or cascaded in series with relative ease by imposing minimal software requirements. That is not the case with analog circuitry.
- The characteristics of analog filter circuits (particularly those containing active components) are subject to drift and are dependent on temperature. Digital filters do not suffer from these problems, and so are extremely *stable* with respect both to time and temperature.
- Unlike their analog counterparts, digital filters can handle *low-frequency signals* accurately. As the speed of DSP technology continues to increase, digital filters are being applied to high-frequency signals in the RF (radio frequency) domain, which in the past was the exclusive preserve of analog technology.

- Digital filters are more *versatile* in their ability to process signals in a variety of ways; this includes the ability of some types of digital filter to adapt to changes in the characteristics of the signal.

Both digital and analog filters can be classified into three categories: low-pass filters, band-pass filters, and high-pass filters. Low-pass filters remove high-frequency content from the input signal. Such filters are used to avoid aliasing artifacts while sampling. High-pass filters, on the other hand, remove the low-frequency content and are used to enhance edges and sharpen an image. Band-pass filters output signals containing the frequencies belonging to a defined band. Such filters are commonly used in "sub-band" filtering for audio and wavelet theory and are described in the subsequent chapters 7, 8 and 9 in the compression part.

5.2 Filtering in 1D

A 1D signal is normally represented in the time domain with the x-axis showing sampled positions and the y-axis showing the amplitude values. All filtering techniques change the time domain signal, but because filters can be defined by their frequency-domain effects, a better understanding can be gained analytically and graphically by studying the effects filters have in the frequency domain. In Figure 2-11, we showed two signals in the time domain and their Fourier transforms. These two signals $x_1(t)$ and $x_2(t)$ have been added together to form the input signal as shown in Figure 2-16. The Fourier transform of this input signal shows that there are peaks at two frequencies, $X_1(f)$ and $X_2(f)$, suggesting that the time domain signal is composed of two frequencies. We refer the reader to Section 6 of this chapter for a more thorough understanding of the frequency domain and frequency transforms.

When the input signal is passed through a filter, the output signal produced changes depending on the filter's response. This results in changing frequency domain characteristics of that signal. Filters always remove frequencies all together or lessen the contribution individual frequencies have. For the signal $x_1(t) + x_2(t)$, the Figure 2-16 shows the effects of low- and high-pass filters. The middle row shows the output generated when passed through a low-pass filter. Here the filter is designed to allow lower frequencies, such as $X_1(f)$ to remain while the higher frequency $X_2(f)$ is removed. In case of low-pass filters, all frequencies below a threshold or cutoff are allowed to remain, whereas frequencies above the cutoff frequency are removed. This threshold value depends on the filter design characteristics, which is shown graphically as $H(f)$ in the center boxes. A high-pass filter, on the other hand, removes all the lower frequencies in the signal, allowing only the higher frequencies above a threshold level to pass through the filter.

5.3 Filtering in 2D

In the case of 2D signals, such as images, filters are an important part in digital image processing. Digital image signals are contaminated with interference, noise, and aliasing artifacts in the capture process. Here, filters are used for image separation and

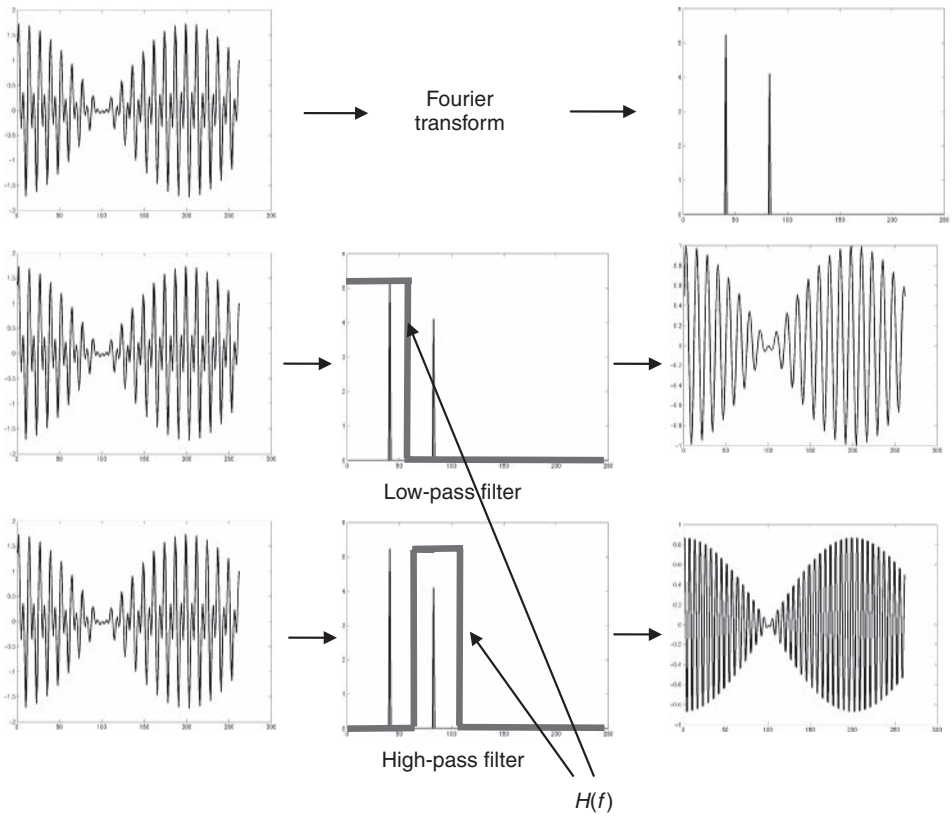


Figure 2-16 The top row shows the two signals from Figure 2-11 composited together to form the signal $x_1(t) + x_2(t)$ and its transform. The center and bottom rows show the effect of a low-pass filter and a high-pass filter on the composite signal. The output produced contains frequencies in accordance with the filter's response.

restoration. Moreover, during content creation and printing the sample resolution may change when images are resized. This image resizing or resampling process introduces aliasing in images as discussed in Section 4.1. Low-pass filters help remove aliasing in images, whereas high-pass filters are used in certain image processing operations such as image sharpening.

An image is represented by pixels whose sample positions are defined along the *x* and *y* directions. Just as in the 1D case, the 2D image case can be represented in the frequency domain by frequencies in 2D. In the 2D case, we have frequencies changing in two directions corresponding to changes along *x* and along *y*. Again, refer to Section 6 for a more thorough understanding of the frequency space transforms. Figure 2-17 shows an input image and the frequencies present in the image in the top row. When this image is passed through a filter, the output generated depends on the frequency response of the filter. In the figure, we show the effect of both low- and high-pass filters.

The reason for the changing numbers in the second column of Figure 2-6 should now be made clear. The sampling rates mentioned are specific to the signal being

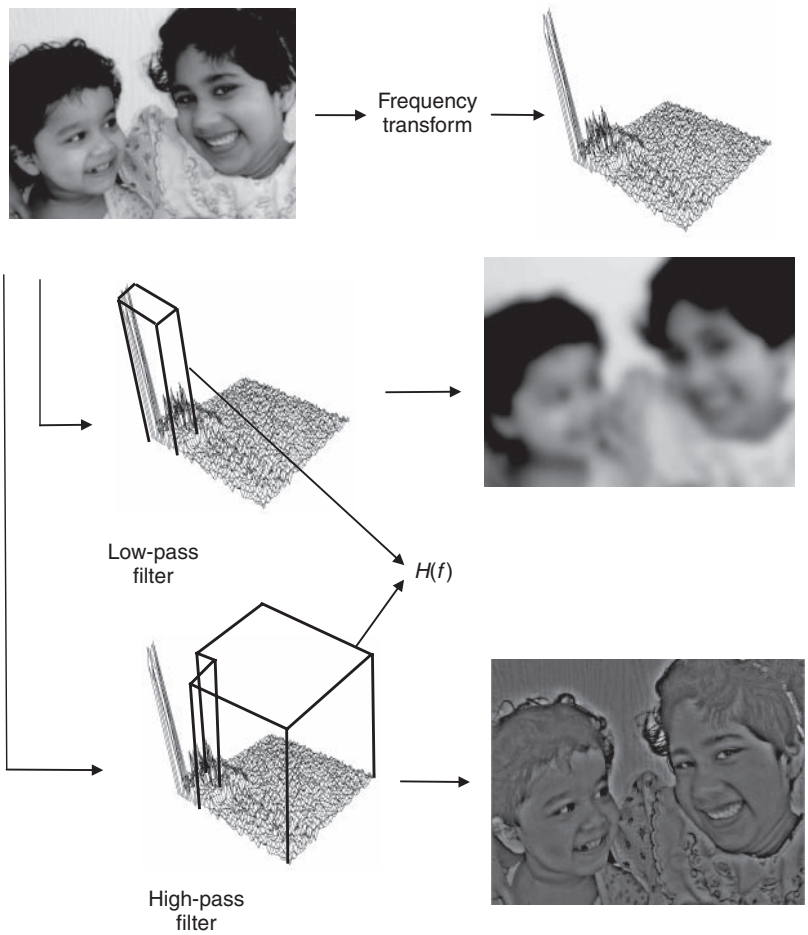


Figure 2-17 Digital filtering on images—The top row depicts an image and its frequency transform, showing frequencies in two dimensions. The middle row shows the effect of a low-pass filter. Here, only the frequencies inside the box are used to reconstruct the output image. The bottom row shows the effect of a high-pass filter.

digitized. Earlier, we discussed the choice of the 8 kHz frequency value to sample human voice. Similarly, the human ear can detect sound across the frequency range of 20 Hz to 20 kHz. Components higher than 20 kHz cannot be detected and there is no need to capture them in the digitized signal. But if they are present during the sampling process, the sampled signal could be corrupted by aliasing. Therefore, prior to sampling, frequency components above 20 kHz are removed by a band-pass or low-pass filter, and the resulting signal is sampled at 40 kHz. In practice, though, the sampling rate is typically set at 44 kHz (rather than 40 kHz) to avoid signal contamination from the filter roll-off. It is not possible to realize a low-pass filter with a cutoff of *exactly* 40kHz. This is the case with "ideal filters," which remove all unwanted

frequencies, preserving the remaining frequencies exactly. In the frequency domain, such ideal filter responses would resemble a finite rectangle, as illustrated in the top half of Figure 2-18. These ideal characteristics are only at best approximated by real digital circuits and software dealing with digital signals. The design of filters is based on a compromise between deviation from ideality versus complexity (and cost). Figure 2-18 shows ideal low-pass, band-pass, and high-pass filter transfer functions and what the corresponding practical implementations could look like. These practical implementations do give rise to artifacts in the filtered signal.

We have talked about different filters so far; we now discuss a specific type of digital filter, subsampling.

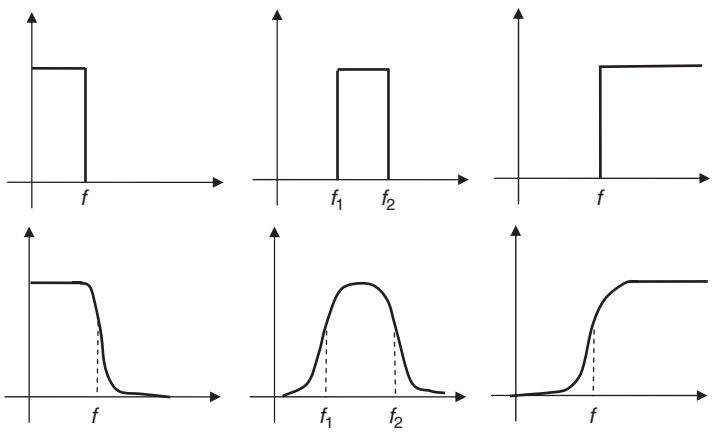


Figure 2-18 Filter responses for low-, band-, and high-pass filters are shown in each row. The top row corresponds to desired ideal responses; the bottom row shows what practical implementations give.

5.4 Subsampling

Filtering data prior to sampling is, thus, a practical solution to acquiring the necessary quantity of reliable digitized data. The cutoff frequency of the filter depends largely on the signal being digitized and its intended usage. However, once in the digital domain, there is frequently a need to further decrease (or increase) the number of samples depending on the bandwidth requirements, storage capacity, and even content creation requirements. For example, even though an image is captured at a high resolution, you might need to appropriately downsize it to incorporate it in a document, advertisement, and so on. Such post digitization sampling adjustments are achieved by a process called subsampling. Subsampling an original signal by n corresponds to keeping every n^{th} sample and removing the rest from the original signal, as illustrated in Figure 2-19. Consequently, the size of data is also reduced by a factor of n.

Although subsampling does achieve the goal of reducing the signal size or bit rate, it can certainly result in aliasing problems. For instance, if a continuous signal $x(t)$ is bandlimited to 4 kHz (such as human voice) and is sampled at 10 kHz, it will

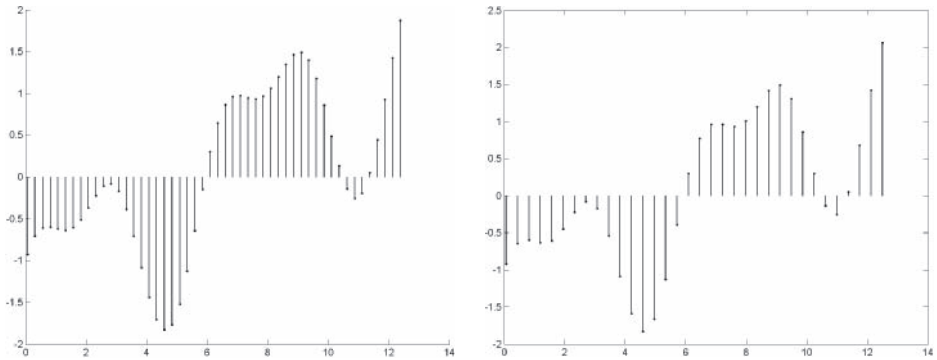


Figure 2-19 Original signal shown on the left, subsampled by 2 producing signal on the right. The number of samples on the right is half the number of samples on the left.

result in a good-quality signal because it is sampled above its Nyquist sampling frequency of 8 kHz. Subsampling this signal now by 2 is equivalent to sampling the original signal at 5 kHz, which is below the Nyquist sampling frequency, will surely cause aliasing effects. The same reasoning can be extended to 2D, where a high-definition image of, for example, 1024 × 1024 pixels is scaled down to 256 × 256 by subsampling. This results in aliasing. An example is illustrated in Figure 2-20. Here, the original image is shown on the top and the subsampled images are shown in the middle row (subsampled by 2) and bottom row (subsampled by 4).

The high-frequency areas, such as the texture on the cloth, do not appear as smooth as the original because of aliasing problems. This can be fixed by introducing a digital low-pass filter (instead of an analog low-pass filter) prior to subsampling. The digital low-pass filter does the same job on digital data as its analog equivalent does on analog data. Such digital filters are commonly used when scaling images, and result in better quality images. The digital low-pass filtered images are shown on the two right images in the middle and bottom rows.

6 FOURIER THEORY

This chapter and signal processing in general uses Fourier theory and the frequency domain representation. Hence, it is appropriate to dedicate this last section to explain the intuition and the mathematical formulation behind the theory. Fourier theory, developed during the 1700s by Jean-Baptiste Joseph Fourier, is widely used in solving problems in science and engineering. It proposes that any continuous periodic function $f(t)$ can be decomposed into or represented by a weighted combination of sine and cosine waves. The Fourier transform is the function that describes this interpretation. Mathematically

$$f(t) = \sum_{i=0}^{i=\infty} A_i \times \sin(i\omega t) + \sum_{j=0}^{j=\infty} B_j \times \cos(j\omega t)$$

Figure 2-20 The first row shows the original image signal, the second row shows the image subsampled by 2, and the third row shows the image subsampled by 4. In the subsampled cases, two images are shown: The left image is obtained without filtering prior to subsampling and the right image is obtained by filtering prior to subsampling. Aliasing effects are clearly seen on the subsampled images to the left, which do not make use of any filtering.

Fourier theory is mathematically involved, but here we try to give an intuitive understanding to some simple holistic concepts behind the theory. The sine and cosine functions defined over the fundamental frequencies (multiples of $\omega = 2\pi/\mathrm{T}$, for some periodic T) are known as basis functions, while A_i and B_i are known as the frequency coefficients. The preceding equation suggests that the function $f(t)$ can then be completely described in terms of its frequency coefficients, that is, given $f(t)$, the A_i and B_i coefficients are well defined and vice versa—given A_i and B_i, the function $f(t)$ is well defined. The function $f(t)$, which is represented by an amplitude at time t, is the time domain representation of the function, while the corresponding A_i and B_i are the frequency domain representations of the function $f(t)$.

The preceding equation is shown expressed in terms of both the sine and cosine (Fourier transform), but it can be expressed in terms of just the cosines (cosine transform) or just the sines (sine transform). All of these families of transforms are attempting to define a function in a coordinate space of functions where $\sin\omega t$, $\sin 2\omega t$, $\sin 3\omega t$ . . . are the basis and the corresponding A_0, A_1, A_2 . . . are the coefficients. This is akin to a point representation in terms of its coordinates in 3D space $= xi + yj + zk$, where unit vectors i, j, k along the x, y, and z axes are expressed in terms of coordinates or weights (x, y, z). For the sake of simplicity, let us assume that we are dealing with cosine transforms. The preceding equation then becomes

$$f(t) = \sum_{i=0}^{i=\infty} B_i \times \cos(i\omega t)$$

where the frequency coefficients are computed as

$$B_i = \int_{-\infty}^{+\infty} f(t)\cos(i\omega t)dt$$

Thus, B_i gives the importance or weight that i^{th} frequency has in defining the function. i is an integer and, thus, every frequency ω_i is an integral multiple of the basic frequency $\omega_1 = 2\pi/\mathrm{T}$. So if the preceding integral evaluates to a nonzero value, it shows that the frequency ω_i is present in the function. Correspondingly, if it evaluates to zero or low value, it goes to show that the frequency ω_i is not present in the function.

Figure 2-21 shows examples of frequency space representations in 1D. The left column shows sample functions in the time domain. As you go down the column, the functions are shown to have increasing variations. The right column shows the weights of frequencies B_i. It can be seen that the first function is a constant value with no variation, and correspondingly the frequency domain representation has a non-zero weight for the zeroth frequency. The next two functions represent cosines of increasing frequency and their frequency transforms show nonzero weights at those frequencies. The last two functions are a result of combining more than one fundamental frequency.

The same understanding can now be extended to 1D signals in the digital domain. In the digital world, signals are represented as discrete samples. The preceding continuous domain equation can be modified to the discrete representation, as

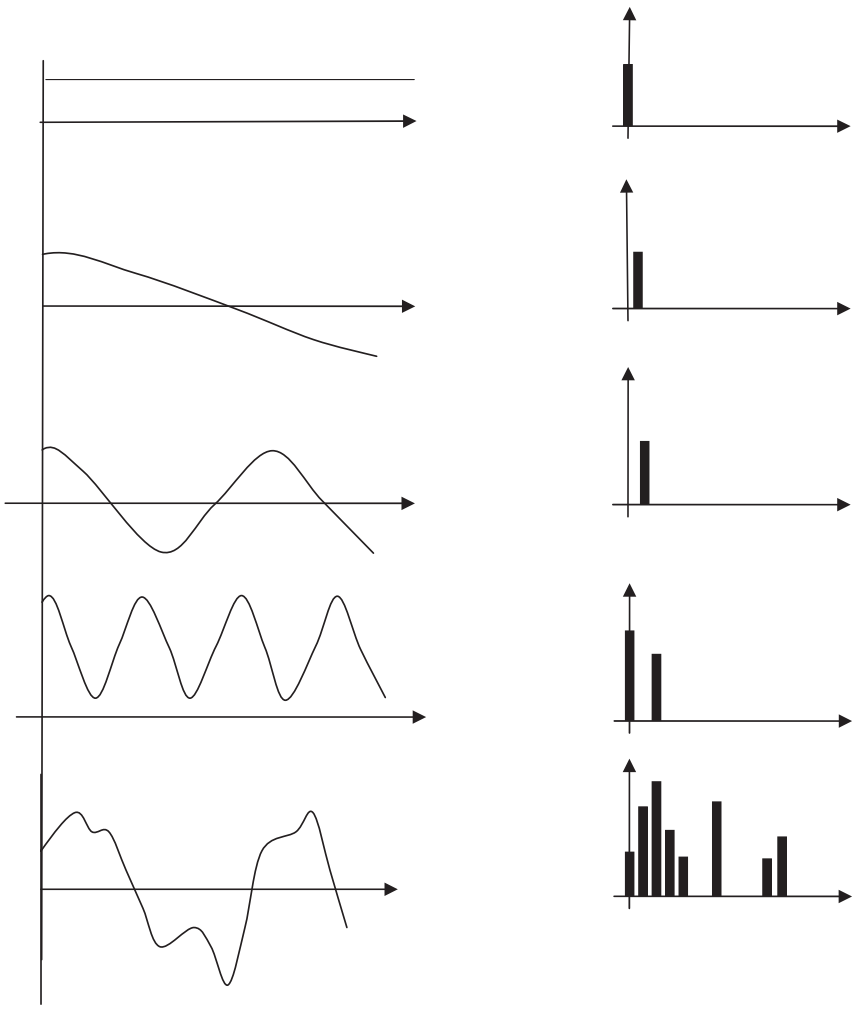


Figure 2-21 The left column shows examples of time domain functions. The right column shows their frequency transforms. For each transform, the nonzero coefficients are shown. Although the first three functions are simple, and have only one fundamental frequency, the last two are composed of more than one.

shown in the following formulation. If we assume $f_s(n)$ is the corresponding sampled function defined for n in the range [0,N-1], then

$$f_s(n) = \sum_{i=0}^{i=(N-1)} A_i \times \sin\left(\frac{i(2n + 1)\pi}{2N}\right) + \sum_{j=0}^{j=(N-1)} B_j \times \cos\left(\frac{j(2n + 1)\pi}{2N}\right)$$

Here, both the sine and cosine functions are defined as discrete signals, while A_i and B_i represent the coefficient values for the i^{th} fundamental frequencies. If we assume

for the sake of simplicity that the function is approximated only by the cosine transform, then the preceding equation reduces to

$$f_s(n) = \sum_{i=0}^{i=(N-1)} B_i \times \cos\left(\frac{i(2n + 1)\pi}{2N}\right)$$

where B_i can be defined to be

$$B_i = \sum_{n=0}^{n=(N-1)} f_s(n) \times \cos\left(\frac{i(2n + 1)\pi}{2N}\right)$$

Figure 2-22 shows examples of commonly used digital signals and their corresponding frequency transforms.

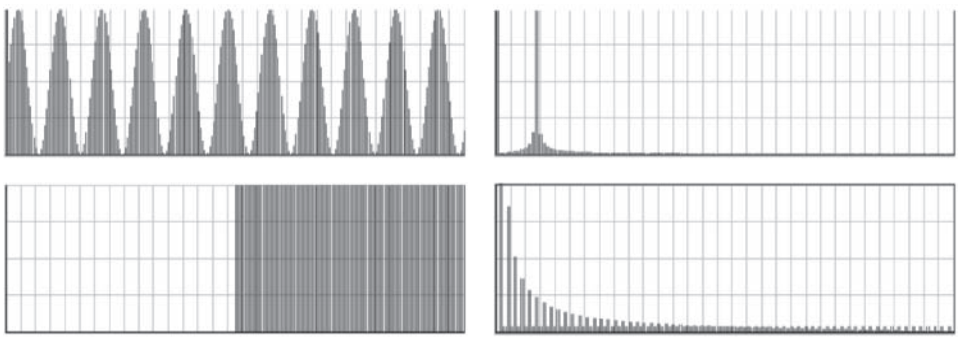


Figure 2-22 The discrete frequency transform operates on discrete signals. The left column shows a sinusoid and a step function. The right column shows the weights of the frequency coefficients. While the sinusoid has a distinct peak, the step function needs to be approximated by all frequencies.

The previous explanations can be extended to two dimensions. Here, the input signal is an image, or image block, and the basis functions are now in 2D. These basis functions represent the fundamental image frequencies and can vary in two directions, the x direction and the y direction. They can be mathematically expressed as follows:

$$\begin{aligned} f(x,y) = {} & B_{00} + B_{01} \times \cos(\omega y) + B_{02} \times \cos(2\omega y) \\ & + B_{10} \times \cos(\omega x) + B_{11} \times \cos(\omega x)\cos(\omega y) + \ldots \\ & + B_{20} \times \cos(2\omega x) + B_{21} \times \cos(2\omega x)\cos(\omega y) + \ldots \end{aligned}$$

Or

$$f(x,y) = \sum_{i=0}^{i=n} \sum_{j=0}^{j=n} B_{ij} \times \cos(i\omega x)\cos(j\omega y)$$

where n is the number of fundamental frequencies that approximate the function and the frequency coefficients B_{ij} are computed as

$$B_{ij} = \int_{-\infty}^{+\infty} f(x,y) \cos(i\omega x) \cos(j\omega y)\, dxdy$$

In the formula the term cos(iωx)cos(jωy) represents a frequency in two dimensions, termed ω_{ij} where i and j are integers. B_{ij} gives the importance, or weight, that frequency ω_{ij} has in defining the function. So, if the preceding integral evaluates to a nonzero value, it shows that the frequency ω_{ij} occurs in the image. Conversely, if it evaluates to zero or a low value, it goes to show that the frequency ω_{ij} is not present in the image. This can be figuratively explained as shown in Figure 2-23. Here, the first image has vertical lines, which means that the pixels are changing only in the horizontal direction. There is only one dominant frequency present, which is shown by a peak in the frequency domain. If the thickness of the lines is allowed to change in the image (or spatial domain), the peak in the frequency domain moves to some other location. The second image has a similar pattern but in the horizontal direction and its pixels are changing only in the vertical direction. The corresponding frequency domain representation is now reversed as shown. The third image is a real image with pixels changing in both directions. Correspondingly, the frequency space representation shows many frequencies; the lower ones are more dominating than the higher ones.

7 EXERCISES

1. **[R03]** Suppose you have an 8-bit A/D converter that has a full-scale input range of −2V to +4V. When a particular voltage is applied, the computer records the hex number A2. Assuming a perfect calibration, answer the following questions:
   - What output voltage does this value correspond to?
   - What is the digitization (quantization) error in the voltage?
   - By how much percent would this error change if 12 bits were used to approximate the output instead of 8 bits?
2. **[R03]** The bandwidth of a speech signal is from 50 Hz through to 10 kHz and that of a music signal is from 15 Hz through to 20 kHz. You want to digitize these signals using the Nyquist criterion.
   - What is the bit rate produced for the speech signal if 12 bits are used per sample?
   - Perform the same for the music signal when 16 bits per sample are used.
   - How many megabytes of storage do you need for 10 minutes of stereophonic music?
3. **[R03]** Suppose you scan a 5 × 7 inch photograph at 120 ppi (points per inch or pixels per inch) with 24-bit color representation. For the following questions, compute your answers in bytes.

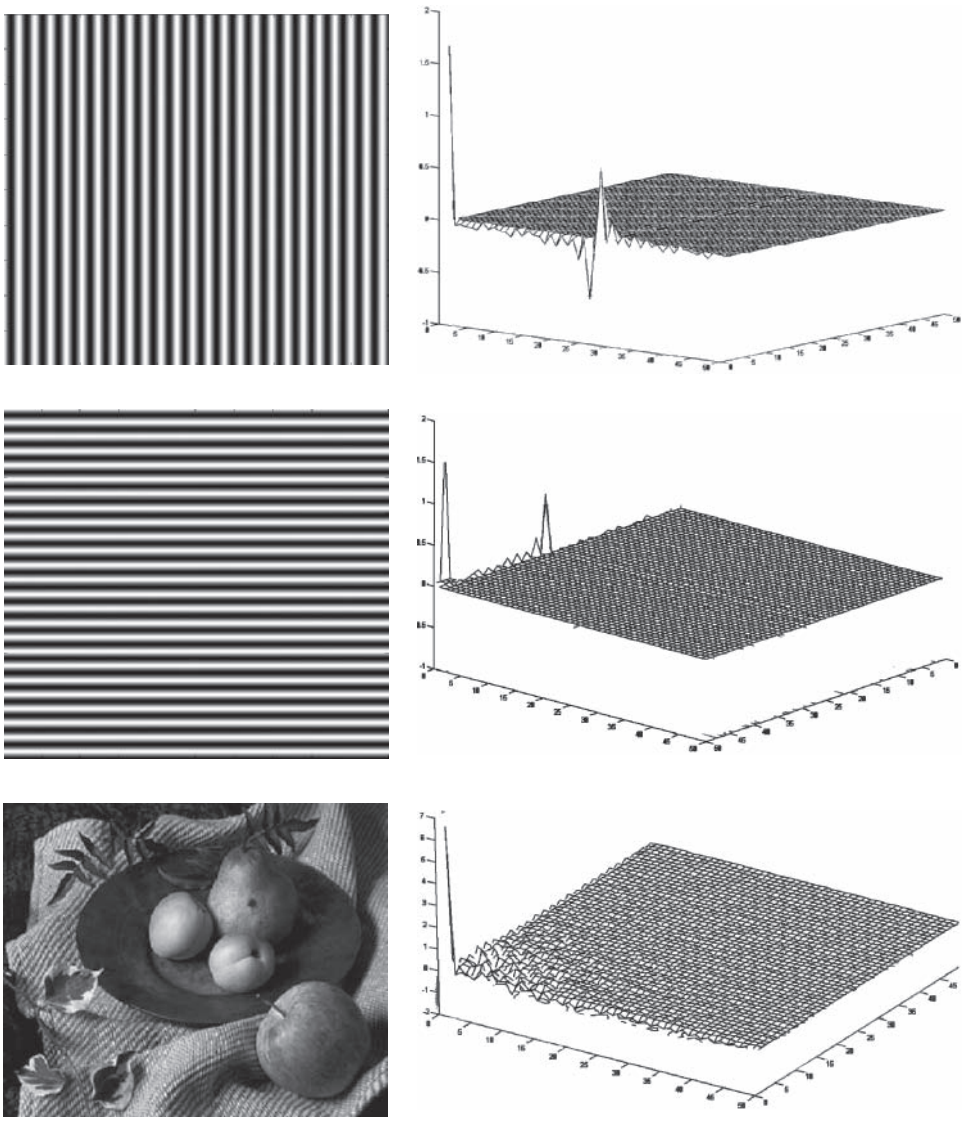


Figure 2-23 Frequency transforms in 2D. On the left, input images are shown and on the right, their frequency transforms are shown. The first image, which consists of vertical lines, has pixel variations in the horizontal direction only. This is shown by one dominant frequency in the horizontal direction in the frequency domain. The second image has a lot of frequencies. In this case, the lower frequencies are more dominant than the higher ones.

- How big is the file?
- If you quantize the full-color mode to use 256 indexed colors, how big is the file?
- If you quantize the color mode to black and white, how big is the file?

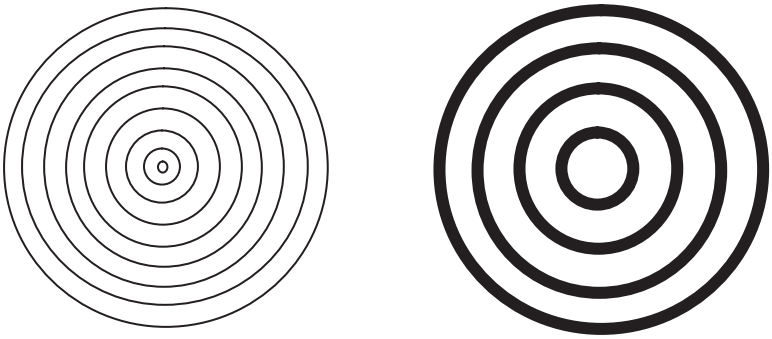
4. **[R03]** A movie on a video server occupies 4 GB. You want to download it on your local machine.
   - What should the link speed be if you want to download it in 30 minutes? Assume that the local and remote computers are fast enough to have negligible read/write time.
   - If you were to get the same movie via a slow 56K modem connection of today, how long will it take you? (Assume that the modem communicates at a rate of 56 kilobits per second)
   - If you had a DSL connection at 780 Kbps, how long would it take you?
5. **[R04]** The high-frequency limit of human hearing extends to approximately 20 kHz, but studies have shown that intelligible speech requires frequencies only up to 4 kHz.
   - Justify why the sampling rate for an audio compact disc (CD) is 44.1 kHz. What is the Nyquist rate for reliable speech communications?
   - Why do you think people sound different on the phone than in person?
   - Suppose intelligible speech requires 7 bits per sample. If the phone system is designed to precisely meet the requirements for speech (which is the case), what is the maximum bit rate allowable over telephone lines?
   - CDs use 16 bits per sample. What is the bit rate of music coming off a CD? Is a modem connection fast enough to support streamed CD quality audio?
6. **[R05]** The following sequence of real numbers has been obtained sampling an audio signal: 1.8, 2.2, 2.2, 3.2, 3.3, 3.3, 2.5, 2.8, 2.8, 2.8, 1.5, 1.0, 1.2, 1.2, 1.8, 2.2, 2.2, 2.2, 1.9, 2.3, 1.2, 0.2, −1.2, −1.2, −1.7, −1.1, −2.2, −1.5, −1.5, −0.7, 0.1, 0.9. Quantize this sequence by dividing the interval [−4, 4] into 32 uniformly distributed levels. (Place the level 0 at −3.75, the level 1 at −3.5, and so on. This should simplify your calculations.)
   - Write down the quantized sequence.
   - How many bits do you need to transmit it?
   - While storing similar signals, you notice that you need to reduce the number of bits by 50%. Propose what you can do to achieve this.
   - Does your result signal have more error than the former case? Can you think of ways to reduce the error while maintaining the bit rate to a 50% level?
7. [R04] Normally, quantization intervals are equispaced, which results in uniformly quantized signals. This is the case with most signals used. However, now you decide to venture out and design a nonuniform quantizer, which results in unequal quantization intervals.
   - Explain the benefit of such a design, giving a sample application where it would be of advantage.
   - Can you think of a mathematically defined nonuniform quantization function?

8. **[R07]** Temporal aliasing can be observed when you attempt to record a rotating wheel with a video camera. In this problem, you will analyze such effects. Let us say we have a car moving, and the wheels of the car appear to be rotating at a speed different from the speed of the car (or even in a direction different from the motion of the car). Assume that the car is traveling at 36 km/hr is being captured by a video camera. The tires have a diameter of 0.4244 meters. Each tire has a white mark to gauge the speed of rotation.
   - Determine the Nyquist sampling rate (film frame rate) for the tire rotation.
   - If the film is being shot at 24 frames per second (fps), at what speed (in rotations per second) will the tires appear to be rotating?
   - At what speed do the tires appear to rotate if the frame rate is set at 12 fps? 7.5 fps? 6 fps?
   - 24 fps is fairly standard for film. What should be the frequency for the tire rotation at this sampling rate to not have aliasing? How fast can the car be traveling (in km/hr) before aliasing occurs?
9. **[R05]** You are asked to design antialiasing filters for (1) FM Radio, (2) a CD recording, and (3) a telephone system. The corresponding sampling rates are given in the table shown in Figure 2-6 in the text. What will be typical filter design requirements for these applications?
10. **[R04]** Note: you might need some theory from the next chapter to answer this question. Suppose a camera has 450 lines per frame, 520 pixels per line, and a 25 Hz frame rate. The color subsampling scheme is 4:2:0, and the pixel aspect ratio is 16:9. The camera uses interlaced scanning, and each sample of Y, Cr, Cb is quantized with 8 bits.
    - What is the bit rate produced by the camera?
    - Suppose we want to store the video signal on a hard disk, and to save space, we requantize each chrominance (Cr, Cb) signals with only 6 bits per sample. What is the minimum size of the hard disk required to store 10 minutes of video?
    - Repeat the exercise (both questions) assuming a color subsampling scheme of 4:2:2.

PROGRAMMING ASSIGNMENTS

11. **[R05]** Aliasing effects are observed for all kinds of media types. Here, we want to study aliasing effects in images similar to some of the examples seen in the text. Write a simple program to create a radial pattern of circles. A pattern of circles is centered at the center of the image and has two parameters describing it—M and N where M is the thickness of each circle in pixels and N is the difference between their successive radii in pixels.

    Here are two sample images for $M = 1$, $N = 5$, and $M = 4$, $N = 10$, respectively



Start with an original image that is 512 × 512 with an initial white background. Follow these steps:

- Create a radial pattern of black circles with M and N as properties described previously.
- Implement a subsampling routine to subsample the image by K. When you subsample by K, your image size will be 512/K by 512/K. For example, for K = 4, your image size will be 64 × 64.
- You subsampling procedure will cause aliasing artifacts observed as incompelete circles and jagged edges. Implement an antialiasing routine that will minimize such artifacts (for example, a low-pass averaging filter with a kernel of 3 × 3).

Perform the preceding steps for the following:

1. M = 1, N = 4, K = 2 and K = 4
2. M = 3, N = 10, K = 2 and K = 4
3. M = 3, N = 5, K = 2 and K = 4

Play around with the parameters and give a brief write-up on how aliasing effects vary with M, N, and K. Specifically include comments on when one parameter is allowed to vary while the others remain constant.

12. **[R06]** In the previous programming assignment, we studied aliasing effects. Here, you will adapt the same program to understand moiré patterns. The previous assignment took two parameters as input M and N, where *M* is the thickness of each circle in pixels and *N* is the difference between their successive radii in pixels. Modify this program to take another set of parameters DX and DY. DX and DY specify the offsets relative to the center at which you will draw another set of concentric circles defined by M and N.
    - Modify the code to take DX, DY as input and show two sets of concentric circles superimposed. The center of the first set is the image center and the center of the second set is offset DX, DY relative to the first center.
    - Observe the patterns perceived by varying DX, DY, M, and N. For a fixed M and N, if DX and DY are allowed to vary, how do the patterns change? You should add user interfacing to allow you to change DX, DY

interactively—that is, you should be able to interactively superimpose the two sets of concentric circles for a given M, N and see the moiré pattern effects.

- Comment on your observations of how these patterns are formed by varying M, N, DX, DY.
- Write an antialiasing filter to decrease the effect of the observed moiré patterns.

13. **[R06]** In this assignment, you will study temporal sampling and aliasing effects in a sequence of images of a revolving wheel. You will have to implement a program that will take two parameters as input—*m* and *n*, where *m* is the rotation speed of the wheel (*m* revolutions per second) and *n* is the refresh rate for display (*n* frames per second). The following is a step-by-step description of what you are required to implement:
    - Write a program that displays at the refresh rate. This means you will have to draw an image, clear it, and draw the next image every $1/n^{\text{th}}$ second.
    - Implement a way to draw a wheel rotated about its center according to its speed at every frame.
    - What happens for a constant *m* as you vary *n* and vice versa?
    - Suppose that the value of *m* is 20 revolutions per second. What should be the value of *n* such that the wheel appears not to move?

This page intentionally left blank

CHAPTER 3

Media Representation and Media Formats

Media is represented in various forms—text, images, audio, video, and graphics. Images are commonly used to capture and represent a static visual snapshot of the world around us. The world, however, is not static, but continuously changes in time. To record this change, we need to capture the time evolution of changing images (video) and sound (audio). Graphical illustrations and animation help to visually convey the changing information. The digital media forms need to be represented and stored so that they can be viewed, exchanged, edited, and transmitted in a standard manner. This chapter deals with the representation of digital media and commonly used media formats. We also talk about the evolution of analog media formats and how they still influence the current digital formats. It is assumed that you are already familiar with the concepts involving the analog-to-digital conversion process covered in the Chapter 2.

1 DIGITAL IMAGES

When we speak of images, we normally refer to "still" images. Images by themselves are used in various forms for a variety of applications. These might be photographs, gray or color, or used with text in documents. A fax is another image representation used in communication. Images can be combined to create interesting applications involving mosaics, panoramas, and stereo imagery. Furthermore, images form the basic elements of video. In Figure 3-1, an example shows a black-and-white image, a color image (to see the color image, reference the color insert in this textbook), and a panorama obtained by "stitching" images together.

Figure 3-1 Example showing different types of images. The upper-left image shows a gray-level image with each pixel represented by a single channel, whereas the upper-right image shows the same image in a color representation where each pixel is represented by three color channels. The bottom image shows a large mosaic created by combining different images. See the color insert in this textbook for a full-color version of this image.

1.1 Digital Representation of Images

All images are represented digitally as pixels. An image is defined by image width, height, and pixel depth. The image width gives the number of pixels that span the image horizontally and the image height gives the number of lines in the image. Each pixel is further represented by a number of bits, which is commonly called the pixel depth. The pixel depth is the same for all pixels of a given image. The number of bits used per pixel in an image depends on the color space representation (gray or color) and is typically segregated into channels. The total number of bits per pixel is, thus, the sum of the number of bits used in each channel. For instance, in grayscale images, the gray-level value is

encoded on 8 bits for each pixel. In color images, each R, G, B channel may be represented by 8 bits each, or 24 bits for a pixel. Sometimes a additional fourth channel called the alpha channel is used. It is discussed in more detail in the text that follows. When the alpha channel is present, it is represented by an additional 8 bits, bringing the total bit depth of each pixel to 32 bits. The size of the image can, thus, vary depending on the representations used. For example, a color image has a width of 640 and height of 480. If the R, G, B color channels are represented by 8 bits each, the size of color image $= 640 \times 480 \times 3 \times 8 = 7.37$ Mbits (921.6 Kbytes). If this were a gray image, its size would be $640 \times 480 \times 8 = 2.45$ Mbits (307.2 Kbytes).

The number of channels typically ranges from one to four. One channel per pixel produces a grayscale image, and the number of bits in this case gives the possible range of the grayscale values. Three channels per pixel produce a color image. In the case of a typical color image, three channels represent the red, green, and blue components. Color representation and the associated theories are dealt with in Chapter 4. Images that are captured using digital devices, such as cameras and scanners, are normally colored. However, when these color or continuous tone images are printed, printing technologies often prefer to print halftone images where the number of colors used is minimized to lower printing costs. The halftone printing process creates ranges of grays or colors by using variable-sized dots. The resolution of halftone images is measured in terms of the frequency of the halftone dots, typically in dots per inch, instead of pixels. The dots control how much ink is deposited at a specific location while printing on paper. Figure 3-2 shows examples of halftone images. Varying their size and density creates the illusion of gray or continuous color. For a process color image, four halftone channels are used: cyan, magenta, yellow, and black—one for each ink used.

Figure 3-2 Halftone image examples of the original images in Figure 3-1. Ranges of grays are perceived by using variable-sized dots. The size of dots is smaller for the image on the left.

Sometimes, an additional channel, called the alpha channel (or α channel, or α matte), is also used. In such cases, gray-level images have two channels (one gray channel and one alpha channel), whereas color images have four channels (one for R, one for G, one for B, and one for alpha). The alpha channel suggests a measure of the

transparency for that pixel value and is used in image compositing applications, such as chroma keying or blue screen matting, and in cell animation. The alpha channel typically has the same bit depth as all the other channels, for example, 8 bits for the alpha channel, resulting in each pixel having 32 bits (8 bits for R, G, B and 8 bits for alpha). The 8-bit alpha channel value for each pixel associates the degree of importance of each pixel in a compositing operation. A value of 0 indicates that the pixel will not be composited, a high value of 255 (for 8-bit alpha values) indicates that the pixel will be entirely composited, while an intermediary value indicates a percentage used in the composition. Intermediary values are normally required for pixels at the boundary of the content being composited so that it blends into the background image without aliasing effects. Figure 3-3 shows an example of how the alpha channel works. Here, two images—background and foreground—have been composited by making use of the foreground image's alpha channel.

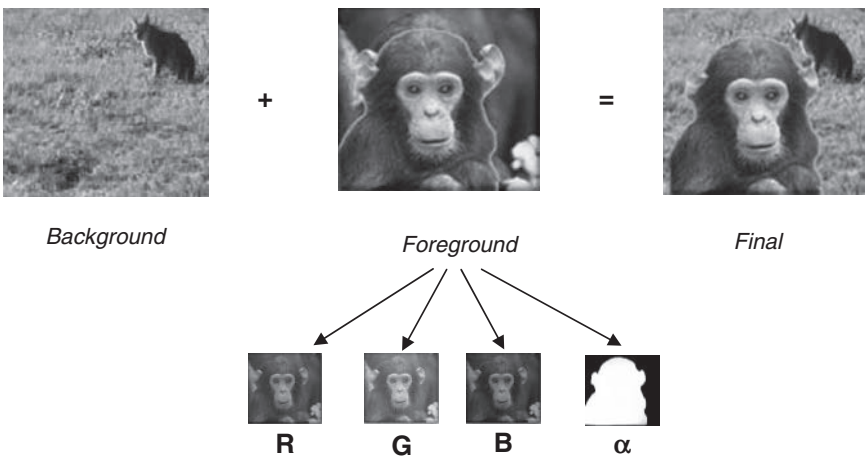


Figure 3-3 Usage of alpha channels for compositing. The foreground color image is represented by three color channels and one alpha channel. The alpha channel is used to composite the pixels of the foreground with the background image, producing the final image on the far right. See the color insert in this textbook for a full-color version of this image.

In the example in Figure 3-3, we have used 8 bits per channel in the image, which is a commonly used value for images captured by digital cameras and multimedia content creation tools. However, this number might be higher or lower for certain applications, depending on the intended use of images. For example, images created in digital film postproduction use 16 bits per channel, with a total of 48 bits per pixel (or 64 if the alpha channel is present). This is needed because all the work is done in high-color resolution, which then gets ultimately mapped to other lower imaging and video formats used in DVD distribution, broadcast, or transfer to film. On the other hand, video game platforms have limited game/texture memory, which is used to load all the assets used by the game engine to create imagery. Such platforms use 16 bits per pixel, where the 16 bits are divided as 5 bits per color channel, and 1 bit for the alpha channel.

1.2 Aspect Ratios

Image aspect ratio refers to the width/height ratio of the images, and plays an important role in standards. Different applications require different aspect ratios. Some of the commonly used aspect ratios for images are 3:2 (when developing and printing photographs), 4:3 (television images), 16:9 (high-definition images), and 47:20 (anamorphic formats used in cinemas). By importing an image into digital image processing software such as Adobe Photoshop, you could crop the image to the desired width and height for documenting purposes. Figure 3-4 shows examples using these standard image formats.

The ability to change image aspect ratios can change the perceived appearance of the pixel sizes, also known as the *pixel aspect ratio* (PAR) or *sample aspect ratio* (SAR). Most image capture instruments have the same sampling density in the horizontal and vertical directions. The resulting image, therefore, has square pixels even though the number of samples in the horizontal direction (width) and vertical direction (height) might vary. A square pixel has a PAR of 1:1. If the image aspect ratio is changed, the pixel aspect ratio will need to change accordingly to accommodate the change in area, thus making the image appear stretched in one direction. This problem is common with image and video applications, where images using one aspect ratio standard are viewed on television screens supporting a different format. In such cases, the image appearance changes and pixels appear to stretch. Figure 3-5 shows the original 4:3 aspect ratio image changed and mapped to a high-definition format. The change from a 4:3 standard format to a 16:9 high-definition format causes the content to stretch horizontally. The pixel aspect ratio has changed—in this case from its original 1:1 value on a standard definition to a 1.333:1 value in the high-definition case. On the other hand, if the anamorphic image is remapped and viewed on a standard definition television, the PAR is changed to 1:0.75, thus making the image appear stretched vertically. Normally, the images do not appear stretched when viewed on either monitor because of a preformatting step that is used during the conversion, which renders the PAR to be 1:1.

1.3 Digital Image Formats

Images can be acquired by a variety of devices. Most of the time, they are captured by a digital camera or a scanner, or even created by using many commercially used packages such as Adobe Photoshop, or a combination of the two. Standard formats are needed to store and exchange these digital images for viewing, printing, editing, or distribution. A variety of such formats are described in this section.

Initially, when images were not large and there was no need for compression, uncompressed formats such as *bmp* (bitmapped image) were prevalently used. In these cases, images were represented as bitmaps and stored in a binary file. Files such as these, with no compression, are called *raw image files*. Raw image files are predominantly used today when you do not want artifacts of compression in your representation and display. In addition, you might want to perform specific customized operations on images, such as convolution, filtering, dithering, and so on for which images will need to be in an uncompressed format. Normally, all the initial captured

Figure 3-4 Top: An image that uses a 4:3 aspect ratio. Middle: The same image captured in a 16:9 image format. Bottom: The same image cropped to an anamorphic format.

or scanned images are kept in their raw uncompressed formats for processing and editing. In their final form, these files are stored in a compressed format to save memory and bandwidth during network transfer. Among the most commonly used compressed storage/transmission file formats are *.jpg* for photographic-type images

Figure 3-5 Illustration of pixel aspect ratio changes. The top two images show the 4:3 image (left) converted to a 16:9 format. The pixels appear horizontally stretched. The bottom two images show the anamorphic image resized to fit the 16:9 format (left) and a 4:3 format (right). The images appear stretched vertically because of irregular sampling in both dimensions causing nonsquare pixels.

and .gif or *.png* for poster-like images. 0 illustrates some of the commonly used raw/uncompressed and compressed image formats and their uses.

The raw and compressed image formats are a result of a raster representation of images. Raster images are images stored as row of pixels and have a width and height. The images described in Section 1.1 are raster images. Today, vector image file formats are also used. Vector graphics files store image data in terms of geometric objects. The objects are specified by parameters such as line styles, side lengths, radius, color, gradients, and so on. This information can be stored in either binary or text form. If a vector graphics file is text-based, you can look at it in a text editor and read the statements that define the objects. It is possible to create a text-based vector image file by hand with a text editor or even alter an existing one. Knowledge of the grammar used and syntax of the object definition is required for this process, along with a lot of attention to detail. Vector graphics files are usually not edited by hand because drawing programs give you such powerful higher-level facilities for creating and manipulating vector graphic objects. A vector graphics file might have statements such as the ones shown in Figure 3-6 and may be stored in ASCII or binary format.

Some file formats combine vector graphics with bitmap images. We refer to these as *metafiles*. The term *metafile* evolved from attempts to create a platform-independent specification for vector graphics. The Computer Graphics Metafile (CGM), originally standardized under the International Organization for Standardization (ISO) in 1987, and evolving through several revisions, is an example of a standardized metafile format designed for

```
Object myRectangle
LineType Dotted
LineWidth 4
LineColor  0 0 0
FillColor 255 0 0
Rectangle (100,200) (200,220)
EndObject
Object myCircle
LineColor 0, 0, 0
FillColor 0 0 255
Circle (200, 200), 50
EndObject
```

Figure 3-6 Vector file description showing two objects—a red rectangle and a blue circle

cross-platform interchange of vector graphics with the optional inclusion of bitmaps. CGM files can be encoded in readable ASCII text, or compiled into a binary representation. In recent years, the World Wide Web Consortium (W3C) has supported the development of WebCGM, which is designed to incorporate the CGM vector format into Web pages using the Extensible Markup Language (XML). An alternative to WebCGM for Web vector graphics being developed by W3C is Scalable Vector Graphics (SVG). Generally, WebCGM is considered appropriate for technical graphics, and SVG is preferable for graphic arts. The table in Figure 3-7 shows a compiled feature list of the commonly used raster, vector, and metafile file formats. The Features column also mentions what types of compression are used by the image format. Please refer to Chapter 7 on Image Compression in the next section for a detailed explanation of what these compression standards involve.

File suffix	File name	File type	Features
.bmp	Windows bitmap	Uncompressed raster	Represents from 1 to 24 bits per pixel. Normally uncompressed but can use lossless run length encoding (RLE)
.pcx	Windows Paintbrush	Uncompressed/ compressed raster	Used only on Microsoft Windows platforms. Has similar features to .bmp.
.gif	Graphics Interchange Format	Compressed raster	Predominantly used on the Web. Allows 256 indexed colors and simple animations. Alpha channel supported. Uses LZW compression Proprietary to CompuServe

(Continued)

Figure 3-7 Table illustrating various commonly used file formats and the salient features each format supports

.jpg, .jpeg	Joint Photographic Experts Group	Compressed raster	For continuous tone pictures (photographs). Lossy and lossless compression supported. No alpha channel supported. Level of compression can be specified. Commonly used on the Web
.png	Portable Network Graphics	Compressed raster	Allows 1–48 bits of color. Supports alpha channel. Designed to replace proprietary .gif files. File format approved by W3C
.psd	Adobe Photoshop	Uncompressed layered raster	Used for image editing. Supports a variety of color models. Supports varying pixel bit depths Image can be organized into layers. Commonly used processing file format.
.psp	Paint Shop Pro	Uncompressed layered raster	Similar to .psd
.tif, .tiff	Tagged Image File Format	Uncompressed raster, also compressed raster	Used in traditional print graphics. Can be compressed using lossless and lossy methods of compression, including RLE, JPEG, and LZW. TIFF comes in many flavors
.fh	Macromedia Freehand	Compressed vector format	Proprietary to Macromedia, used by Flash Players. Supports animation
.cdr	CorelDRAW	Uncompressed vector format	Proprietary to Corel
.swf	Macromedia Shockwave Flash format	Uncompressed vector format	Proprietary format created by Macromedia (now Adobe). Contains vector representations and animations that can be put on the Web.
.dxf	AutoCAD ASCII Drawing Interchange Format	Uncompressed vector format	ASCII text stores vector data. Used for 2D/3D graphical images.
.ps or .eps	Postscript, or Encapsulated Postscript	Uncompressed metafile	Supports text, fonts, vectors, and images.
.ai	Adobe Illustrator	Metafile format	Proprietary format. Similar to .eps.
.pdf (portable document format)	Adobe PDF document	Compressed metafile	Supports text, fonts, and images. Commonly used document format. Supports hyperlinks. Supports authorized access.
.pict	Macintosh Quickdraw	Compressed metafile	Used predominantly on Macintosh platforms. Can use RLE or JPEG compression. Supports grayscale, RGB, CMYK, or indexed color.

2 DIGITAL VIDEO

Throughout the twentieth century, motion pictures stored on film have been the vehicle for much of our art, information, and entertainment. The medium changed to analog video stored on tapes, producing a revolution by giving the public direct access to the movies. Digital video is further altering the field by making available unprecedented levels of visual quality, distribution, and interaction.

2.1 Representation of Digital Video

Video, whether analog or digital, is represented by a sequence of discrete images shown in quick succession. Each image in the video is called a frame, which is represented as a matrix of pixels defined by a width, height, and pixel depth. The pixel depth is represented in a standardized color space such as RGB. These image attributes remain constant for all the images in the length of the video. Thus, as with all images, video has the same properties such as width, height, and aspect ratio. In addition, two important properties govern video representation: frame rate and scanning format.

The rate at which the images are shown is the frame rate. Video standards and applications do not necessarily adhere to the same frame rate. Film is displayed at 24 frames per second. Television standards use 30 frames per second (NTSC) or 25 frames per second (PAL). If the frame rate is too slow, the human eye perceives an unevenness of motion called flicker. The NTSC and PAL television formats are explained in detail in the paragraphs that follow. Both have a frame rate that is fast enough for motion to appear smooth in accordance to the persistence of human vision as well as a rate that can be synchronized for transmission.

Although digital video can be considered a three-dimensional signal—a 2D image changing over time—analog video is converted to a 1D signal of scan lines, as shown in Figure 3-8. This scan line conversion was introduced to make analog television broadcast technology work, and is central to the manner in which televisions (and all other cathode-ray tubes) display images. The electron gun(s) in a television project electrons on the phosphor screen from left to right in a scan line manner and from top to bottom successively for each frame. The phosphor screen glows at each location on a scan line creating a color at all positions on the line. The color glow fades quickly, but the frame rate ensures that electron gun(s) recolor the scan line before it fades. Scanning formats, which is an outcome of the analog technology, can be represented as interlaced or progressive. These formats are discussed in the subsections that follow.

Digital display technologies display media in a digital format. Digital video display on these devices, such as LCD or plasma, does not require the scanning mechanism described previously. However, when the technology for digital video started to evolve, the television instruments were still rendering analog signals only. As a result, the digital video standards have their representations and formats closely tied to analog TV standards—NTSC (developed by the *National Television Systems Committee*), PAL (*Phase Alternating Line*), and SECAM (*Système Electronique Couleur Avec Mémoire*). The first commercial television program was broadcast in monochrome in

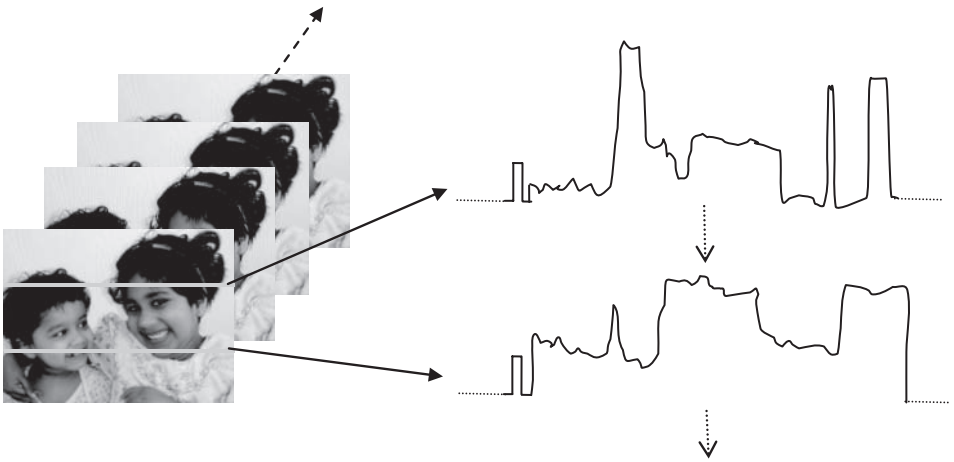


Figure 3-8 Left: Video is represented as a sequence of images. Right: Analog video of one frame scanned as a 1D signal. Each scan line is scanned from left to right as an analog signal separated by horizontal syncs. Two scan lines are shown; each begins with a horizontal sync and traces through the intensity variation on that scan line.

the United States in 1941, with standards developed by NTSC. NTSC's standards for color television were published in 1954, and they have evolved since then to cover VCRs, and have also influenced digital cable, HDTV, and digital video. It is natural that we take a closer look at the evolution of analog TV.

2.2 Analog Video and Television

Although digital video is thought of as a three-dimensional signal in space and time, the analog video signal used in broadcast is scanned as a one-dimensional signal in time, where the spatiotemporal information is ordered as a function of time according to a predefined scanning convention. This 1D signal captures the time-varying image intensity information only along scanned lines. Television requires this analog scanned information to be broadcast from a broadcast station to all users, as illustrated in Figure 3-9.

The standardization process implemented in the broadcast of analog video for television mandated a few requirements, which were necessary for making television transmission viable: YUV color space conversion and interlaced scanning. These requirements, although not necessary for digital video representation, still need to be supported in the digital world because of the well-entrenched standards for analog television displays. Analog displays will gradually transition into digital display devices, but for now, both need to be supported.

2.2.1 Conversion to YUV

Video frames, like images, are represented using a color format, which is normally RGB. This RGB color space is used by cathode-ray tube–based display devices, such

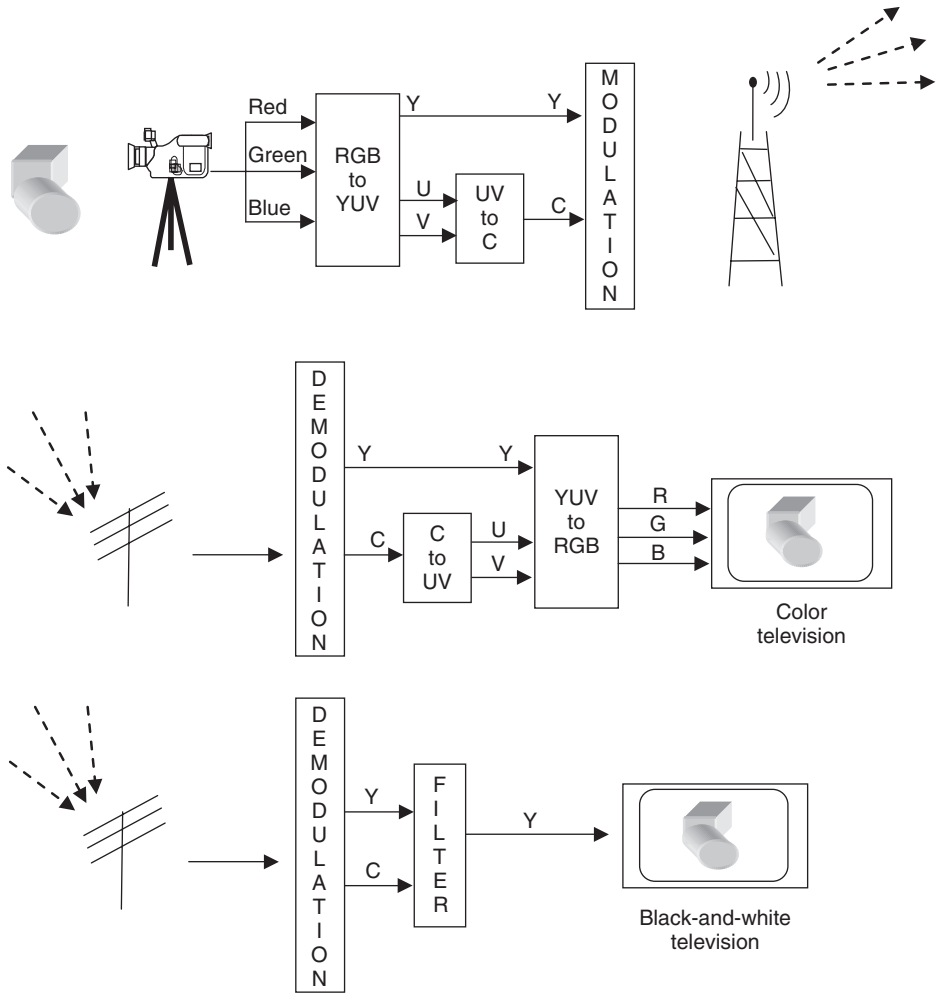


Figure 3-9 Television works by sending scan line information in interlaced YUV format.

as the television, to display and render the video signal. For transmission purposes, however, the RGB signal is transformed into a YUV signal. The YUV color space aims to decouple the intensity information (Y or luminance) from the color information (UV or chrominance). The separation was intended to reduce the transmission bandwidth and is based on experiments with the human visual system, which suggests that humans are more tolerant to color distortions than to intensity distortions. In other words, reducing the color resolution does not affect our perception.

2.2.2 Analog Video Scanning

Video is scanned as a 1D signal, as explained in the preceding section, where each raster line is interspaced with horizontal and vertical syncs. For synchronization of

transmission purposes, the line-by-line analog raster signal has to be rendered on your television in a corresponding manner, as the data is received. This synchronization is carried out by the cycles in the power outlet (60 Hz for NTSC, 50 Hz for PAL). Every 1/60th of a second, the electron gun is reset by the vertical sync to draw the beginning of the next frame. However, to meet the synchronization needs, each frame is broken down into two fields—an odd field and an even field. The odd field consists of the odd-numbered scan lines and the even field consists of the even-numbered scan lines, as shown in 0.

The electron gun at the back of the TV tube first draws the odd lines of the on-screen image, and then during a second pass, it draws the even-numbered lines. For NTSC signals, this all occurs within 1/30 of a second and each field is drawn at 1/60th of a second. But the resulting video drawn by interlaced scanning

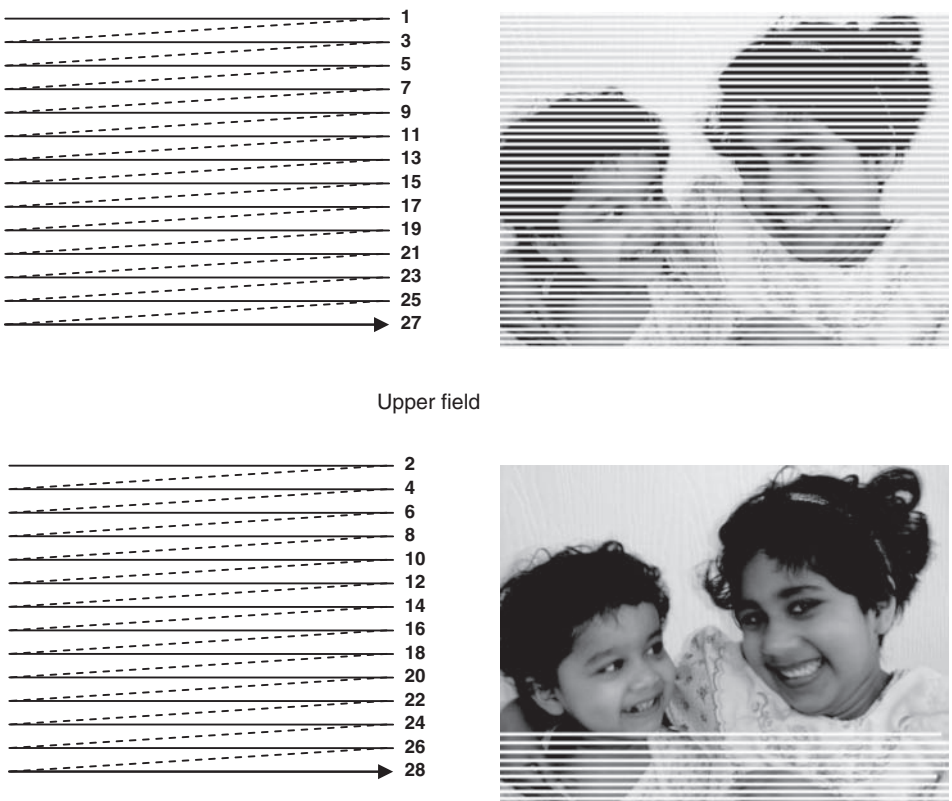


Figure 3-10 Interlaced scanning. The top figure shows the upper "odd" field consisting of odd-numbered lines. The bottom shows a lower "even" field interspersed with the odd field. Both fields are shown in succession to meet the required frame rate.

techniques might be unacceptable and has occasional flicker and artifacts. This is caused because the video is captured at different moments in time as two fields and, hence, interlaced video frames exhibit motion artifacts when both fields are combined and displayed at the same moment. Video is of better quality when it is captured progressively and drawn progressively, which eliminates the occasional flicker. In Figure 3-11, all lines, whether even or odd, are drawn in succession, resulting in 30 frames per second.

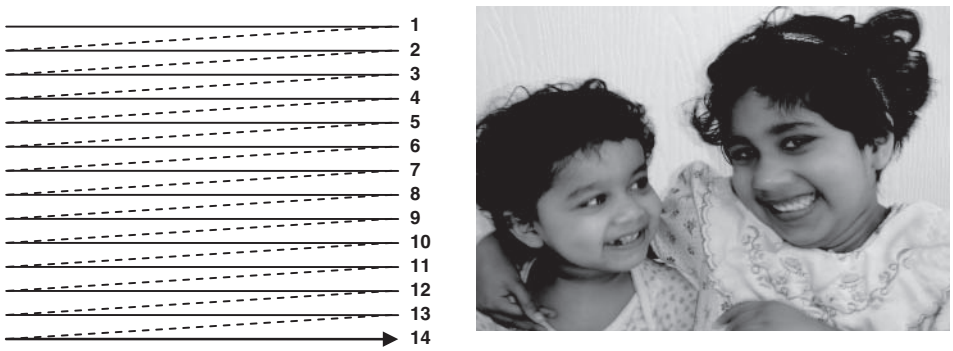


Figure 3-11 Progressive scanning. All the scan lines are drawn in succession, unlike in the interlaced case.

Interlaced scanning techniques were, thus, introduced for the television broadcast system to work in a synchronized manner. Today, digital television and HDTV, which use digital formats and digital synchronization features, are making commercial inroads into home entertainment. However, most television sets are still analog and the transition will take time. In the meantime, the digital domain displays need to understand TV standards and conversely, the digital formats need to be rendered on analog TVs.

2.3 Types of Video Signals

Video signals have been traditionally transmitted as analog signals for television broadcast. This was achieved by combining all the color and luminance information into one signal called *composite video*. Much of this format used for analog broadcast had to do with bandwidth availability. However, with higher digital bandwidths on digital networks, you could transmit them separately—such as *S-Video* and *component* video—to get better visual quality. The latter types of video signals are commonly seen as video outputs in DVD players and digital video recording instruments and are explained in the following sections. Most of the differences have to do with the way the originally captured R, G, B components of the video signal are transmitted to the end receivers such as televisions. In all cases, this is converted to a YUV (or equivalent) signal and the individual components are sent collectively or separately.

2.3.1 Composite Video

Composite video is also called baseband video or RCA video. It is the analog waveform that conveys the image data in the conventional NTSC television signal. Composite video contains both chrominance (color) and luminance (brightness) information, along with synchronization and blanking pulses, all together in a single signal. As mentioned earlier, this was done to reduce bandwidth and achieve real-time transmission. However, in composite video, interference between the chrominance and luminance information is inevitable and tends to worsen when the signal is weak. This is why fluctuating colors, false colors, and intensity variations are seen when a distant NTSC television station sends signals that are weak and not properly captured at home with old-fashioned "rabbit ears," or outdoor "aerial" antennae. Some DVD players and videocassette recorders (VCRs) accommodate composite video inputs/outputs for the purpose of connecting to standard NTSC televisions, which only accept composite video.

2.3.2 S-Video

S-Video (*Super-Video*, sometimes referred to as *Y/C Video*) is a video signal transmission in which the luminance signal and the chrominance signal are transmitted separately to achieve superior picture clarity. The luminance signal (*Y*) carries brightness information, and the chrominance signal (*C*) carries color information. Here, the chrominance signal (*C*) is formed by combining the two chrominance signals *U* and *V* into one signal along with their respective synchronization data, so at display time, the *C* signal can be separated into *U* and *V* signals. This is unlike the traditional composite video where all three channels are combined together into one signal. Separating the *Y* and *C* channels and sending them separately reduces problems caused by interference between the luminance and chrominance signals and yields a superior visual quality. Although the bandwidth required for analog broadcast of S-Video is not available yet, these signals are typically produced by some DVD players and computer monitors.

2.3.3 Component Video

Component video strives to go a step further than S-Video by keeping all three *Y, U, V* (or equivalent) components separate. Consequently, the bandwidth required to broadcast component video is more than the composite or S-Video and, correspondingly, so is the visual quality. The separation of these components prevents artifacts due to intersignal interference.

2.4 YUV Subsampling Schemes

Video signals captured by digital cameras are represented in the RGB color space, which is also used to render video frames on a display device. However, for transmission and other intermediary processing, the YUV space is commonly used as explained in Section 2.2.1. The YUV space separates the color and luminance information. The color information (UV) is then further subsampled to gain more bandwidth. Experiments with the human visual system have shown that this reduction in bandwidth still maintains an acceptable quality of video for broadcast because

the human eye is not as sensitive to subtle differences in color as it is to differences in brightness. In analog video, subsampling is achieved by allocating half as much bandwidth to chrominance as to luminance. In digital video, subsampling can be done by reducing the number of bits used for the color channels on average.

Depending on the way subsampling is done, a variety of subsampling ratios can be achieved. This is depicted in Figure 3-12, which can be interpreted as follows: The circles represent pixel information. Potentially, we could store 1 byte each for Y, U, and V components, resulting in 24 bits per pixel. In subsampling, the luminance component Y is left untouched—that is, 1 byte is reserved for the luminance data per pixel. An X at a pixel position suggests that we also store the chrominance components for this position.

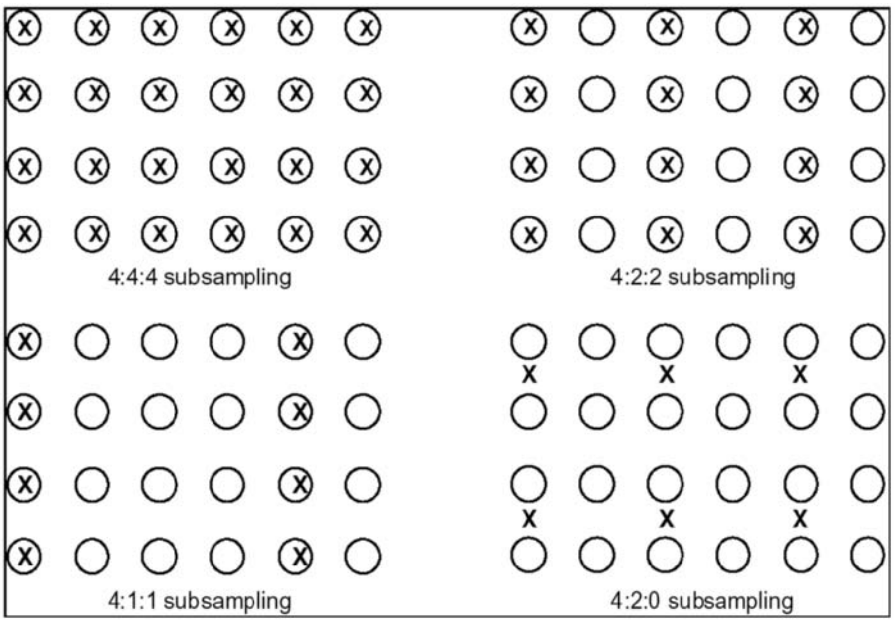


Figure 3-12 YUV subsampling schemes used in video. The upper-left image shows a 4:4:4 scheme without subsampling. The upper-right image shows the 4:2:2 scheme where luminance information is kept everywhere but chrominance information is kept for every other pixel. The lower-left image shows the 4:1:1 scheme where chrominance information is kept for every fourth pixel in a row. The lower-right image shows the 4:2:0 scheme.

Thus, in the 4:4:4 scheme, each pixel has luminance (8 bits) and chrominance (8 bits for U and 8 bits for V), resulting in 24 bits per pixel. In the 4:2:2 subsampling scheme, chrominance information is stored for every other pixel bringing the equivalent bits per pixel down to 16. In the 4:1:1 subsampling scheme, chrominance is stored every fourth pixel in a row, whereas in the 4:2:0 scheme, the average of the U values for a 2 × 2 pixel area is stored, and similarly for the V values. Since there is only 1 U and 1 V sample for every four luminance samples, the equivalent bits per pixel is brought down to 12 bits per pixel.

2.5 Digital Video Formats

The analog TV formats such as NTSC, PAL, and SECAM have been around for a long time and have also made their way into VHS technology. Prior to discussing digital video formats, Figure 3-13 describes the analog counterparts.

Property	NTSC	PAL	SECAM
Frame rate	30	25	25
Number of scan lines	525	625	625
Number of active lines	480	576	576
Aspect ratio	4:3	4:3	4:3
Color model	YIQ	YUV	YDbDr
Primary area of usage	North America (USA and Canada), Japan	Asia	France and Russia

Figure 3-13 Table illustrating analog video formats and their details

The digital video formats have been established for digital video applications. The CCIR (Consultative Committee for International Radio) body has established the ITU-R_601 standard that has been adopted by the popular DV video applications. For example, the CIF format (Common Interchange Format) was established for a progressive digital broadcast television. It consists of VHS quality resolutions whose width and height are divisible by 8—a requirement for digital encoding algorithms. The Quarter Common Interchange Format (QCIF) was established for digital videoconferencing over ISDN lines. Some of the commonly used digital video formats are illustrated in Figure 3-14.

2.5.1 High-Definition Television

High-definition television (HDTV) has been getting media attention for several years now. This section attempts to explain what HDTV is all about and how it differs from the other television standards. The usual NTSC analog TV signal in the United States has 525 scan lines, with 480 actually visible. The usual TV has an effective picture resolution of about 210,000 pixels. This level of resolution was amazing 50 years ago, but today, consumers are accustomed to better resolutions such as 1024 × 768 and even higher, which are now commonly supported by most graphics hardware that come with computers. The standard definition TV technology seems pale when compared with these resolutions. Digital television (DTV) uses the CCIR standards shown in Figure 3-14 for broadcast transmission over the air or via a cable/satellite system. Most of the requirements for adhering to the standard resolutions are due to support for bandwidth. A class of digital television called HDTV supports a higher resolution display format along with surround sound. The visual formats used in HDTV are as follows:

- 720*p*—1280 × 720 pixels progressive
- 1080*i*—1920 × 1080 pixels interlaced
- 1080*p*—1920 × 1080 pixels progressive

Format name	Lines per frame	Pixels per line	Frames per second	Support for interlaced format	Subsampling scheme	Image aspect ratio
CIF	288	352		N	4:2:0	4:3
QCIF	144	176		N	4:2:0	4:3
SQCIF	96	128		N	4:2:0	4:3
4CIF	576	704		N	4:2:0	4:3
SIF-525	240	352	30	N	4:2:0	4:3
SIF-625	288	352	25	N	4:2:0	4:3
CCIR 601 NTSC (DV, DVB, DTV)	480	720	29.97	Y	4:2:2	4:3
CCIR 601 PAL/SECAM	576	720	25	Y	4:2:0	4:3
EDTV (576p)	480/576	720	29.97	N	4:2:0	4:3/16:9
HDTV (720p)	720	1280	59.94	N	4:2:0	16:9
HDTV (1080i)	1080	1920	29.97	Y	4:2:0	16:9
HDTV (1080p)	1080	1920	29.97	N	4:2:0	16:9
Digital cinema (2K)	1080	2048	24	N	4:4:4	47:20
Digital cinema (4K)	2160	4096	24	N	4:4:4	47:20

Figure 3-14 Table illustrating digital video formats and their details

Standards are in place for HDTV. They use the MPEG2-based video compression format with a 17 Mbps bandwidth. The MPEG2 technology is well known now and has made headway into the DVD format as well. Although HDTV signals can be stored and transmitted effectively using MPEG-2 technology, a lot of bandwidth is required to transmit numerous channels. Production studios are currently upgrading hardware/software equipment to meet this consumer demand.

The aspect ratio of HDTV is 16:9 (1.78:1), which is closer to the ratios used in theatrical movies, typically 1.85:1 or 2.35:1. Currently, broadcasters must either pan/scan the image (crop the full picture of the film down to 4:3, eliminating part of every scene in the process) or letterbox it (present the full picture only on the middle part of the screen, with black bars above and below it).

In the twenty-first century, the extent and scope of motion pictures and television will be defined by creative ways in which we choose to view and interact with

them. New levels of stereoscopic realism will be offered by virtual reality systems; new interactions of sight and sound will be discovered; and we will develop new kinds of critical awareness as we begin to take personal control of the balance between film and still, "fast-forward" and "stop-motion," and "close-up" and "long shot."

3 DIGITAL AUDIO

Physics describes sound as a form of energy that is transmitted through a medium as pressure waves making rarefactions and compressions. These pressure changes can be captured by an electromechanical device, such as a microphone, and converted into an analog signal. Typically, a recording device consists of a magnet surrounded by a coil, which is attached to a diaphragm. As sound waves vibrate the diaphragm, the coil moves about the magnet, which induces an electric current proportional to the amplitude of the vibration. The electric current is captured as an analog signal, which can be further stored on an analog recording device such as a magnetic tape. The first sound capturing/reproducing equipment appeared in the late 1800s in the form of a phonograph created by Thomas Alva Edison and a gramophone by Emile Berliner. These sounds were initially monophonic having only one channel. Stereophonic sounds aim to add dimensionality to the sound experience by using more than one channel. Engineers make a technical distinction between *binaural* and *stereophonic* recording. Whereas binaural is reserved for the use of two channels, stereophonic refers to the use of more than two channels. In binaural recording, a pair of microphones records two different sounds or channels, which are then played back on two different speakers without mixing or cross talk. This creates the perception of spatial sound. Binaural sound systems were introduced in the 1920s. BBC made radio's first stereo broadcast in 1925 and soon made its way to other distribution modes, such as movies and television. Multichannel stereo sound was first used in Walt Disney's *Fantasia* in 1940, where three channels created a surround sound experience. Since then, the concept of surround sound has been extended to the use of five (and upwards) channels that have now become standard in home entertainment systems, movie theaters, game platforms, and so on. Another related concept in stereophony is *spatial* audio, which attempts to create surround sound like effects using a few channels. The following sections first discuss the digitization of analog sound, explain its digital representation, and further the explanation for surround sound as well as spatial audio.

3.1 Digital Representation of Audio

Analog audio signals are typically represented as waveforms, either simple or complex. A simple sinusoidal wave corresponds to a pure tone at a single frequency, or pitch. The amplitude of the wave gives the strength of the sinusoid at that time. A complex wave, on the other hand, consists of multiple frequencies or sinusoidal waves combined together. Most audio signals that are of interest to us, such as voice, music,

and so on, are composed of multiple frequencies, where the amplitude of the complex signal is the joint combination of the amplitudes of the individual frequencies. This understanding, which is normally referred to as the frequency domain representation of the analog signal, has been explained in Chapter 2.

Digitizing an analog audio signal requires sampling and quantization. The process of conversion to digital sound is known as pulse code modulation (PCM). The analog sound is sensed at evenly spaced time intervals, producing digital audio samples. The number of samples per time unit (sampling rate) must be specified during the digitization process. Given a sample, the signal amplitude at that position is encoded on a fixed number of bits. All samples are represented by the same number of quantization bits. The sampling rate and the quantization bits per sample are the main properties of the PCM signal and need to be carefully chosen so that it is possible to reconstruct the analog equivalent. The digital audio signal is finally rendered by converting it to the analog domain and so the choice of sampling rate and sample size need to be chosen appropriately in order to faithfully re-create the original sound. In addition to sampling rate and quantization, another characteristic commonly used to describe audio signals is the number of channels, which may be one (mono), two (stereo), or multichannel (surround sound). Also of growing interest in research and industry is the notion of spatial audio. Mono and stereo sound technology are the most commonly used, the following sections discuss surround sound and spatial audio in more detail.

3.2 Surround Sound

Surround sound aims to create a three-dimensional sound experience. It refers to the use of multiple audio tracks to engulf the listening audience in many sources of sound, making them feel as if they are in the middle of the action or concert. Surround sound usage is standard in movie theaters, where the surround sound movie soundtrack and the loudspeaker equipment allow the audience to hear sounds coming from all around them. It plays a large part in producing what movie makers call *suspended disbelief*. Suspended disbelief occurs when the audience is completely captivated by the movie experience and is no longer aware of their real-world surroundings.

True surround sound formats rely on multiple dedicated speakers that literally and physically surround the audience. A custom rule in the industry forces the use of a specific number of channels. The commonly accepted version is 5.1 surround sound, which consists of six speakers. One center speaker carries most of the dialog (because the actors usually speak while making their on-screen appearance) and part of the soundtrack. The left and right front speakers carry most of the soundtrack (music and sound effects), and might carry parts of the dialog when the director wants to intentionally offset the source of the dialog to either side, from its default dead-center screen location. A pair of surround sound speakers are placed to the side of (and slightly above) the audience to provide the surround sound and ambient effects. Finally, a subwoofer can be used to reproduce the low and very low frequency effects (LFE). Psychoacoustic studies have shown that humans use higher frequencies rather than lower frequencies for directional perception. The subwoofer plays low frequency sounds only and, hence, can be placed anywhere convenient (though people do exhibit a preference), while the other higher-frequency surround sound speakers have well-defined locations.

The 5.1 surround sound concept has also been extended to 7.1 to include two more spatial speakers. Figure 3-15 shows the configurations of these two systems.

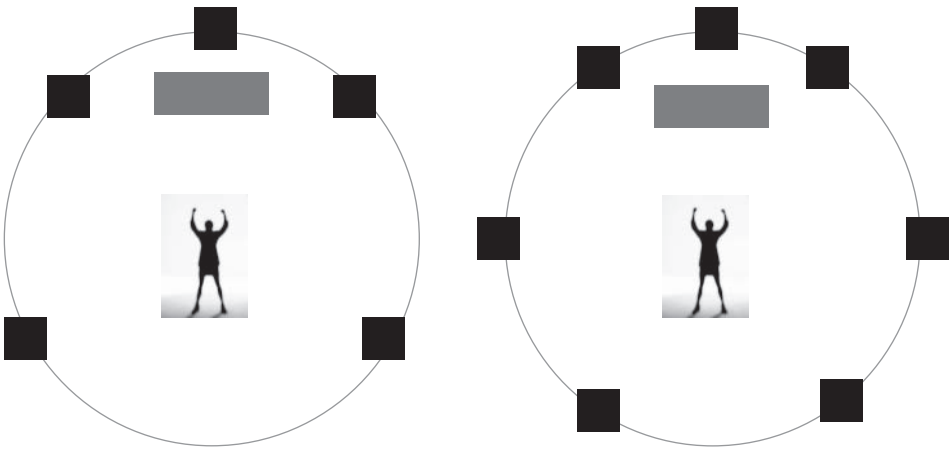


Figure 3-15 5.1 surround sound system (left) and 7.1 surround sound system (right). The black squares show the surround sound speaker placement in reference to a listener in the center, while the light gray rectangle shows the placement of the low-frequency subwoofer.

Movie theaters and other entertainment locations today use the THX system or the Dolby Digital Sound system. The THX system was invented by Tomlinson Holman and is a 10.2 surround sound system. It consists of five front speakers (left wide, left, center, right, eight wide), five surround speakers, two subwoofers, and two height speakers. The .2 of the 10.2 refers to the addition of a second subwoofer. The system is bass managed in a manner such that all the speakers on the left side use the left subwoofer and all the speakers on the right side use the right subwoofer. The center and back surround speaker are split among the two subwoofers.

3.3 Spatial Audio

Spatial audio, though similar in intent to create a surround sound like experience, attempts to create directional effects using fewer channels, typically two stereo channels. As such, this can be classified as *virtual* surround sound processes (for example, Sound Retrieval System [SRS] and other proprietary algorithms) that make use of only two left and right speakers and psychoacoustics effects to emulate true surround sound. Psychoacoustics is addressed Chapter 8 on audio compression where we describe how it is used to devise compression algorithms and standards for digital audio.

The spatial audio process widens the stereo effect by modifying phase information among the channels. This typically makes use of the *Head Related Transfer Functions* (*HRTF*) and *reverberation* to simulate the virtual localization of the sound source as coming from the left, right, front, back, or behind the listener. Given a sound source described by a frequency and located at a specific location with respect to the receiver, the HRTF describes how the sound is filtered by the diffraction/reflection and other

physical phenomena that happen because of the head position/orientation, pinna, and torso before the sound reaches the inner ear for perception. In other words, our brain perceives a sound as coming from a particular direction and elevation in large part due to the prefiltering effects of these external structures.

Reverberation is the persistence of sound in a particular space after the original sound is removed. When a source gives out sound, the sound goes directly to the listener and also reflects off surfaces to create an echo. The echo is ultimately absorbed by surfaces and decays, thus creating a reverberation. This reverberation plays a role in our perception of where the sound source is and can be simulated to create a sense of a virtual source.

3.4 Commonly Used Audio Formats

Digital audio formats emerged with the use and distribution of CD audio discs. These were uncompressed pulse code modulated digital signals in mono and in stereo. However, a number of formats have now become mainstream with the need for streaming, mobile, and surround sound technologies. The table in Figure 3-16 briefly describes the salient features of commonly used audio formats. Each format has its particular strengths and specific uses.

File suffix or logo	Filename	File type	Features
.wav	WAV	Uncompressed PCM coded	Default standard for audio on PCs. WAV files are coded in PCM format.
.au	G.711 μ-law, or ITU μ-law	Uncompressed audio	Universal support for telephone. Packs each 16-bit sample into 8 bits, by using logarithmic table to encode with a 13-bit dynamic range. Encoding and decoding is very fast.
GSM 06.10	Global System for Mobile Communication	Lossy Compressed mobile audio	International standard for cellular telephone technology. Uses linear predictive coding to substantially compress the data. Compression/decompression is slow. Freely available and, thus, widely used
.mp3	MPEG1 Layer3	Compressed audio file format	Uses psychoacoustics for compression Very good bandwidth savings and, hence, used for streaming and Internet downloads.

(Continued)

Figure 3-16 Table illustrating various commonly used audio file formats and the salient features each format supports

.ra	Real Audio	Compressed format	Proprietary to Real Audio. Capable of streaming and downloading. Comparable quality to mp3 at high data rates but not so at low data rates
AAC	Advanced Audio Codec MPEG4	Compressed format	Superior quality to .mp3.
.mid	MIDI—Musical Instrument Digital Interface	Descriptive format	MIDI is a language of communication among musical instruments. Description achieved by frequencies, decays, transients, and event lists. Sound has to be synthesized by the instrument.
	Dolby Digital (formerly called	Compressed 5.1 surround sound	De facto standard of home entertainment (Dolby AC-3) Distributed with DVD, HDTV systems. Provides five discrete channels—center, left, right, surround left, and surround right—plus an additional six for LFE.
	DTS Surround Sound	Compressed 5.1 surround sound	Alternate to Dolby Digital. Distributed with DVDs, but not HDTV. Has higher data rate compared with Dolby Digital.
	THX Surround Sound	Compressed 5.1 surround sound	Designed for movie theaters (THX Ultra) as well home theaters (THX Select). Has become the select brand for surround sound today.
	THX Surround Sound Extended	Compressed 6.1 or 7.1 surround sound	Jointly developed by Lucasfilm, THX and Dolby Laboratories. Also known as Dolby Digital ES. Has a surround back channel, placed behind audience achieving 360^0 of sound.

4 GRAPHICS

Computer graphics has evolved into an advanced field having applications in high-end markets such as visual effects movies, interactive and multiplayer games, scientific visualization, and computer aided design, to name a few. Most of today's

graphics-related applications tend to be three dimensional, interactive, and available on a variety of platforms from computers, to game consoles, to handheld devices. From a multimedia point of view, 2D computer graphics has been a predominant force since the early 1990s to create interactive games. These were distributed in the form of interactive multimedia CD-ROMs for a variety of entertainment and educational purposes.

Graphics objects can be represented as vectors or rasters. Vector graphics are geometric entities saved in a vector format having attributes such as color. For example, a shape can be represented as a sequence of points to be connected in a specific order and filled with a color. The advantage of vector representations is that they provide infinite resolution; however, vector graphics need to be converted to raster images to be displayed. Raster images are represented as a grid of pixels, each pixel having *x,y* coordinates and a value that corresponds to a color. Sample operations performed on raster images include painting, compositing, and filtering effects. Figure 3-17 shows an example of rasters and vectors.

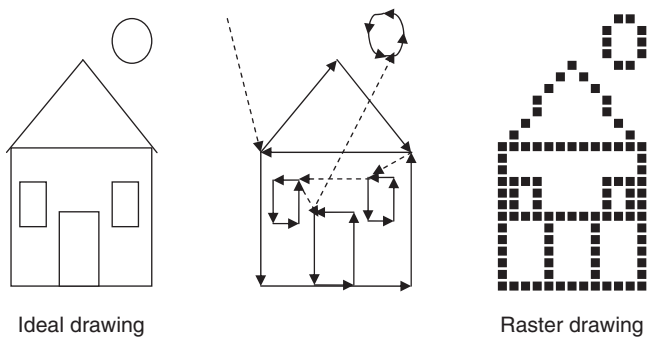
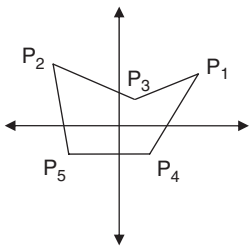


Figure 3-17 The left image shows a drawing that needs to be represented graphically. The middle image shows the corresponding representation using vectors. This is shown as a sequence of points joined by lines. The dotted lines show cursor movements. The right image shows a raster representation of the drawing.

4.1.1 2D Vector Graphics Representations

In vector graphics, objects are represented as smooth or discrete primitives. Smooth 2D primitives include parametrically defined entities, such as circles, ellipses, and other control point–defined curves such as splines. Discrete primitives, such as triangles and polygons have now become a choice for representing vector graphics because all hardware graphics cards have been designed to render such primitives. So for the purpose of this section, the discussion is limited to discrete primitives such as polygonal objects, which consist of *points* connected by straight lines in a specific order. Each connection between two points is defined as an *edge*. The edges connected together form a *polygon*. In 2D, points are represented by their *x*- and *y*-coordinates. A polygon can be defined by a number of points connected in a particular sequence. Figure 3-18 shows five points P_1, P_2, P_3, P_4, P_5 connected together in that order to form a polygon.

$$P_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad P_2 = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \quad P_3 = \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} \quad P_4 = \begin{bmatrix} x_4 \\ y_4 \end{bmatrix} \quad P_5 = \begin{bmatrix} x_5 \\ y_5 \end{bmatrix}$$



$$Polygon\ O = \begin{bmatrix} x_1 & x_4 & x_5 & x_2 & x_3 \\ y_1 & y_4 & y_5 & y_2 & y_3 \end{bmatrix}$$

Figure 3-18 An example of a polygon and its representation. Five 2D points P_1, P_2, P_3, P_4 and P_5 are connected to form polygon o.

4.1.2 Animations Using 2D Graphics

Animations are created by showing different images over time. In 2D graphics, these different images are raster representations of a vector object(s) that at each image frame has undergone a slight change. The changes can be as a result of the object moving to a different location (translation), the object rotating, being scaled up or down, or a variety of combinations of any of the above. For instance, if an object, for example a square, has to be shown moving in a sequence of frames, each of the intermediary frames must show the square translated by successive amounts until it reaches its destination position. In this section we discuss the mathematical representations of these operations. Assume that an object is represented by an ordered list of points as shown here:

$$O = \begin{bmatrix} x_1 & x_2 & x_3 & . & . & x_n \\ y_1 & y_2 & y_3 & . & . & y_n \end{bmatrix}$$

Each point i, represented as (x_i, y_i), can now have its coordinates transformed from frame to frame depending on operations such as translations, rotations, scaling, shearing, and so forth. Figure 3-19 illustrates a translation of an object. Translation involves moving

$$\begin{bmatrix} x_i + t_x \\ y_i + t_y \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

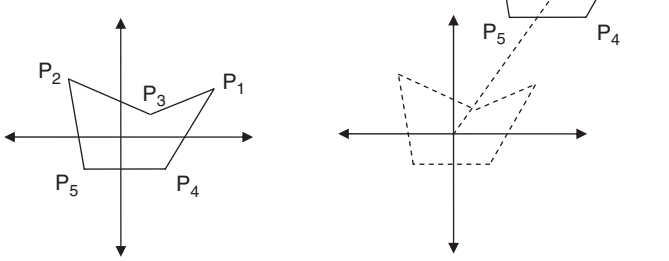


Figure 3-19 Translation – The polygon shown on the left is translated by a vector (t_x, t_y) as illustrated on the right

the object from one location to another. When this is performed incrementally over time and rendered in succession, you get the perception of the object moving.

Rotation is defined by an angle and an axis. The object rotates about an axis by an angle in either the clockwise or anticlockwise direction. Figure 3-20 illustrates an example of a rotation where the polygon rotates by an angle α about the origin.

$$\begin{bmatrix} x_i \,\mathrm{Cos}\alpha & - & y_i \,\mathrm{Sin}\alpha \\ x_i \,\mathrm{Sin}\alpha & + & y_i \,\mathrm{Cos}\alpha \end{bmatrix} = \begin{bmatrix} \mathrm{Cos}\alpha & - & \mathrm{Sin}\alpha \\ \mathrm{Sin}\alpha & & \mathrm{Cos}\alpha \end{bmatrix} \times \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

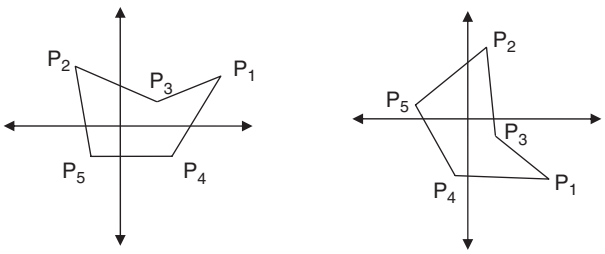


Figure 3-20 Rotation – The polygon shown on the left is rotated by an angle α about an axis that passes through the origin and is perpendicular to the plane of the paper

Scaling is defined by a scale amount and a point about which scaling occurs. The point about which scaling occurs is normally referred to as the scale pivot. The object is scaled by the scale amount about the scale pivot. Figure 3-21 illustrates an example of a scaling where the polygon is scaled by an amount s_x in the horizontal and s_y in the vertical directions. If s_x and s_y are the same, then the scaling is uniform.

$$\begin{bmatrix} s_x \, t_x \\ s_y \, t_y \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \times \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

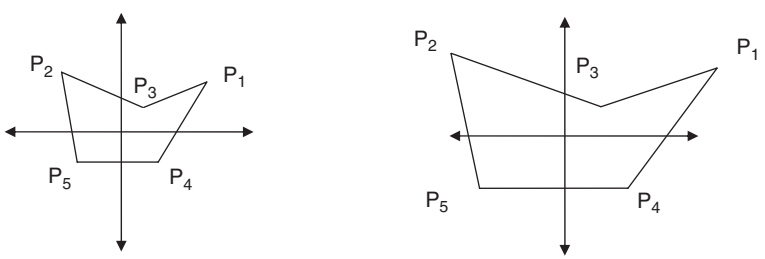


Figure 3-21 Scaling – The polygon shown on the left is scaled non-uniformly by an amount s_x in the horizontal direction and s_y in the vertical direction about the origin

Although these transforms are commonly used to compute the positions of points, lines, and objects in successive frames, one disadvantage in the computation is the need to treat translation differently from other transformations. As shown in the previous figures, translation results are achieved by *adding* a 2 × 1 matrix, whereas

rotations and scaling are achieved by *multiplying* a 2×2 matrix. This inconsistency is solved by representing points and the matrices using *homogeneous* coordinates. Using homogeneous coordinates, points are represented as a triplet of $(x_i, y_i, 1)$. In homogeneous terms, these matrices now become

$$\text{Translations:} \quad \begin{bmatrix} x_i + t_x \\ y_i + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\text{Rotations:} \quad \begin{bmatrix} x_i \cos\alpha - y_i \sin\alpha \\ x_i \sin\alpha + y_i \cos\alpha \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\text{Scaling:} \quad \begin{bmatrix} s_x x_i \\ s_y y_i \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}.$$

5 EXERCISES

1. [**R02**] Section 2.4 explains the various YUV subsampling schemes used. The initial RGB signal, in which every pixel is typically represented by 8 bits for each channel, is converted into a YUV signal with 8 bits for each Y, U, and V. It is then further subsampled. Using the subsampling descriptions in the text and assuming each channel is represented by 8 bits, complete the following:
   - Prove that the equivalent number of bits per pixel for the YUV 4:2:0 scheme is 12.
   - For the YUV 4:2:2 representation, prove that this is equal to 16.
   - What is the equivalent bit representation for the YUV 4:1:1 scheme?
2. [**R04**] Suppose a camera has 450 lines per frame, 520 pixels per line, and a 25 Hz frame rate. The color subsampling scheme is 4:2:0 and the pixel aspect ratio is 16:9. The camera uses interlaced scanning, and each sample of Y, Cr, Cb is quantized with 8 bits.
   - What is the bit rate produced by the camera?
   - Suppose you want to store the video signal on a hard disk and, to save space, you want to requantize each chrominance (Cr, Cb) signal with only 6 bits per sample. What is the minimum size of the hard disk required to store 10 minutes of video?
3. [**R04**] A digital camera captures 640×480 pixel color images. It has a 256 Mbytes flash memory.
   - How many images can be stored in the memory if no compression is used?
   - What is the required compression factor (per frame) to store 5 minutes of video? Assume that all frames compress by the same factor.

4. [**R04**] A video source produces 620 × 480 images, interlaced at a field rate of 59.94 Hz and full color (24 bpp). You first convert the video to 4:2:2 color format representation.
   - What compression rate do you need to achieve to send this video over a 1.5 Mbps line?
   - What if you started with 4:1:1 color (rather than 4:2:2 color)?
5. [**R04**] The interlaced scanning format was established during the development of broadcast television.
   - Which problem was solved by the introduction of interlaced scanning?

   Assume that the first field of each frame in the video is removed.
   - Is the resulting video interlaced or progressive? Why?
   - If an interlaced format is used to record fast-moving content (for example, a soccer game) and slow-moving content (for example, scenery), which case would have problems? Explain!
6. [**R05**] Consider a video signal taken by an interlaced camera at 40 fields per second, a color subsampling scheme of 4:2:0, 10 bits or each color channel per pixel, 480 pixels per line, and 360 lines per frame.
   - What is the bit rate of this video signal?
   - If the original image aspect ratio (ratio of width to height) was 16:9, what was the original pixel aspect ratio?
   - If the first field is removed as suggested in the previous question, how does the pixel aspect ratio change by removing the first field of every frame?
7. [**R06**] A CCD chip samples at 13.5 MHz (525/29.97 NTSC signal), producing 720 active pixels on a line (out of 858 line samples) and 480 active lines.
   - Are the pixels square? Explain.
   - If not, what sampling rate would make them square?
   - How many active pixels would be on a line?
8. [**R05**] The aspect ratio of a computer monitor is 4:3. You have a choice of changing the screen resolution by making the appropriate adjustments.
   - If the screen resolution is set to be 1024 × 768, what is the aspect ratio of pixels?
   - Compute the same if the resolution is 800 × 600.
   - For ITU BT.601, the aspect ratio is 4:3, and the resolution is 720 × 480. In this case, what is the aspect ratio of its pixels?
9. [**R06**] A digitized movie has a resolution of 608 × 256. The video is interlaced with a field rate of 47.952 Hz and uses full color (24 bpp). It is compressed by a compression codec with a compression rate of 95.
   - Can you watch it properly through a 10 Mbps LAN?
   - What compression rate do you need to achieve to watch it in real time through a LAN?
   - What if you convert it to 4:2:2 color first?

10. [**R05**] Commutative properties are often desired when operating on matrices. Two matrices A and B are said to be commutative if A*B = B*A. Here, you are asked to reason with the commutative properties of transformations:
    - Prove that, as a general rule, rotations and translations in 2D do not commute, that is, the results are different for R*T and T*R.
    - What about scaling and rotations? Is there any special case when S*R = R*S?

11. **[R04]** The rectangular object ABCD shown on the left has undergone a translation, rotation and scale to form the square on the right. Find a single 3 × 3 transformation matrix T which when applied to the coordinates of the object points on the left will result in the object on the right as shown in Figure 3-22.

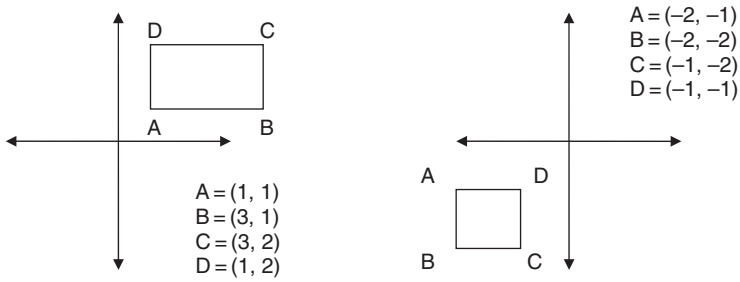


Figure 3-22

12. [**R06**] In Figure 3-23, a wheel shape is shown in its local coordinate system. You want to create an animation of 30 frames that will result in the wheel's center being placed at a point (1.1416, 1) in frame 1 and rolling to point (7.4247, 1) at frame 30. Remember, the wheel rolls over along the line (translates + rotates) without slipping. For a given frame *n*, you want to compute a transformation matrix T(n) that when applied to the wheel in its local coordinate system transforms it to the correct position in frame *n*. Compute T(n)—Remember this 3 × 3 matrix has to be a function of *n*!

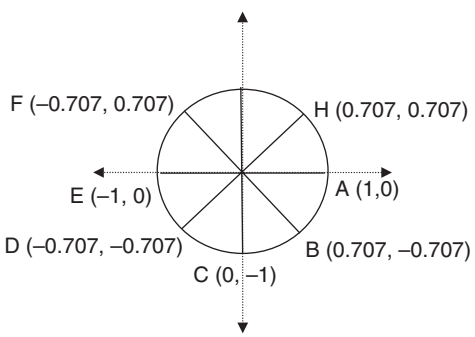


Figure 3-23

PROGRAMMING ASSIGNMENTS

13. **[R06]** This assignment will help you gain a practical understanding of issues that relate to image sampling, image aspect ratios, pixel aspect ratios, and format conversions. Standard definition formats (4:3 aspect ratios) have been prevalent for a long time; high-definition video streams (16:9 aspect ratios) are now being broadcast. Some practical observations and, hence, problems that need to be solved may be noted as follows:

    - High-definition 16:9 signals have started to be broadcast, but most household televisions still display the 4:3 standard definition format. Prior to display, the stream has to be reformatted so that it can be displayed on a 4:3 display. The reduction in image size and pixels can cause problems, such as aliasing and so forth.
    - In addition, high-definition television (16:9) sets might want to view standard definition signals because they are still the standard broadcast medium. The necessary reformatting/scaling causes changes in the pixel aspect ratios. There is also a growing industry interest to convert old, standard definition content from DVDs to higher definition for the Blueray format.

    Write a program that can convert an HD video stream to an SD stream. The conversion process reduces the number of samples (effectively subsampling), which might cause aliasing. Your conversion should incorporate simple antialiasing based on an averaging filter.

    In addition, write a program to convert an SD video stream to an HD video stream. The conversion process here increases the number of samples, which will cause artifacts in the scaled image.

    Answer the following analysis questions:

    a) When you convert SD to HD, there is a pixel scaling effect that changes your pixel aspect ratio. This is disturbing, and not desirable, but unavoidable in these cases. Can you think of any smart operations that you can do to minimize this stretching effect? Write code to create such an output and submit your result showing sample before-and-after output images.
    b) Repeat the same exercise as in a) but do it for the reverse case, that is, converting from HD to SD.
    c) Suggest ideas on how you would go about solving this: Converting an SD to HD image effectively involves increasing resolution when it is not there. This often results in smooth and unsharp features caused by filtering operations. What tricks can you think of to minimize this?

CHAPTER 4

Color Theory

This chapter explains the role and significance of color, the physical basis of color vision, and the psychophysics of color perception. We then show how these results are used to calibrate the capture devices and display devices commonly used in the multimedia pipeline. Various instruments capture visual signals such as images and video. These digital visual signals become part of multimedia content, which gets rendered on different display devices. Color is, thus, an intrinsic aspect of visual imagery that needs to be captured and displayed faithfully. Many figures in this chapter are printed (in full-color) in the color insert section of this textbook.

This chapter first starts by explaining the "color problem" and its significance to multimedia in Section 1. Next, in Section 2, we explain the physical structure of the eye and the physiological aspects that lead to the theory of color perception by the human visual system. The theory is necessary to understand the succeeding Sections 3 on digital color representation and the color calibration of capture/display devices. Lastly, we introduce the different color spaces used in various applications and their interconversions in Section 4 and show how different rendering devices work in Section 5.

1 THE COLOR PROBLEM

Visual capture devices, such as cameras, video cameras, or camcorders, capture visual signals and store them in a digital format. Images can also be scanned into a digital representation using scanners. These stored signals go through some or all parts of a multimedia system illustrated in the introductory chapter, Chapter 1—authoring, compression, networked delivery, decoding, and so on—and the content is finally viewed by the end user. At the end user, this content is rendered on visual display devices, such as televisions, computers, or projectors, or printed using a color printer. For now,

let us neglect the artifacts caused by lossy compression, quantization, and errors in network delivery. The expectation is that the rendered or reproduced color image on the end terminal should "*look the same*" as the color image of the original object, irrespective of which device captured the visual image, or which device is rendering it. Figure 4-1 illustrates this scenario. This appears to be an impossible task, as the signal reaching the capturing sensor is the result of a complex interaction between the material observed and the (unknown) illuminating conditions. Similarly, the signal reaching the eye is the result of the complex interaction between the observed image and the illuminating conditions. In practice, however, people have no problem declaring that that images rendered on different devices "look the same." Resolving this difficulty requires a clear understanding of the physics of color image formation, which involves wavelength analysis, understanding of the psychophysics of image perception by the human visual system, and knowledge of the way capture and display devices work.

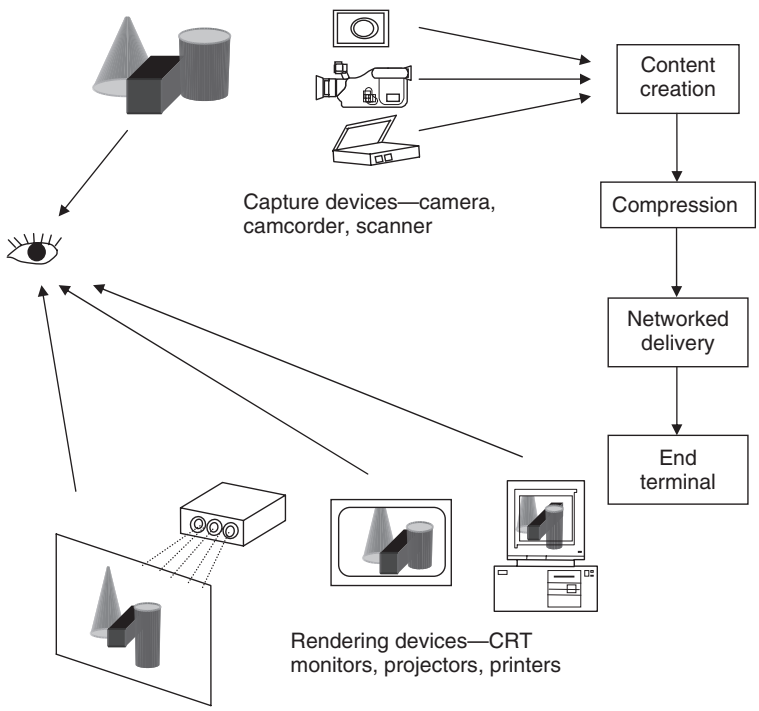


Figure 4-1 Illustration of the color problem. Capture devices capture images of objects having a variety of colors. These images, when rendered at the receiver's display terminal, such as projectors, printers, and CRTs, should "have the same color" as the original objects as perceived by the human visual system. See the color insert in this textbook for a full-color version of this image.

1.1 History of Color and Light

To understand the color problem, which is why and how an observer perceives the same colors when looking at the object or looking at a rendered image of it, we need to understand what color is. Defining color is not an easy task because color is both a physical element and a perceptual sensation interpreted by our brains when light enters the eye.

Historically, the quest for understanding color started with the empirical observations of Greek scholars such as Plato, Aristotle, and Pythagoras. They believed that color represented several important elements such as water, earth, fire, and wind. However, the scientific development of the theory of color started with the work of Isaac Newton. He showed that sunlight could be separated into the various colors of the rainbow when passed through a prism, and then recombined by passing through a second prism. His theory on colors eventually led to the creation of the color wheel in 1666 and later to an understanding of the frequencies that make up light.

1.1.1 The Nature of Light

The visible spectrum of white light contains all wavelengths from 400 to 700 nanometers. Wavelengths below 400 (known as ultraviolet) and those above 700 (known as infrared) cannot be sensed by the human visual system, and, hence, do not contribute to color perception. Furthermore, the distribution of energy among the visible wavelengths or frequencies also varies, depending on illumination. For example, daylight produces a spectral distribution that is different from the distribution due to yellow light from a tungsten filament, which is the material used in ordinary light bulbs. This spectral energy distribution is considered to be characteristic of the light source. Every light source has a unique spectral distribution. The visible spectrum and the spectral distribution of two example light sources are shown in Figure 4-2. A monochromatic

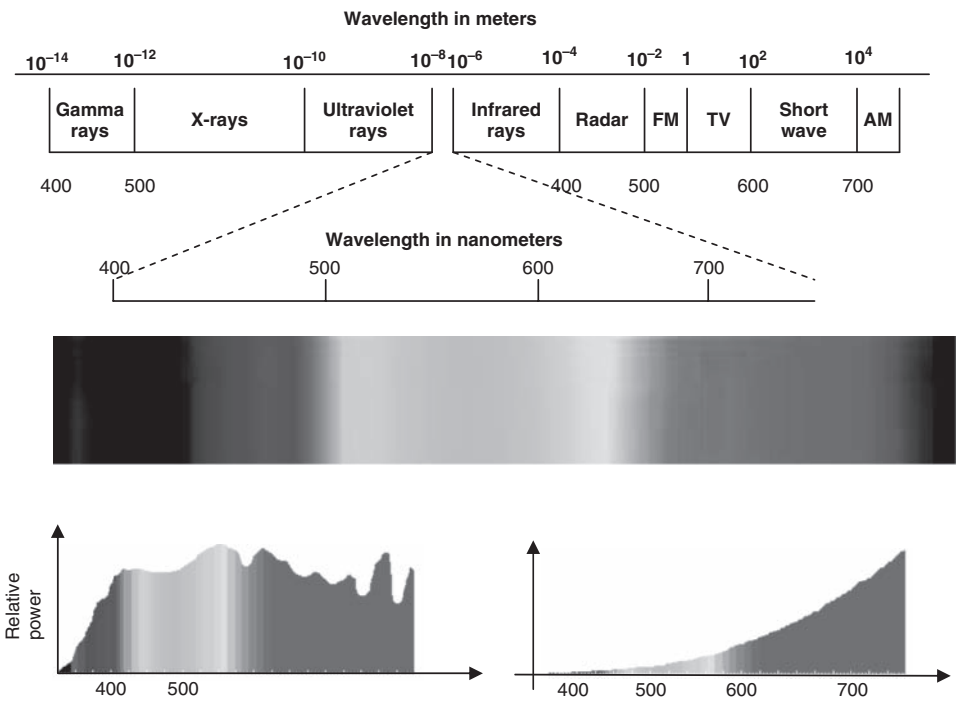


Figure 4-2 An understanding of the visible spectrum. The top figure shows the range of frequencies and where the visible spectrum lies in comparison. The bottom two figures show the distribution of relative spectral power density for the spectra from daylight and an incandescent light, respectively. See the color insert in this textbook for a full-color version of this image.

laser, such as a red laser commonly used as a pointer for presentations or for scanning a compact disc, spans a very narrow wavelength band.

Newton's work helped analyze the visible spectrum, but this is only part of the story. The generation of images involves the interaction of light with material. This interaction involves reflection and absorption. The ratio of reflected to absorbed energy depends on the material illuminated, and on the wavelength of the illuminant, which, in turn, produces an energy field captured by the sensor.

Then, each sensor element, whether in our eye, camera, or scanner, produces a value that measures the absorbed energy in the wavelength it is tuned to.

1.1.2 Theories of Color Vision

One hundred years after Newton, the perception of color was studied by Thomas Young and supplemented by Hermann von Helmholtz and is now widely known as the Young-Helmholtz theory. Young proposed the threefold character of color perception, and speculated as early as 1802 that the eye must have three different kinds of color receptors, corresponding roughly to the red, green, and blue primary colors. James Clerk Maxwell furthered Young's work by exploring the use of three primary colors to generate various colors by additively mixing. His experiments demonstrated that most colors can be matched by superimposing three separate lights known as primaries—a process known as additive mixing. He also showed that, although three primary colors were necessary, the choice of three was not unique and, more important, that no additive combination of three primaries can cover all the perceivable colors. Maxwell's work can be considered the basis for modern colorimetry.

Experiments carried out in the 1920s showed that the red, green, and blue primaries could indeed match all visual colors within a certain range called a gamut, but that they could not match all the spectral colors, particularly in the green range. It was found that if a certain amount of red light was added to the color being matched, all colors could be matched. These quantitative results were based on the postulate of Young, Helmholtz, and Maxwell and expressed in terms of three stimulus values for the red, green, and blue primaries. This was later defined in a standardized system by the Commission Internationale de l'Eclairage (CIE) and is explained later in Sections 3 and 4. However, it was not until 1965 that detailed physiological experiments on the retina of the eye confirmed the presence of three different kinds of color receptors, as explained in the next section.

1.2 Human Color Sensing

We visually sense our environment using our eyes. The eyes sense illumination using photoreceptors on the retina. As shown in Figure 4-3, the retina contains two types of photoreceptors, *rods* and *cones*, which vary in number as well as their sensations. There are more rods (about 120 million) than cones (7 million) and rods are more sensitive. The rods are also distributed more uniformly than cones. The cones are more concentrated near the fovea. Figure 4-3 shows the structure of the eye and how the rods and cones are distributed on the retina.

Not only is the distribution of rods and cones skewed with respect to position and number, but also they are functionally very different. Rods are more than a thousand times more responsive than cones, and respond primarily to changes in illumination.

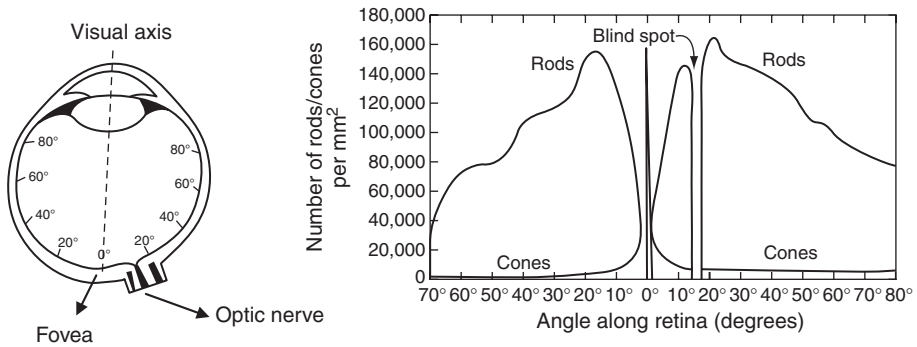


Figure 4-3 The left figure shows the structure of the eye. Also shown are the visual angles to the position on the retina. The right figure shows the distribution of rods and cones with respect to the visual angle. Note that the fovea has no rods but only cones.

The cones are responsible for the perception of color. Moreover, the cones are of three different types, which selectively sense the spectrum. The cones can be divided into "red" cones, "green" cones, and "blue" cones based on the different responses each generates when light is incident on them. The blue cones are distributed outside the fovea and, hence, have a larger spread and higher sensitivity. The green and red cones are concentrated mostly in the fovea and have a lesser sensitivity when compared with blue cones. These sensitivity or spectral response curves are shown in Figure 4-5. This selective sensing of the same spectrum is the physical basis for color, and, as we shall see shortly, color perception is more sophisticated than this.

1.3 Human Color Perception

The physics of image formation is now well understood, but fails to explain a number of phenomena associated with human perception. For instance, the brown and olive-green colors are clearly perceived by human observers, but cannot be produced by color mixing. Furthermore, when an observer looks at a brown color through a thin tube or straw, the color changes to orange! This indicates that color perception is not a strictly local process, but depends on a spatial neighborhood. This was hinted at by Ewald Hering, who proposed a theory in the early nineteenth century, which appeared, until recently, to contradict the Young-Helmoltz theory. He postulated the presence of three opponent processes, one measuring red-green, one measuring yellow-blue, and one measuring black-white. These processes required a neighborhood operation. Any color could then be described by three numbers, but on very different axes. Scientists were sharply divided as to which theory was correct, whereas in fact they both are: Young-Helmoltz describes the sensing; Hering describes further processing!

One of the most amazing aspect of human color perception is known as color constancy and is best demonstrated by Edwin Land (the founder of Polaroid) in the 1950s. He was interested in photography, and noticed that, unlike the eye, a camera does not compensate for the color of a light source. When we see a white sheet of paper under different illuminating conditions such as sunlight, a flash, or tungsten light, we perceive the paper to be white under all conditions. However, a picture taken with a camera of the

same paper looks different. To create pictures such that the paper has the same white color, the camera settings need to be adjusted according to the light source, whereas the human visual system does this automatically and effortlessly. Land illuminated a patchwork of color patches of paper with three slide projectors, each equipped with a different filter (red, green, and blue) in an otherwise dark room. Each projector could be controlled independently and could vary its intensity. Land could turn the three projectors sequentially on and measure the intensity at any point, giving him three numbers, defining the color reflected at each point. By manipulating the mix of intensity of the three projectors, he could make any patch, for example an orange one, give the same reading as a green patch under normal illumination. The naïve expectation is that the orange patch would now look green, but instead, subjects reported that the color did not change! An example of color constancy is shown illustrated in Figure 4-4, where a colored ball is shown imaged under different illumination conditions. The colors are perceived to be the similar. This clearly demonstrates that color perception is not a local process, and that humans use larger parts of the image to estimate or compensate for lighting.

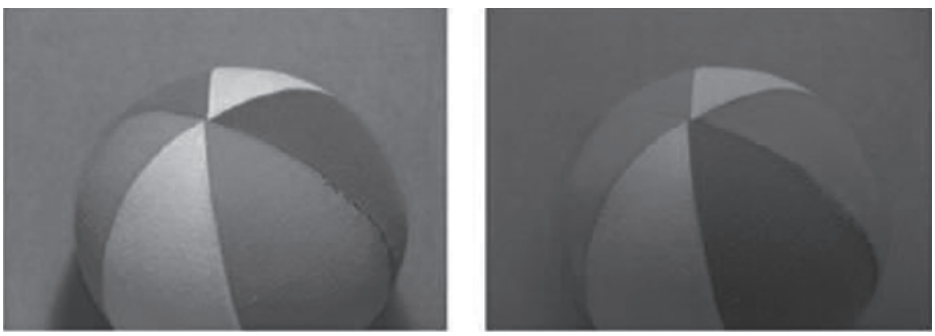


Image (a) Image (b)

Figure 4-4 Images of a scene taken under different illumination conditions. Image (a) is a scene under Tungsten illumination. Image (b) is the same scene under a blue illuminant having the same temperature as deep blue sky. See the color insert in this textbook for a full-color version of this image.

This amazing ability of the human visual system to adapt to and correct for lighting variations can be used to our advantage in the context of Figure 4-1. As long as the images produced by the capture and rendering devices are the same, the user will ignore effects due to illumination. In the next section, we show that the problem can be simplified by the trichromacity theory.

2 TRICHROMACITY THEORY

It is now well established both theoretically and physiologically that three types of receptors are necessary to generate the perception of the variety of colors in the visible spectrum. In humans, these three receptors are called red, green, and blue cones and they selectively sense the spectral distribution entering the eye. This selective sensitivity leads

to the perception of color. The condition of having three receptors is known as trichromacy, exhibited by humans and a few primates. Most mammals exhibit dichromacy (two receptors) only, and thus have a very limited set of colors that they can sense. Monochromacy (one receptor) is common among sea mammals, and does not result in any color perception, only intensity variations sensed by rods. Humans also tend to exhibit dichromacy when the three-cone system is nonfunctional, such as in cases of partial color blindness. In the following sections, we mathematically model the response of cones and show how it has been used to calibrate color.

2.1 Cone Response

The cones are of three different types, which selectively sense the spectrum and can be divided into red cones, green cones, and blue cones based on measured different response curves, as shown in Figure 4-5. Each cone is said to have a different spectral absorption.

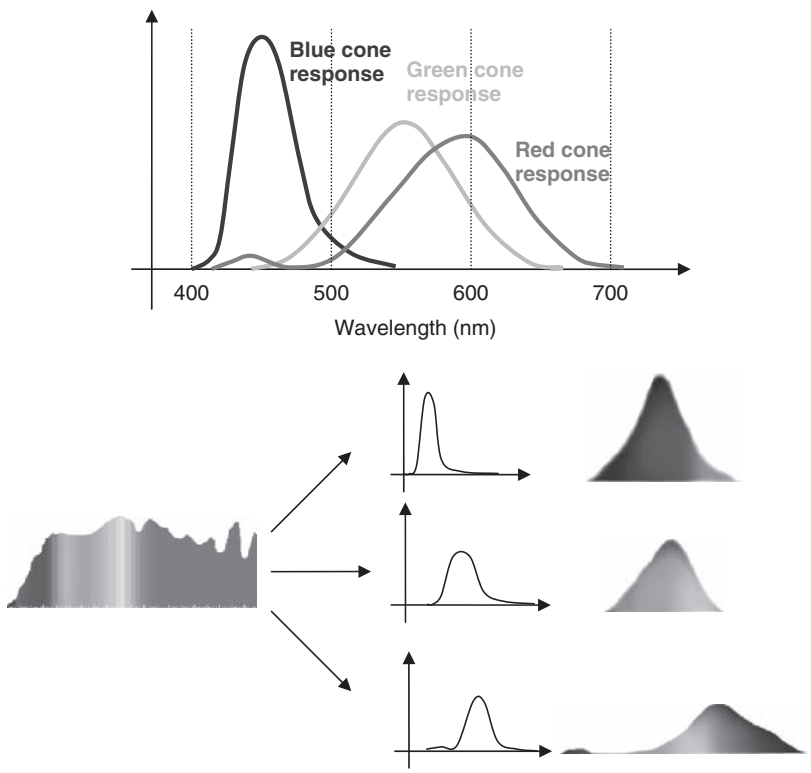
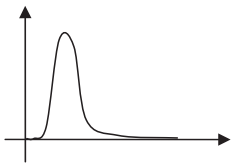


Figure 4-5 The top figure shows the cone response curves for blue, green, and red cones. The blue cones have a greater response compared with green and red cones. The bottom figure shows the selective sensitivity for the white light spectrum. See the color insert in this textbook for a full-color version of this image.

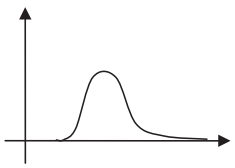
The selective sensing of a cone can be mathematically modeled by representing the response curves as parametric curve functions $s(\lambda)$. These functions can then be

convolved with the spectral representation $f(\lambda)$. The integral shown in Figure 4-6, evaluated within the limits of the visible spectral wavelength, gives the energy of the spectrum, which is sensed by the sensitivity curve of that cone. This model is used in later sections to arrive at an important result regarding the sensation of color.

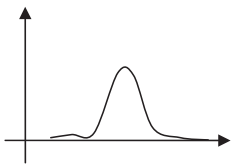
$$c = \int_{\lambda=400}^{\lambda=700} s(\lambda)\, f(\lambda)\, d\lambda$$



$$c_1 = \int_{\lambda=400}^{\lambda=700} s_1(\lambda)\, f(\lambda)\, d\lambda$$



$$c_2 = \int_{\lambda=400}^{\lambda=700} s_2(\lambda)\, f(\lambda)\, d\lambda$$



$$c_3 = \int_{\lambda=400}^{\lambda=700} s_3(\lambda)\, f(\lambda)\, d\lambda$$

Figure 4-6 The top equation shows a model of estimating the energy response of a cone with a response curve $s(\lambda)$ for a given spectrum $f(\lambda)$. The following equations show the energy response for the blue, green, and red cones, which are summed to c_1, c_2, and c_3, respectively. These three values are called the tristimulus vector.

The different responses are not the same for all people, but they do have a similar overall structure. An understanding of this design should help explain a few common observations, abnormalities, and pathological conditions regarding how people sense colors. For example, it is obvious that two individuals can correctly differentiate between tomatoes and oranges based on the color. However, the two might not agree on which tomato exhibits the reddest color. This is because the red cones can have minor differing responses in both individuals. Also, conditions like complete or partial color blindness can be attributed to incorrect responses of these cones. Next, we try to formalize a mathematical model to explain the cone responses.

2.2 The Tristimulus Vector

The integrals explained in Figure 4-6 of the previous section need continuous functions $s_1(\lambda)$ $s_2(\lambda)$ $s_3(\lambda)$ along with $f(\lambda)$. In practice, it is better to use sampled versions of the same.

A spectrum is represented as an array of numbers that are measured intensities at different wavelengths. Mathematically, this is analogous to a two-dimensional plot with the x-axis showing the wavelengths present in that spectrum, and the y-axis showing the energy each wavelength carries, measured in terms of the spectral power density or SPD. A spectrophotometric device usually has a resolution of 1–3 nm, thus measuring 100 to 300 values to quantify just the visible range.

The previous equation can then be rewritten as

$$c_i = \sum_{j=400}^{j=700} s_i(\lambda_j) f(\lambda_j), \text{ defined for the three cones } i = 1, 2, 3$$

Furthermore, $s_i(\lambda)$ can be defined to be a column matrix S, as shown in the following equation. In this case, $S^T f$ represents the tristimulus vector $[c_1, c_2, c_3]^T$.

$$f(\lambda) = \begin{bmatrix} f(\lambda_1) \\ f(\lambda_2) \\ .. \\ f(\lambda_n) \end{bmatrix} \qquad S = \begin{bmatrix} s_1(\lambda_1) & s_2(\lambda_1) & s_3(\lambda_1) \\ s_1(\lambda_2) & s_2(\lambda_2) & s_3(\lambda_2) \\ .. & .. & .. \\ s_1(\lambda_n) & s_2(\lambda_n) & s_3(\lambda_n) \end{bmatrix}$$

$$S^T f = \begin{bmatrix} s_1(\lambda_1) \times f(\lambda_1) + s_1(\lambda_2) \times f(\lambda_2) + \ldots + s_1(\lambda_n) \times f(\lambda_n) \\ s_2(\lambda_1) \times f(\lambda_1) + s_2(\lambda_2) \times f(\lambda_2) + \ldots + s_2(\lambda_n) \times f(\lambda_n) \\ s_3(\lambda_1) \times f(\lambda_1) + s_3(\lambda_2) \times f(\lambda_2) + \ldots + s_3(\lambda_n) \times f(\lambda_n) \end{bmatrix}$$

$$= \begin{bmatrix} \sum_i s_1(\lambda_i) f(\lambda_i) \\ \sum_i s_2(\lambda_i) f(\lambda_i) \\ \sum_i s_3(\lambda_i) f(\lambda_i) \end{bmatrix}$$

$$= \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

From the preceding equation, the result is that $S^T f$ gives the sensation of color for the spectrum defined by f. It should be clear from this that, if we have two spectral distributions f_1 and f_2 and $f_1 = f_2$, then this necessarily means that $S^T f_1 = S^T f_2$, or f_1 and f_2 give the same sensation of color. Also, because your color sensation is related to the sum of the product of two discrete functions S and f, it should also be evident even though f_1 and f_2 may not be the same, $S^T f_1$ can be equal to $S^T f_2$, suggesting that the two different spectral distributions could create the same color sensation.

This result is important because it now allows the color problem to be simplified. If the recording instrument is recording a spectral energy distribution f and the rendering device is generating a spectral energy distribution g, we do not have to make $f = g$, but g has to be generated such that $S^T f = S^T g$.

3 COLOR CALIBRATION

In this section, we use the tristimulus vector theory and show how it has been used to calibrate capture devices and rendering devices so as to solve the color problem. First, we need to understand the working of a capture device such as a camera and any rendering device such as a CRT or a projector.

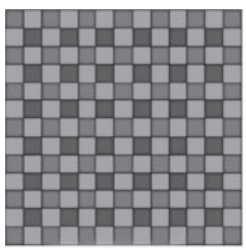
3.1 Color Cameras

The cameras we consider here for explanation are charge-coupled device (CCD) cameras. These were proposed as early as the 1970s and have now become commonplace in most modern applications, from consumer-level digital cameras to special-purpose cameras used in the fields of astronomy and microscopy. A CCD sensor consists of a rectangular grid of electron-collection sites laid over a thin silicon wafer to record the amount of light energy reaching each of them. When light photons strike the silicon, a charge is released proportional to the amount of light energy incident on the sensor area site. The image is read out of the CCD one row at a time, each row being transferred in parallel to a serial output register with one element in each column. Each light sensing element integrates the incident light over the whole spectrum, and as such each is essentially monochromatic. One sensor at each site on the grid thus produces one value, which is typically stored in 8 bits for black-and-white images. This design works for grayscale images but not for color images. For color images, each pixel location needs to have three samples based on the trichromacity theory explained previously.

Color CCD cameras use the same technology as black-and-white cameras, but need to be modified to sense three or more values at each grid point. There are various ways of doing this, depending on the correctness and hardware expense desired in the digitally captured image content. For most inexpensive consumer-level devices, a color grid that performs the role of a filter is overlaid over the silicon CCD sensors. Here, the filters are made of transparent materials that allow selective parts of the spectrum (red, green, and blue) to pass through. Ideally, each CCD sensor should sense all three filtered results, but practically only one of either the red, green, or blue filters is placed over each pixel area, allowing each CCD sensor to sense one part of the spectrum that is incident on it. Also, although each sensor has one filter, the filters are not equally distributed but organized in a pattern known as the Bayer filter pattern, which has more green filters than red and blue. This is shown in Figure 4-7. The bias toward green is because of the increased sensitivity needed for the green parts of the spectrum, as we shall see in Section 4.

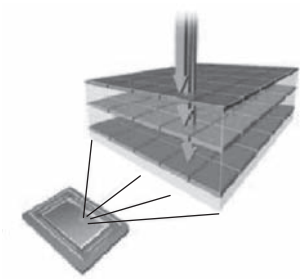
The advantage of the Bayer filter pattern method is that only one sensor is required and all color information (red, green, blue) is recorded simultaneously. This means that the camera can be smaller and cheaper, and, hence, this is a popular technique adopted in most consumer devices. However, each pixel area is still recording only one piece of information, either the red, the green, or the blue output. Once captured and digitized, the digital cameras need to use specialized demosaicing algorithms to convert the mosaic of separate colors recorded at each pixel into an equally sized mosaic of true colors having a red, green and blue value at each pixel. These values are determined by averaging

the values from the closest surrounding pixels. There are other ways of handling color in digital cameras. The Foveon systems capture red, green, and blue values by embedding these different spectral photo detectors in the same location at all pixels on the silicon wafer. This Foveon X3 technology works because red, green, and blue light components each penetrate silicon to different depths to release the appropriate electrical charges. Note that the Foveon system needs more incident light to produce equivalent pictures, as more light is absorbed. Other systems place a prism that splits the incident light via a prism into its constituent spectral components, which are then selectively sensed by different sensors. Such systems do provide better colors in digital imagery but need to reduce the resolution of the image because each pixel element requires more silicon real estate to provide a large enough sensing area. Recently, there have been systems that use a modified version of the Bayer pattern to use four filters—RGBE. The previous Bayer pattern's green filters have been split into two different filters to sense the green part of the spectrum more definitively by using an additional emerald filter E. This is seen in some SONY cameras, and claims to produce better color reproduction.

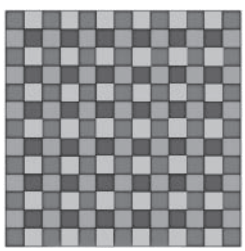


Bayer mosaic pattern

Bayer mosaic pattern overlaid on a CCD chip



Fovean X3 technology



RGBE mosaic pattern

Figure 4-7 Various filters used with CCD imaging devices to capture color. The upper-left image represents a traditional Bayer pattern. Note the biased locations of green compared with blue and red (see color insert in this textbook for full-color version of this image). The upper-right image shows the Bayer filter overlaid on a silicon CCD image capture plane. The lower-left image shows a tricolor capture at every pixel using the Foveon X3 technology. The lower-right image shows SONY's modified Bayer filter known as the RGBE filter.

Regardless of the technology used, the CCD units need to filter the incoming light ray, prior to sensing. These filters are physically manufactured elements, which are transparent and coated with material that blocks a certain part of the spectrum. Each area now has three outputs corresponding to the three filters. Natural questions regarding the nature of these filters are, how should their responses be defined and how should these be manufactured? The output of these filters is then further sampled and quantized producing values a_1, a_2, and a_3. The filters have a response, which is shown as m_1, m_2, and m_3 for the three different filters. Similar to Section 2, the cumulative filter effect can be modeled as follows:

$$a_i = \int_{\lambda=400}^{\lambda=700} m_i(\lambda) f(\lambda)\, d\lambda, \text{ or in discrete quantities}$$

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = M^T f = \begin{bmatrix} \sum_i m_1(\lambda_i) f(\lambda_i) \\ \sum_i m_2(\lambda_i) f(\lambda_i) \\ \sum_i m_3(\lambda_i) f(\lambda_i) \end{bmatrix}, \text{ where } M = \begin{bmatrix} m_1(\lambda_1) & m_2(\lambda_1) & m_3(\lambda_1) \\ m_1(\lambda_2) & m_2(\lambda_2) & m_3(\lambda_3) \\ .. & .. & .. \\ m_1(\lambda_n) & m_2(\lambda_n) & m_3(\lambda_n) \end{bmatrix}.$$

Different filters m_1, m_2, and m_3 will produce different values for a_1, a_2, and a_3. Thus, how a spectrum is translated into a trinumber representation largely depends on the filters used and their response.

3.2 Rendering Devices

Visual display devices attempt to create various spectra, using three electron guns in the case of a CRT. Each gun, let us call them p_1, p_2, and p_3, corresponds to a monochromatic wavelength in the red, green, and blue areas of the spectrum, respectively, and is capable of producing the monochromatic spectra g_1, g_2, and g_3. This is typically known as the RGB system, where all three guns simultaneously focus on a particular area of the phosphor screen and additively combine to produce a spectrum $g = g_1 + g_2 + g_3$. An illustration of how a CRT system works is shown in Figure 4-8. The spectrum is perceived by the observer to have a color corresponding to $S^T g$. The RGB system used in all cathode-ray tube (CRT) devices, including televisions and computer monitors, is a simple and direct approach to the problem of color description that incorporates the principles of additive color mixture.

The system described previously produces a spectrum g as the cumulative effect of g_1, g_2, and g_3. However, it needs to produce different spectra that can correspond to different colors. This is done by applying voltage gains v_1, v_2, and v_3 to the guns p_1, p_2, and p_3. Thus,

$$g = g_1 + g_2 + g_3 = v_1 p_1 + v_2 p_2 + v_3 p_3 = vp$$

Here, p_1, p_2, and p_3 are known as the primaries and correspond to fixed wavelengths. Every display device has a fixed set of primaries. Changing these primaries will lead

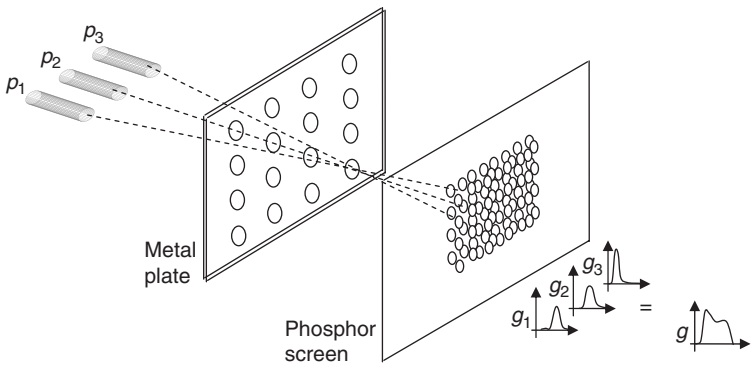


Figure 4-8 A typical CRT system showing primaries p_1, p_2, and p_3. The individual spectra g1, g2, and g3 produced by them get additively mixed to produce a combined spectrum g, which the observer will perceive as a color.

to changing the spectrum of each, and, consequently, with the voltage gains, it will lead to a different range of spectra and, ultimately, to a different perception of colors produced by the display system.

CRT displays are no longer the display of choice for consumer applications, and are no longer manufactured. The competing technologies are liquid crystal displays (LCDs) and plasma. Plasma technology works by sending electric pulses to individual pixel cells. These pulses excite xenon and neon gases, which then emit light. This light activates the proper ratio (defined in software) of red, green, or blue phosphors in each cell. LCD displays work as follows. Liquid-crystal-filled cells sandwiched between two sheets of glass receive voltage from an array of thin film transistors (TFTs). These crystals precisely orient to filter white light generated by a lamp behind the screen. LCD TVs use a subtractive color process: They selectively block wavelengths from the broad spectrum to achieve the right color mix. Plasma and LCD technologies produce increasingly similar performance, with some noted differences. Plasma screens have a better, darker black and the dynamic range (between white and black), also called contrast, is higher. The viewing angle is wider for plasma screens than for LCD screens. Plasma screens are better than LCDs at playing fast-moving video, as their response time is shorter. Plasma screens are subject to screen burn-in, which is a real problem if they are used as computer monitors. Plasma TVs consume about 33% more power than LCDs at equivalent resolution.

3.3 The Calibration Process

The objective of the calibration process is to ensure that the spectrum g, which is produced by the display device, generates the same color perception as the spectrum f, which was captured by the camera. In other words, we need

$$S^T g = S^T f$$

The camera captures a_1, a_2, and a_3 for a spectrum f falling at a pixel location. These are used as the voltage gains for the primaries in the display devices to produce the spectrum g. Thus, g is created as

$$g = g_1 + g_2 + g_3 = a_1 p_1 + a_2 p_2 + a_3 p_3$$
$$g = Pa$$

Display devices have a choice of primaries, whereas cameras have a choice of filters. To make $S^T g = S^T f$, the primaries p_1, p_2, and p_3 and the filter outputs a_1, a_2, and a_3 have the following relationship:

$$S^T g = S^T f, \text{ and } g = Pa$$
$$S^T P a = S^T f$$
$$a = (S^T P)^{-1} S^T f$$

Thus, to achieve the same color sensation, the primaries and the filters (m_1, m_2, and m_3) have to be related. This relationship is not uniquely defined. The CIE (Commission Internationale de l'Eclairage) solves this by fixing primaries to predefined standard values. Thus, if p_1, p_2, and p_3 are fixed for all display devices, the filter responses m_1, m_2, and m_3 can be computed. Although this requires the knowledge of the unknown S matrix, m_1, m_2, and m_3 can be computed by having an observer perform the function of S. This is done by color-matching experiments, as described in the next section.

3.4 CIE Standard and Color-Matching Functions

The variation of primaries in the display devices, along with the variation of filters in the capture devices makes it impractical to manufacture standardized capture and display devices such that colors captured by any capture device match the colors rendered by any display device. The CIE (Commission Internationale de l'Eclairage) solved this problem by proposing that the primaries p_1, p_2, and p_3 be fixed. This opened the way to establishing an internationally agreed upon system for color. The primaries selected by CIE were $p_1 = 700.0$ nanometers (red), $p_2 = 546.1$ nanometers (green), and $p_3 = 435.8$ nanometers (blue). The fixed p_1, p_2, and p_3 primary values made it possible to calibrate filter responses. This is done by involving a standard observer in color-matching experiments, which has led to all the color space representations commonly used in colorimetry today.

The color-matching experiment has a setup where a CRT-based rendering device is capable of rendering colors using the fixed primaries p_1, p_2, and p_3. These colors, as explained in the previous section, are those perceived as $S^T g$, where the spectrum g is generated as $g = a_1 p_1 + a_2 p_2 + a_3 p_3$. Here a_1, a_2, and a_3 are the gains applied to primaries. A standard observer can control these a_1, a_2, and a_3 values to generate different spectra and, hence, observe different colors. In the experiment, the standard observer is shown a spectrum f and is made to adjust a_1, a_2, and a_3 so that the spectrum generated at the CRT using the CIE defined primaries and the observer inputs a_1, a_2, and a_3 have the same color sensation as the spectrum f shown to him. In other

words, the observer is ensuring that $S^T g = S^T f$. The spectrum f shown to the observer can be understood by the equation described in Section 3.1.

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = M^T f = \begin{bmatrix} m_1(\lambda_1) & m_1(\lambda_2) & \ldots & m_1(\lambda_n) \\ m_2(\lambda_1) & m_2(\lambda_2) & \ldots & m_2(\lambda_n) \\ m_3(\lambda_1) & m_3(\lambda_2) & \ldots & m_3(\lambda_n) \end{bmatrix} \times \begin{bmatrix} f(\lambda_1) \\ f(\lambda_2) \\ \vdots \\ f(\lambda_n) \end{bmatrix}$$

In the preceding equation, the M matrix is unknown and consists of $3xn$ unknown values. For one spectrum f shown to the observer, we can compute the $f(\lambda_1), f(\lambda_2), \ldots f(\lambda_n)$ values via a spectrometer. The observer adjusts the a_1, a_2, and a_3 values to make the colors match and each color reading yields three equations. Therefore, with n color reading, that is, n triplets of a_i and correspondingly n spectra values for f, the system of equations defined previously are well constrained and functions m_1, m_2, and m_3 can be obtained. These three responses provide a way to map any input spectrum to a tristimulus representation in a CIE-based space called the XYZ color space. Figure 4-9 shows a visualization of this XYZ color space, which is arrived at when m_1, m_2, and m_3 values are plotted on the three axes. There are several CIE-based color spaces, but all are derived from the fundamental XYZ space.

An important result of the XYZ color space worth mentioning here is that the solution obtained by the color-matching experiments for m_1, m_2, and m_3 are not physically realizable, that is, it is an imaginary color space and only exists in a mathematical realm, but it can represent all the colors of the visible spectrum. This is because it was obtained by showing the observer a number of real colors. This space is typically used to study color ranges and representations of other color spaces, but it cannot be used in practice. In the next section we explain the derivation of useful and practical color spaces from this XYZ color space.

4 COLOR SPACES

A color space is a model in which colors are represented in terms of fixed intensity values. In theory, a color space can be one-, two-, three-dimensional, or even more, depending upon the number of components chosen. Three-dimensional color spaces are typically used to describe the choice and range of commonly perceived colors. The most common method of identifying color in such color spaces is a three-dimensional coordinate system. The area enclosed by the color space gives all possible colors that the color space can represent and is termed as the *color gamut* of that space.

So far, we have arrived at the representation of the CIE XYZ color. A graphical illustration of all the colors that this color space spans in terms of its X, Y and Z color channels is shown Figure 4-9. Although this space(s) theoretically suffices for describing color, it cannot be used practically because the primaries X, Y, and Z are not physically realizable. Numerous other color spaces have been designed for different practical applications. These have realizable primaries, but as a result cannot represent some colors in the visible spectrum. Some commonly used color spaces are described in the following sections.

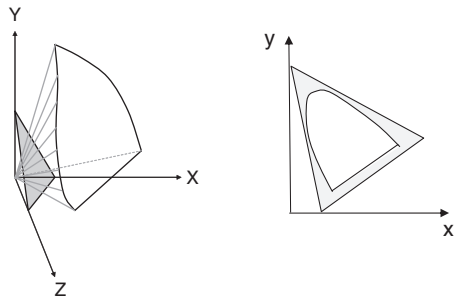


Figure 4-9 The left image shows the volume of all visible colors in CIE XYZ coordinate space. This is a cone with its vertex is at the origin. Usually, it is easier to suppress the brightness of a color, which we can do because perception of color is approximately linear. We do this by intersecting the cone with the plane X + Y + Z = 1 to get the CIE xy chromaticity space shown in the image to the right.

4.1 The CIE XYZ Color Space

The CIE XYZ color space is a popular standard. The XYZ space allows colors to be expressed as a mixture of the three tristimulus values X, Y, and Z, which are the color-matching functions arrived at by the CIE as explained in the previous section. The CIE defined the primaries so that all visible light maps into a positive mixture of X, Y, and Z, and so that Y correlates approximately to the apparent lightness of a color. The mixtures of X, Y, and Z components used to describe a color are expressed as percentages ranging from 0 to 100.

There are also other color spaces derived directly from the XYZ space. These spaces are primarily used to relate some particular aspect of color to the XYZ values. One common example of this is the *xy* color space, also known as the chromaticity space or the *Yxy* space. In this space, the three-dimensional XYZ color space is normalized and transformed into a 2D space where the luminous or lightness effect is removed. This can be obtained by intersecting the XYZ space with the plane X + Y + Z = 1. The resulting space is shown in the left image of Figure 4-9 and can be described by the coordinates.

$$(x,y,z) = \left(\frac{X}{X+Y+Z}, \frac{Y}{X+Y+Z}, \frac{Z}{X+Y+Z}\right)$$

Z can now be defined as

$$Z = 1 - X - Y$$

The *z* component bears no additional color information and is often omitted. Note that because *xy* space is just a projection of the 3*D XYZ* space, each point in *xy* corresponds to many points in the original space. The missing information is luminance *Y*. Color is usually described by *xyY* coordinates, where *x* and *y* determine the chromaticity and *Y* the lightness component of color. In Figure 4-10, we show the color representation of the XY color space. Also shown are the less commonly used XZ and ZY color spaces, which can be similarly derived. Another noteworthy observation is that none of these color spaces show the color black or any shades of gray. Since the derivation of all these chromaticity spaces involves a projection of the XYZ color cone onto a plane, the color spaces contain only pure chroma.

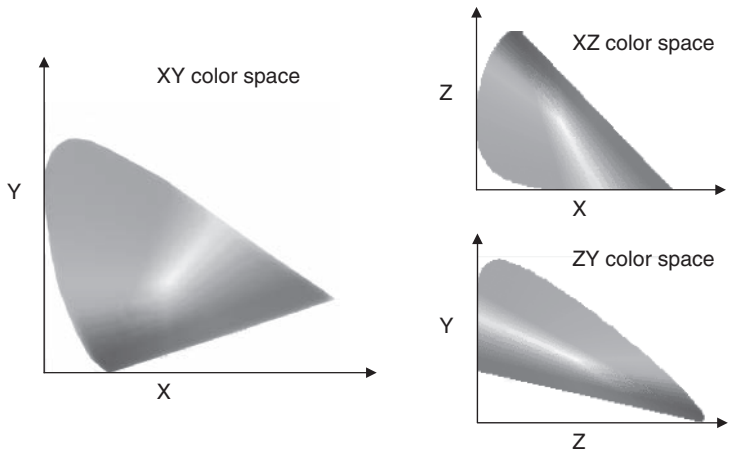


Figure 4-10 The left figure shows the more commonly used XY space. The periphery of the XY space consists of the wavelengths of the visible spectrum. The right figure shows two other XZ and ZY color spaces, also derived from the XYZ color space. See the color insert in this textbook for a full-color version of this image.

4.2 RGB Color Space

The RGB color space is a linear color space that formally uses single wavelength primaries (645.16 nm for R, 526.32 nm for G, and 444.44 nm for B). The different spectra are generated by voltage gains applied to these primaries, and the resulting colors are usually represented as a unit cube—usually called the RGB cube—whose edges represent the R, G, and B weights. Here, red is usually shown as the x-axis, green being the y-axis, and blue being the z-axis, as in Figure 4-11. The diagonal line, if you imagine it, connecting the black color (0,0,0) to white color (1,1,1) is made up of all gray colors.

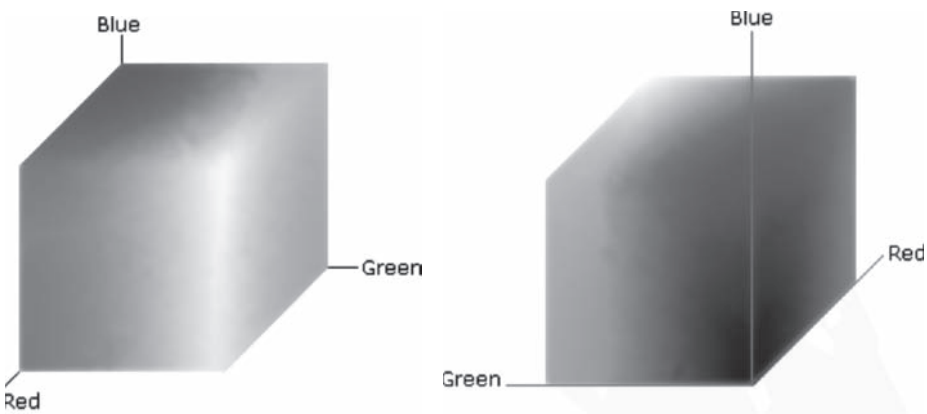


Figure 4-11 Like most color spaces, the RGB color space is normalized. That is, all color values are restricted to the range of zero to one inclusive. So black is (0.0, 0.0, 0.0), and white is (1.0, 1.0, 1.0). In the RGB color space, the primary colors are red, green, and blue. The secondary colors are cyan, yellow, and magenta. See the color insert in this textbook for a full-color version of this image.

The RGB color is the system used in almost all color CRT monitors, and is device dependent (that is, the actual color displayed depends on what monitor you have and what its settings are). It is called additive because the three different primaries are added together to produce the desired color. Correspondingly, there is also a subtractive color space known as the CMY space, which is discussed in the next section.

4.3 CMY or CMYK Color Space

The CMY color space stands for cyan, magenta, and yellow, which are the complements of red, green, and blue, respectively. This system is used for printing. Sometimes, this is also referred to as the CMYK color space, when it refers to black as part of it. The CMY colors are called *subtractive primaries*; white is at (0,0,0) and black is at (1,1,1). If you start with white and subtract no colors, you get white. If you start with white and subtract all colors equally, you get black.

It is important to understand why the CMY color space is used in printing. If white light is incident on a printed paper, the color ink pigments absorb part of the spectrum and the complement of the incident white light spectrum is reflected to the observer's eye. Thus, if an observer has to perceive a color, which is represented by three RGB coordinates, then the ink that prints on the paper has to absorb the complement of that RGB color, which is the color's CMY equivalent. Figure 4-12 shows how CMY and RGB relate in terms of additive and subtractive color mixing.

In the RGB color space, if all the components R, G, and B are used equally, they all combine additively to form white, as shown in the central area of the left image in Figure 4-12. In the CMY systems, the C, M, and Y combine subtractively to form black. While printing, to print white in CMY is very trivial and can be obtained by setting C = M = Y = 0 (that is, no pigment is printed). Conversely, equal amounts of

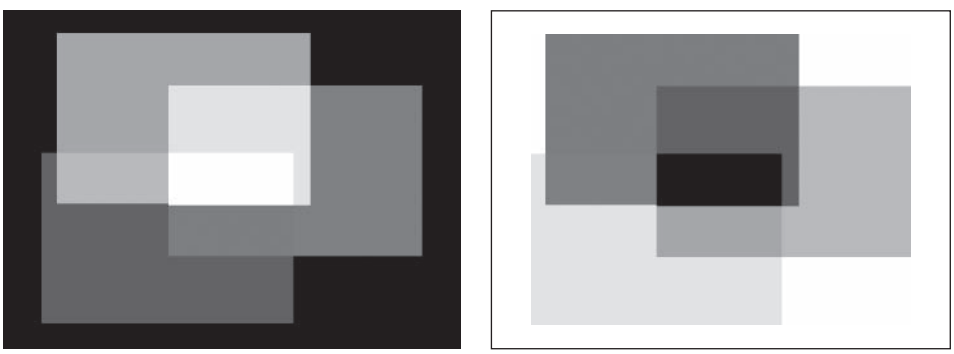


Figure 4-12 Additive (left) and subtractive (right) color mixing (see color insert in this textbook for full-color version of this image). In additive color mixing, the left figure shows the outer black (no color) rectangle with three primary RGB rectangles. Their additive effects are shown. Correspondingly in the subtractive case (right), the outer rectangle is white (no color) and the three CMY rectangles have subtractive effects as shown. In the full-color version of this image, you can see that each color on the left is the complement of the corresponding color on the right.

C, M, and Y should produce black. Producing black color is very common in printing, and it is impractical to use all three C, M, and Y pigments each time to produce black. First, consuming all three pigments to produce black can prove expensive. Second, while in theory black is produced this way, in practice, the resulting dark color in neither fully black nor uniform. Hence, a fourth pigment K is commonly used in the CMY system to produce true black. Use of K along with CMY generates a superior final printed result with greater contrast.

The conversion from RGB to CMY can be defined as follows: Let C_{RGB} be a color in RGB and the corresponding representations in CMY and CMYK by C_{CMY} and C_{CMYK}. Then

$C_{RGB} = \{R, G, B\}$
$C_{CMY} = \{C, M, Y\} = \{1\text{-}R, 1\text{-}G, 1\text{-}B\}$, and

C_{CMYK} is given by

If min(C, M, Y) = 1 then $C_{CMYK} = \{0,0,0,1\}$ else K = min (C, M, Y) and

$$C_{CMYK} = \left\langle \frac{C - K}{1 - K}, \frac{M - K}{1 - K}, \frac{Y - K}{1 - K}, K \right\rangle$$

So far, we have talked about the importance of CMYK in printing and its relationship to RGB. The color separation process for printing also needs to be understood. All color documents stored in a digital format are normally stored using RGB for display purposes. For printing purposes, these documents are required to be converted into the CMYK color space. Printing devices print by using inks that correspond to the cyan, magenta, yellow, and black pigments. However, they do not use all inks at the same time, but one at a time. While printing, each separated color component is made into a film that is used to prepare a printing plate for that color as shown in Figure 4-13. The final image is achieved by printing each color plate, one on top of another in a sequential manner. This is similar to layering colored transparencies on top of each other in the specific order of cyan, magenta, yellow, and black.

4.4 YUV Color Space

The YUV and YIQ are standard color spaces used for analog television transmission. YUV is used in European and Asian TV standards (such as PAL and SECAM) and YIQ is used in the North American NTSC television standard. Y is linked to the component of luminance, and U and V or I and Q are linked to the components of chrominance. Y comes from the XYZ standard explained previously. Another similar space, the YCrCb space is used in the JPEG and MPEG compression standards for images and video, respectively.

The YUV space has a very practical bandwidth-saving usage. It is known from visual experiments that humans are not as susceptible to changes in chrominance as they are to luminance. In other words, luminance changes are captured by the human

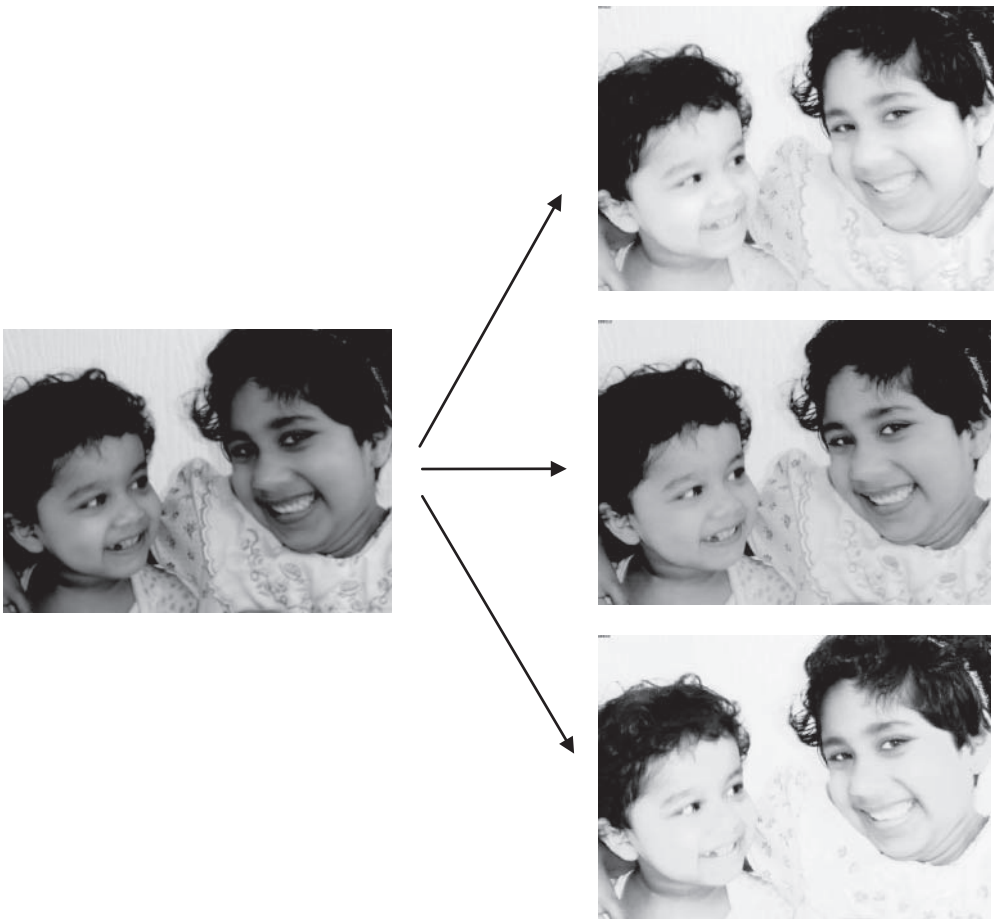


Figure 4-13 Color separations. The original left image is shown separated into three different color duotone color plates according to the CMYK model (see color insert in this textbook for full-color version of this image). The upper-right image shows cyan+black, the middle image shows magenta+black, and the bottom image shows yellow+black.

eye more precisely than chrominance changes. Because humans are less tolerant to chrominance, it might be worth transmitting less color information than luminance information. The YUV color space allows the representation of a color in terms of its luminance and chrominance separately, thus allowing the chrominance information to be subsampled in video signals. Correspondingly, we have the YUV 4:2:0 and YUV 4:2:2 standards, as explained in Chapter 3.

The RGB space can be transformed easily into these sets of spaces by a simple linear transform. An example of this transformation from RGB to YUV is as follows:

$$T_{RGBtoYUV} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & 0.312 \end{pmatrix}$$

4.5 HSV Color Space

The representations of colors in the linear spaces mentioned so far—XYZ, RGB, CMY—are designed for devices to make capture, rendering, and printing possible. But for a human observer, the colors defined in these spaces might not necessarily encode properties that are common in language, or that are important in digital imagery applications, where intuitive user interfaces are needed by artists to select and modify colors. Useful color terms include the following:

- *Hue*—The property of a color that varies in passing from red to green. Roughly speaking, hue represents the dominant wavelength in spectral power density of that color.
- *Saturation*—The property of a color that defines the colorfulness of an area judged in proportion to its brightness; for example, red is more saturated than pink. Roughly speaking, the more the spectral energy that is concentrated at one wavelength, the more saturated the associated color. You can desaturate a color by adding light that contains power at all wavelengths (white).
- *Brightness*—Also sometimes called *lightness* or *value*, the property that varies in passing from black to white.

For example, if you are interested in checking whether a color lies in a particular range of reds, you might want to encode the hue of the color directly and then see how its saturations and brightness can be adjusted to select your desired color or range. This can be done rather intuitively in the HSV color space which is shown in Figure 4-14.

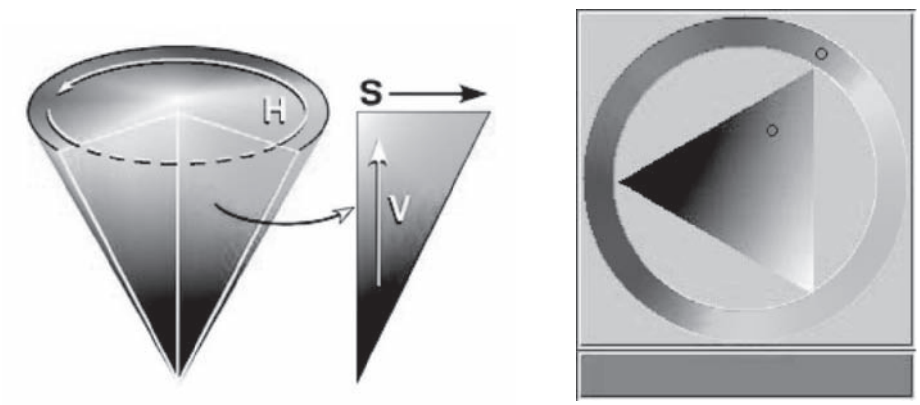


Figure 4-14 Left: A common depiction of the HSV color space. The circle on the top shows all possible hues along the periphery (see color insert in this textbook for full-color version of this image). The distance from the periphery to the center of the circle defines the amount of saturation. Scaling the circle along the vertical cone axis gives a measure of the brightness. One cross for a select green H is shown, where varying the S and V produces the colors shown. Right: Another depiction of the HSV color space. The circle describes the hue. Selecting a hue orients the inner triangle with one corner at the selected hue. The distance from the other two corners measures the amount of saturation and brightness.

The color spaces discussed thus far are mostly linear, that is a final color is obtained by the combined linear effect of three channels. One difficulty with linear

color spaces has to do with the fact that the individual coordinates do not capture human intuitions about the topology of colors; it is a common intuition that hues form a circle like the wheel or circle shown on the cone in Figure 4-14. As you walk around the circle by equal distances, the hue changes from red, to orange, to yellow, and then to green, and from there to cyan, to blue, to purple, and then to red again. Another way to think of this is to think of local hue relations: red is next to purple on this color circle and orange; orange is next to red and yellow; yellow is next to orange and green; green is next to yellow and cyan; cyan is next to green and blue; blue is next to cyan and purple; and purple is next to blue and red. However, as we shall see in the following section, this linear relationship is not linear.

4.6 Uniform Color Spaces

Evaluating and comparing color differences is needed in many applications. The color spaces dealt with so far do not make this a necessarily easy task. To explain this further, consider the following experiment in chromaticity space. An observer might be shown a color, which is then progressively modified ever so slightly until a color change is perceived by the observer. When these differences are plotted on a color space, they form the boundary of a region of colors that are indistinguishable from the original colors. Usually, ellipses are fitted to the just noticeable color differences. It turns out that in CIE chromaticity space, these ellipses depend strongly on where in the space the difference occurs, as illustrated in Figure 4-15.

Figure 4-15 suggests that given two colors having coordinates (x_1, y_1) and (x_2, y_2), the size of the color difference between them $(x_2 - x_1)^2 + (y_2 - y_1)^2$ is not a good indicator of the perceived color difference. If it were a good indicator, the ellipses would be circles. A uniform color space is one in which the distance in coordinate space accurately reflects the perceived difference between the colors. In such a space, if the distance in coordinate space were below some threshold, a human observer would not be able to tell these colors apart.

The CIE LUV space is a uniform color space that has the above-mentioned property and can be obtained by transforming the chromaticity (X,Y) color space as follows:

$$(u,v) = \left(\frac{4X}{X + 15Y + 3Z}, \frac{9Y}{X + 15Y + 3Z}\right).$$

Although this transformation suffices as a fair indicator of the color (chroma) difference, it omits any differences in brightness. The CIE LAB is a popular uniform color space that deals with luminance as well as chromaticity by mapping the XYZ coordinates:

$$L^* = 116\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - 16$$

$$a^* = 500\left[\left(\frac{X}{X_n}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}}\right]$$

$$b^* = 200\left[\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - \left(\frac{Z}{Z_n}\right)^{\frac{1}{3}}\right].$$

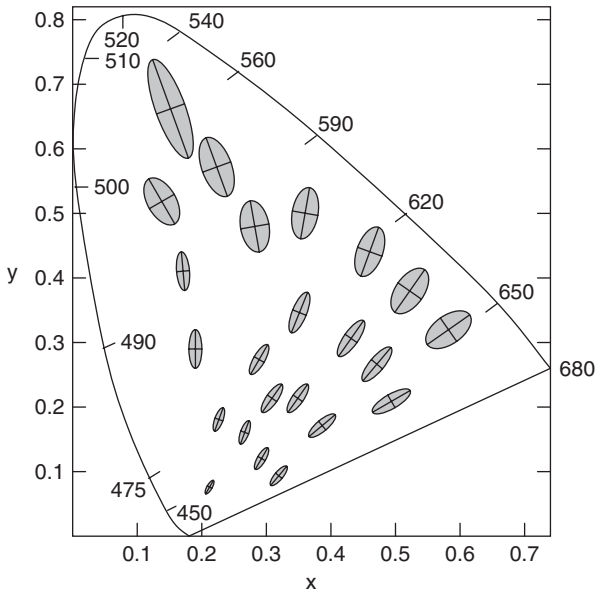


Figure 4-15 MacAdam ellipses. The ellipses shown signify variations in color matches in chromaticity space. The center of each ellipse represents the color of a test light; the size of the ellipse represents the scatter of lights that the human observers tested would match to the test color; the boundary shows where the just noticeable difference is. The ellipses have been magnified considerably for clarity. Notice that the ellipses at the top are larger than those at the bottom of the figure, and that they rotate as they move up. This means that the magnitude of the difference in x, y coordinates is a poor guide to the difference in color.

where X_n, Y_n, and Z_n are the X, Y, and Z coordinates of a reference white patch and that both (X/X_n) and (Y/Y_n) are > 0.008856. The reason to care about the LAB space is that it is substantially uniform. In some problems, it is important to understand how different two colors are as perceived by a human observer, and differences in LAB coordinates give a good guide.

4.7 Device Dependence of Color Spaces

Based on the CIE standard, devices are supposed to use the same primary frequencies R, G, and B as their basis for producing the gamut or ranges of colors. In practice, obtaining the same R, G, and B frequencies might not be possible because of manufacturing problems and the different devices might have very similar, though not the necessarily the same frequencies for R, G, and B. This means that RGB and CMY color spaces vary from monitor to monitor and from printer to printer and, hence, are called device-dependent color spaces. This device dependence of color is induced by a variety of reasons, such as not using the exact primaries, temperature-based changes in color frequencies, and monitor gammas.

5 GAMMA CORRECTION AND MONITOR CALIBRATION

The cathode-ray tube (CRT) is a display device that has been commonly used in most computer displays, television receivers, oscilloscopes, and many other devices. It was developed by Karl Ferdinand Braun in 1897. A CRT monitor has a phosphor-coated screen, which emits a visible light when excited by a beam of electrons. The light has a particular spectrum and is perceived to have a color depending on the voltage applied to the primary electron guns, as we have seen in Section 4. The relationship between the luminance generated by a device and the actual signal it is supposed to render is known as the transfer characteristic (or transfer function) of the device. The luminance generated by a CRT, or any phosphor display device, is generally not a linear function of the applied signal because the phosphor medium does not respond linearly. For example, if a computer reads a luminance value from an image and sends it to the display device, the displayed color tends to look dimmer than in the original photograph. Also, the amount of dimness varies from color to color. Brighter colors are perceived to be dimmer than darker colors. This change in the generated color is normally adjusted by nonlinearly increasing (or decreasing) the voltage applied to the primaries of the device. This nonlinearity is modeled as a power-law response to the luminance voltage. For each gun, the power of the emitted light $S_i(\lambda)$ is a function of the control voltage v_i, which is approximated as follows:

$$S_i(\lambda) = \left(\frac{v_i}{v}\right)^{\gamma} p_i(\lambda)$$

where v is the maximum value of the voltage and γ is the exponential factor, also known as the monitor's gamma. The previous transformation is called gamma correction with γ termed as the gamma value. The gamma value, therefore, defines the relationship between the amount of light given out from the device and the 8-bit RGB values in the original signal. Typical CRT displays have gamma values between 1.5 and 3.0 and the value is constant for all the channels; 2.2 is a frequently used standard. Some graphics hardware partially compensates for this response function, bringing the effective system gamma down to 1.8 or 1.4. The correct display of images, whether computer-generated or captured, requires proper correction for the system's gamma response.

The other types of monitors used today include liquid crystal displays (LCDs) and plasma displays. Displays made using organic light-emitting diodes (OLEDs) are also destined to become another practical display technology due to their low power consumption and portability. LCD monitors were initially used with laptop computers and not for any serious color evaluation. Hence, their transfer characteristics and the need to use it to calibrate colors were mostly ignored. Besides, most LCD screens were poorly suited for image editing operations and evaluation because the light they give out varies with the viewing angle. However, as the technology improved and as larger LCD displays began making their way into homes either for use with desktop computers or as television monitors, manufacturers were forced to take a closer look at the color calibration issue. Unlike CRT displays, liquid crystal displays do not follow a pronounced nonlinear luminance-voltage transfer function that can be emulated by the power law function. To produce correct chrominance and luminance, LCD

displays typically have gamma lookup tables, one table for each of the primary R, G, and B colors. These lookup values are arrived at statistically and work well when the color of a pixel does not change. However, if the input signal has colors changing at a pixel, the voltage used to light a pixel is influenced not only by the gamma lookup table, but also to a small extent by the voltage used to light the previous color. This residual voltage causes color cross talk, which is hard to eliminate or compensate using lookup tables for a single color. Thus, in the case of LCD displays, gamma tables have been useful but at the same time are unable to produce high-quality color characteristics as seen in CRT displays. Plasma displays are also known to use similar lookup tables for color correction. This is the fundamental difference in the way gamma correction is done between older CRT monitors and the recent digital displays. The former adjusts the voltage gains directly on the electron guns, whereas the latter goes through a lookup process.

6 EXERCISES

1. [**R02**] The chromaticity diagram in (x, y) represents the normalized color-matching functions X, Y, and Z. Prove that

$$Z = [(1 - x - y)/y]\ Y.$$

2. [**R03**] Here, you are tasked with mapping the gamut of a printer to that of a color CRT monitor. Assume that gamuts are not the same, that is, there are colors in the printer's gamut that do not appear in the monitor's gamut and vice versa. So, to print a color seen on the monitor, you choose the nearest color in the gamut of the printer.
   - Is this a good or bad algorithm in terms of reproducing colors?
   - Will this work better for constant color tones or varying color tones?
   - Can you suggest improvements rather than just choosing the nearest color?
3. [**R03**] The chromaticity space is a 2D space obtained by projecting the 3D conical XYZ space onto a plane.
   - What is the equation of this plane?
   - Why is there no black color in the chromaticity space?
   - Where do the CIE primaries lie in the chromaticity space?
4. [**R06**] The artist within you has definitely played with colors and mixed them to produce various different colors. Here are a few observations that might need further exploration:
   - When you mix blue color and yellow color pigments, the resulting color formed is green. Can you explain why this is so?
   - Instead of pigments, if you take torches that shine the same colored lights, blue and yellow, and shine them on a white wall on the same area, what color will you see? Why is it different from the previous answer?
5. [**R05**] One of the uses of chromaticity diagrams is to find the gamut of colors given the primaries. It can also be used to find dominant and complementary

colors. The dominant color of a given color D (or dominant wavelength in a color D) is defined as the spectral color that can be mixed with white light to reproduce the desired D color. Complementary colors are those that when mixed in some proportion, create the color white. Using these definitions and the understanding of the chromaticity diagram that you have, answer the following questions:

- In Figure 4-16 find the dominant wavelength of color D. Show this wavelength. Assume that E denotes the equiluminous that represents white color.
- Do all colors have a dominant wavelength? Explain your reasoning.
- Find the color that is complementary to the color C. Show this color.

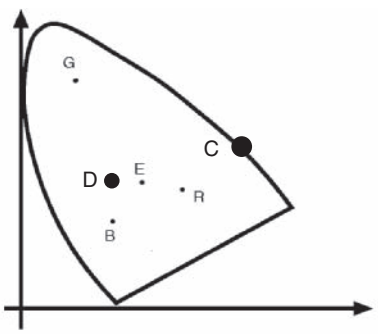


Figure 4-16

6. [**R08**] (*This exercise might require some understanding of nonlinear transforms.*) Here R, G, and B are three primaries already positioned on the chromaticity diagram in Figure 4-17. We have also selected an eqiluminous, which is the point where all the three R, G and B primaries have the same weight is denoted by the point E. Note that this is a nonlinear space, so the position of the primaries does not necessarily make the triangle equilateral.
   - Calculate the locus of points whose R = 0.5. Draw the locus in the given figure. Explain your reasoning.

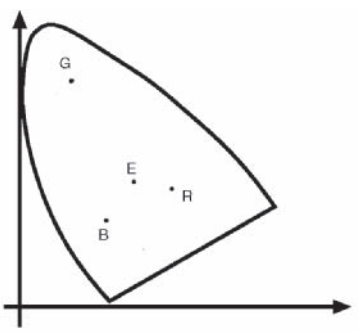


Figure 4-17

- Calculate the locus of points whose R = 0.75. Draw the locus in the given figure. Explain your reasoning.
- Given the previous R = 0.5 locus, how does this line map in the RGB space? Explain the mapping from the chromaticity diagram to the RGB space.

7. [**R08**] In this chapter, we discussed the CIE XYZ space and the RGB space, both being linear spaces. Derive the equations for transforming the RGB space to the XYZ space and vice versa. Remember that is a linear transformation!

8. [**R08**] A color defined by three primaries. P_1, P_2, and P_3 can be represented as a linear combination of those primaries. Every primary P_i is also a color in that color space and represented as $P_i = (x_i, y_i, z_i)$. Given any color in that C(x, y, z), we know that as long it lies in the gamut defined by P_1, P_2, and P_3, it can be represented as a linear combination of them (for example, $C = \text{sum}[\alpha_i P_i]$).
   - Find the chromaticity coordinates of P_1, P_2, and P_3.
   - Find the chromaticity coordinates of C in chromaticity space in terms of the chromaticity coordinates of P_1, P_2, and P_3.
   - Prove that the chromaticity coordinates of any color C (which is a linear combination of primaries P_1, P_2, and P_3) can be represented as linear combinations of the chromaticity coordinates of the respective primaries.

9. [**R06**] Suppose you have two CRTs, one with primaries P_1, P_2, P_3 and the other with P_4, P_5, P_6. Each P_i has a different location on the CIE chromaticity space with the equiluminous points E_{123} and E_{456} as shown in Figure 4-18 and Figure 4-19.

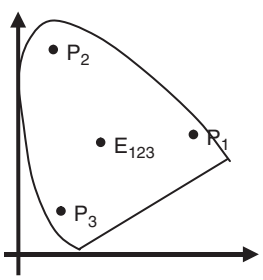


Figure 4-18

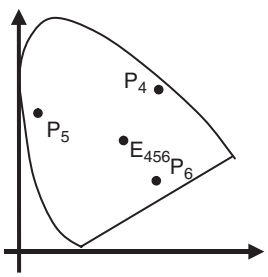


Figure 4-19

- Define what is meant by the color gamut of a CRT screen. Draw gamuts of the two CRTs in the figure.
- Now suppose you want to manufacture one new unconventional CRT that will have six primaries chosen to be the ones above. What will be the combined gamut? Draw it.
- Is the new gamut always greater than, less than, or equal to the individual gamuts? Explain. Will the color corresponding to E_{123} be the same as the color corresponding to E_{456}? Explain.

PROGRAMMING ASSIGNMENTS

10. [**R08**] This programming exercise will help you understand a practical problem in color conversion. The source code to read a display of a color image has been supplied. All source code is available under the Student Downloads section of www.cengage.com.
    - Adapt the code to convert the color image to a black-and-white image. Now think of the reverse situation. There are many old printed black-and-white photographs that can be scanned in a digital image. Also there are many old black and white movies available. You want to convert a black-and-white photograph to a color one.
    - Given a grayscale image, how would you approach the problem of converting it to a color image? What do you need to do to the input image, and how will you go about giving it a color? Is this mapping unique?
    - Modify your program to read in a black-and-white image and have a way of turning it into a color image.
11. [**R06**] The pseudo code shown in Figure 4-20 converts a color in the RGB color space to the HSV color space. Use this to write a program to convert RGB values to HSV values.

```
if (R > G) then
        Max = R;Min = G; position = 0;
else
        Max = G;Min = R; position = 1;
end
if (Max < B) then
        Max = B; position = 2
end
if (Min > B) then
        Min = B;
end
V = Max;
if (Max != 0) then
        S = (Max − Min)/Max ;
else
        S = 0;
end
if (S != 0) then
        if (position = 0) then
                H = (1+G-B)/(Max-Min);
        else if (position = 1) then
                H = (3+B-R)/(Max-Min);
        else
                H = (5+R-G)/(Max-Min) ;
        end
end
```

Figure 4-20

- What does the primary value of red in RGB map to in the HSV system?
- Given an RGB color value, how can you use this program to find the dominant wavelength in the color?

12. [**R07**] Here, you are asked to simulate a demosaicing algorithm that takes place in a real digital camera. As explained in the text, consumer-level cameras do not capture three color channels per pixel, but have Bayer filter that captures only one value, either R, G, or B at each pixel. Refer to the grid shown in 0. After these values are captured, they are then interpolated to create RGB values for each position, which is what you want to implement here. Correspondingly, complete the following tasks:

- Implement a program that reads an RGB image and creates an intermediary image that captures individual R, G, or B at pixels according to the Bayer pattern grid. That is, create a new 8 bits per pixel image from the 24 bits per pixel image by throwing away the 16 bits at each pixel and keeping only the appropriate 8 bits (either for R, G or B) according to the Bayer pattern.
- Use this intermediary image to create another RGB image by interpolation of the individual values. A G value at every pixel where G is not defined can be obtained by bilinear interpolation as follows:

$$G(i.,j) = \frac{1}{4}\sum G(i + m, j + n)$$

where $(m,n) = [(0,-1), (0,1), (-1,0), (1,0)]$.

Similarly, bilinear interpolation schemes can be appropriately defined for the R and B channels. But these are less regular and will need more than four samples. One such possible scheme for a green pixel that needs an R value is as follows:

$$R(i.,j) = \frac{1}{8}\sum R(i + m, j + n)$$

where $(m,n) = [(1,0), (-1,0), (3,0), (-3,0), (-1,2), (1,2), (-1,-2), (1,-2)]$.

Implement suitable bilinear filters to fill the whole image grid to get RGB values at each pixel.

- The preceding formulas assume no correlation between the individually captured R, G, and B channels and weight all the neighbors equally. But you should get better results by assuming that the color channels are correlated. How can you extend the preceding scheme to get better results?

This page intentionally left blank

CHAPTER 5

Multimedia Authoring

Chapter 1 gave a broad overview of multimedia. Multimedia content consists of one or more of the media types—text, audio, images, video, and 2D/3D graphics—assembled into a coherent and interactive presentation. This chapter explains how the media types can be individually or collectively put together to create multimedia content. Creating useful content can be viewed as a production, which needs knowledge of media types, art, and, in many cases, programming and scripting. Also, the production often turns out to be a time-consuming task depending on the amount of media information that needs to be captured or collected, the structural and temporal organization/layout requirements, and various means of interaction that need to be mapped out. These tasks normally make use of authoring tools, which are software programs that help an author create and publish content.

From an authoring perspective, multimedia content can be created for a passive viewing or a passive experience where the user's action has no effect. Examples of these include watching a video, reading a digital document, and even watching a movie in theaters. In all these examples, the viewer cannot alter the planned flow of data, either spatially or temporally, and the result is a linear experience. Multimedia presentations can also be authored so as to have a more active participation of the viewer or reader. Examples of these include a hyperlinked digital document such as a Web page, a video game, or a DVD formatted movie. In all these cases, an action from the user results in a different pathway where the media elements shown might change in position spatially and/or in time temporally. This results in a nonlinear experience. Whether passive or interactive, the content is normally authored using software tools that help create, set up media elements, and combine them as per needs.

It is not easy to make powerful authoring tools, which allow the user to create all types of content that involve the different media types. Often, tools are designed to work on a specific media type, such as Adobe Photoshop, which works with images,

Avid, which works with standard- and high-definition digital video and audio, or Autodesk Maya, which provides ways to create and render 2D and 3D animations. Other authoring tools bring the individually created content together and provide the capability to set up the different media information structurally and provide a way to author interactivity. To provide these workflows, authoring tools make use of user interfaces and specific authoring paradigms, which might also involve scripting, programming, for example Java scripting, and Java applets.

This chapter starts by describing a few examples of multimedia documents and applications. This is meant to elicit an understanding of the variety of functionality required, which will better portray the requirements expected of content creation tools to create multimedia content. In Section 2, we formally garner a few requirements common to multimedia authoring tools, and explain them in the following sections. An outcome of describing the requirements relates to the issues associated with intramedia and intermedia authoring. These are discussed in Sections 3 and 4, respectively. Section 5 explains a few common authoring metaphors that most multimedia authoring tools need. In Section 6, we attempt to discuss user interfaces, but rather than concentrating on generic principles to design user interfaces, we explain a few principles to design mobile interfaces and guidelines needed to use multiple devices as user interfaces. Finally, in the last few sections, we explain advanced authoring concepts, such as device-independent authoring, multimedia asset management, and a variety of multimedia services, which along with the discussed traditional multimedia authoring topics provide invaluable experiences to users.

1 EXAMPLES OF MULTIMEDIA

A common multimedia example today is creating a digital document. This might be a Web page, an article to be published in an online magazine, or even a book. In most cases, information is presented using text, images, and perhaps even video and graphics animations. The interactive aspects make use of hypertext and hyperlinks, which allow the user to nonlinearly read through the presentation. You might also use programming and scripting to illustrate concepts and interactivity. If we ignore the scripting aspects, the presentation includes text laid out with other media elements, which at times are hyperlinked. Another relevant aspect relates to the size of the multimedia files, as they need to match the specific bandwidth for delivery. It might be intended for Internet viewing, transmitted along high-bandwidth cable networks or lower-bandwidth cell phone networks. In such cases, the author and/or authoring tools need to be aware of compression and quality related to compression.

Figure 5-1 shows a common example of what a Web page looks like today. You can see that compiling the myriad of text, images, and video elements along with its formatting and interactive hyperlink setup can be a cumbersome task. This is especially so when the content is changing on a dynamic basis over time. The example shown in Figure 5-1 is an instance of the Web site *www.nytimes.com*. The information shown changes all the time. This task can be simplified by having proper authoring tools so that the process can be automated to a large extent.

Figure 5-1 Example of a typical Web page today with text, image, and video formatted together with hyperlinks set up. Creating such a document can be a nontrivial task without the use of appropriate tools to put the different media types together, format them, and set up interactive hyperlinks.

Creating a document such as the one shown in Figure 5-1 requires the spatial organization of text and media elements, formatting them appropriately so that they appear as they intended to be viewed. The interactivity elements that are shown as hypertext links need to be defined to point to the appropriate locations, either within the same document, or to other documents. The document also needs to be published in a format that can be viewed in a standard manner using an HTML browser such as Microsoft Internet Explorer or in the Adobe PDF reader. Although the text can be written in a variety of editor programs, the formatting, typesetting, and hyperlink setup can become more complex, requiring time on the part of the author. Importing other media

elements, such as images also has its own issues such as support of various image formats, for example gif and jpg. Also, the publishing aspect requires knowledge of the syntax and setup of the Hypertext Markup Language (HTML) or the PDF binary format. Such authoring tasks are greatly helped by software authoring tools, where the author can be agnostic of all these media support, HTML formatting, and other publishing-related details. Commonly used authoring tools to create such documents include Macromedia Dreamweaver and Microsoft Word, FrontPage, and even PowerPoint.

Another example shown in Figure 5-2 is a photo album such as the one adopted by Yahoo! or Picasa by Google. Here, a slide show of photographs is assembled to allow users to browse photographs in various categories. Each photograph is hyperlinked to more information that relates to the photograph or a particular news topic. For example clicking on the photograph may open a higher resolution photograph, a webpage or may even display a video. Such applications allow users to "visually" parse information before deciding to invest more time into reading a detailed report pertaining to the topic. Although generating content such as this is no different conceptually from the preceding example, it does differ when the photographs to be displayed need to change dynamically depending on the current incoming news, and their hyperlinks

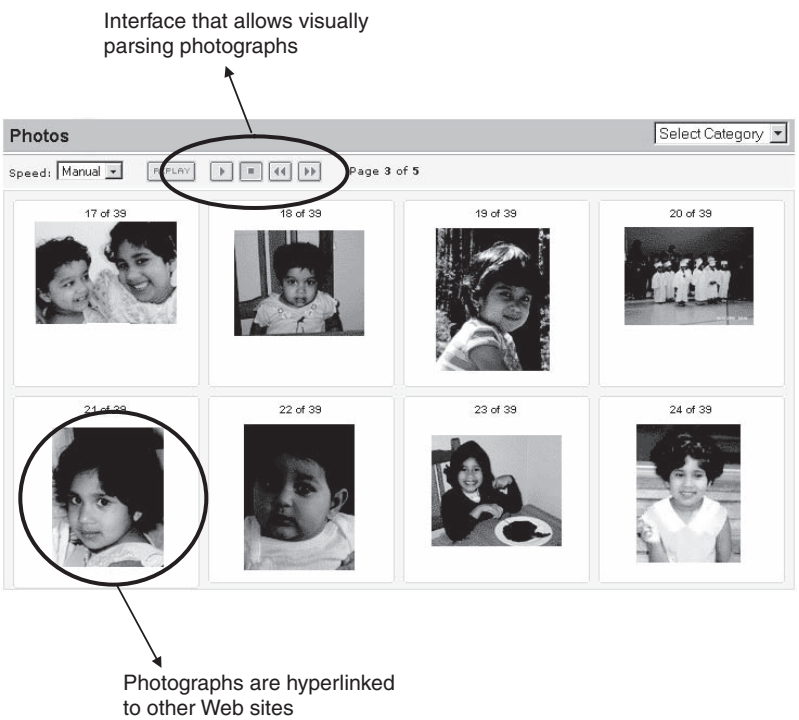


Figure 5-2 Photo album: Here, the user can manually or automatically browse through pictorial information that is hyperlinked to more topical information. Depending on "interesting" choices, the user might choose to nonlinearly interact and view information.

need to be dynamically updated. Quantitative metric algorithms such as "most viewed images" or "most distributed or e-mailed images" and their corresponding hyperlinks are used to collate information automatically.

The next example discussed is a multimedia application, which in addition to spatial organization needs temporal depiction and an understanding of the issues that relate to compression. Figure 5-3 illustrates a video-on-demand application that was created by MPEG-4 authoring tools designed by iVAST and Envivio. The images depicted are screen shots from an MPEG-4 player while a presentation is in progress. The first image in Figure 5-3 shows the start screen, which is a video lounge in the drama category with movie titles having hyperlinked images. The difference here is that the images are hyperlinked to other MPEG-4 files. You can navigate to the other categories such as the horror

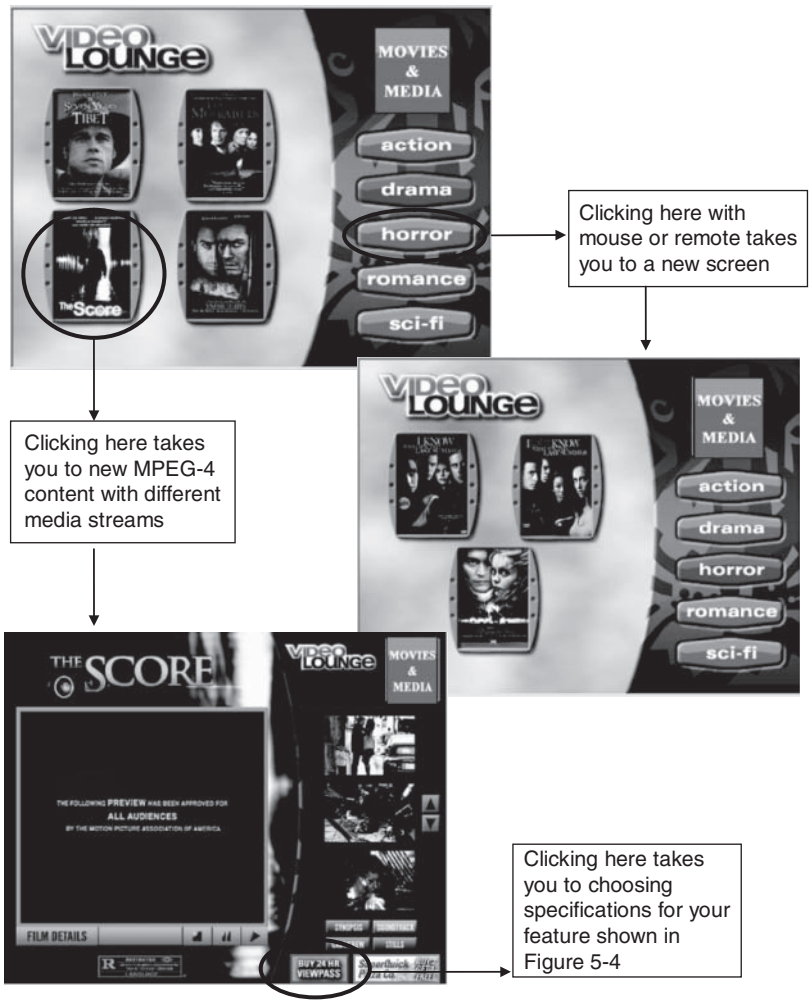


Figure 5-3 A video-on-demand application authored and published in MPEG-4. The highlighted circles illustrate the flow when that media item is clicked.

category, which is shown in the second image in Figure 5-3. By clicking on a movie icon, such as "The Score" with a mouse if on a PC, or with a remote if on a set-top box, we get another MPEG-4 scene shown in the third image. Here, there exists a variety of information that allows a user to experience what the movie is about. The user can do this by reading a text description of the movie, seeing image stills, listening to sound tracks, or seeing a brief preview. If the user chooses to watch this movie, selections need to be made from a number of options such as high/standard definitions, video format, and audio language, and the presentation is streamed to the user, assuming that there is enough bandwidth. An example of these choices is shown in Figure 5-4.

This example is unlike the others in the sense that it has been created and distributed entirely in the MPEG-4 platform, which is an ISO standard and will be explained in more detail later in Chapter 14. However, of interest here is the explicit choice given to the viewer, which has come about by way of authoring the presentation. The media streams are meant to be streamed from a media server to an MPEG-4 client, which could be on a PC, or embedded in a set-top box, or on a handheld PDA. It is important to understand that although the content authoring process here makes use of procedures common to the other examples, additionally the content has been authored to be device independent. The MPEG-4 authoring tool used in this example needs to provide the author with a choice of compression, quality, and bandwidth support along with various

Figure 5-4 Video-on-demand application in MPEG-4 (continued)—figure depicts a choice of encoding, video format, audio, and other options for purchase so on that a user can choose the appropriate viewing

MPEG-4 profiles that content can be authored to support. The profiling information is included in the content's bitstream and is necessary when the content is designed for multiple platforms. It allows MPEG-4 players to decide whether on not to play the content depending on the device's capabilities. For instance, the content in Figure 5-3 was authored for a broadcast bandwidth and will be supported by MPEG-4 players on set-top boxes and desktop PCs, but likely not supported by mobile cell phone devices where the player has limited capabilities because of bandwidth and device power constraints.

You can see by the three different examples that authoring tools are necessary to create the rich interactive media content. Designing such authoring tools is not trivial. The discussed examples and intended experiences should give you an idea of the scope and complexity needed to be supported by multimedia authoring tools and the authoring process. Most often, producing content involves the use of various authoring tools—some that deal specifically with video and images and others that combine them together. The next section analyzes the general requirements for a multimedia authoring environment, and investigates a number of authoring systems that are prevalently used. Some of these are proprietary and result in the author publishing multimedia documents that are in a proprietary format and, thus, need a proprietary viewer to view them, for example, Macromedia, Viewpoint, or Tribemedia. There are also others that are based on open standards and do not need dedicated and proprietary software for viewing, such as a DVD presentation or authoring tools based on the MPEG-4 format.

2 REQUIREMENTS FOR MULTIMEDIA AUTHORING TOOLS

You can derive a comparison between using multiple authoring tools to create multimedia content and word processors that are commonly used to create text documents. Both tasks involve collecting and compiling data and formatting it to produce coherent and organized information. When writing a book, or any text, the author's task is to put together words and sentences that convey the required idea or concept. The author also needs to maintain the flow of thought from paragraph to paragraph. This spatial organization and flow, which is sometimes not easy, can be compared with a multimedia author's organization of media and temporal flow in a production. A book's author is only concerned with text, and typically uses a text-based authoring tool or a word processor such as Microsoft Word, which provides the necessary tools to edit, typeset, and format the text. Correspondingly, a multimedia author needs to deal with all forms of media to create multimedia productions. Multimedia authoring tools should provide similar features to edit media and put different media elements together in a cohesive manner. In the case of media, these requirements become more comprehensive, not easy to generalize, and at times can be abstract. It is generally considered more difficult to make a harmonious multimedia production when compared with writing coherent text. In this section, we try to address such general-purpose, as well as specific, requirements for the multimedia authoring process.

The variety of multimedia applications and presentations used today has different modes/means of interaction, different delivery platforms, and is meant for different market segments and audiences. Consequently, it is hard to write one authoring program that will be able to create all applications. Normally, an authoring program

is designed to create content that can be published for a specific platform or multimedia format, for example DVD authoring tools or Web publishing tools. However, common tasks to all multimedia authoring processes can be enumerated as follows:

- Creating, editing, and making the individual media items that make up the presentation production ready
- Assembling the items into a coherent presentation, including the specification of the temporal and spatial layout of the media elements
- Specifying the interaction between the media elements, which often amounts to also delineating the flow of content as a viewer interacts with the presentation

Authoring methodologies do require dealing with media elements both individually and collectively. The first point in the preceding list deals with getting an individual media item ready for usage in a production. In most cases, the individual media items, such as digital images, video, and audio, cannot be used in their captured form and need to be edited and formatted individually within dedicated software meant for one media type. The processes that refine, transform, edit, and change individual media types deal with the *intramedia aspects* of authoring. These are explained in Section 3. The next two points in the preceding list refer to the *intermedia aspects* of authoring, where the authoring tool assembles the different media elements together. This requires organization spatially as well as temporally and normally makes use of well-established paradigms, which are discussed in Section 4.

3 INTRAMEDIA PROCESSING

The intramedia issues deal with processing an individual media type so that it can be made production ready for authoring. Intramedia processing depends on the media type and typically makes use of dedicated software related to that media. This is a necessary step because most often the captured media, such as images and videos, need to be edited or formatted. For example, video captured often has undesirable parts that need to be edited out, parts where color and contrast need to be enhanced. Such specific video-related aspects are performed in a dedicated video-editing tool such as Adobe Premiere or Apple Final Cut Pro. Digital images captured via a digital camera might need to be cropped, resized, sharpened, filtered, and saved in other formats for use in a multimedia production. Adobe Photoshop is the software of choice for editing images. Similarly, Autodesk Maya or 3ds Max is used to create 2D/3D graphics and animation content. Adobe Audition is a good software tool used to edit and clean audio data. All these examples show the need for captured media data to be transformed before it can be used in a presentation as is, or combined with other media elements. In general, a variety of commercial media software tools are used for the editing of individual media types. There are also others that are open source projects and available as freeware. The following sections discuss common issues related to the processing of the different media types in these software tools.

3.1 Intramedia Issues Related to Images

Images can be captured by digital cameras or scanned into digital form by a scanner. Although the image can be used as is, it is not uncommon to transform and edit the image per requirements of a multimedia presentation or publishing. Images can also be created in software and filled with colors or shapes or composited with digitally captured images. These tasks are often needed for a variety of applications, such as Web page publishing, advertising, art viewing, and so on and make use of commercially and proprietary software such as Adobe Photoshop, Paint Shop Pro, and so on. Moreover, complex multimedia applications also make use of images such as a photo album viewer, compositing graphic images in video for effectively displaying information, for advertising, and even for copyright protection. The transformation used on images might alter the image properties, such as width, height, and color channels. Images are often cropped to capture only the relevant part of an image. They can be further scaled and resized to get them into an appropriate format. Image operations also include filtering the image to enhance the content, using, for instance, an edge sharpening filter, or compositing. Images are also painted on and retouched to change colors or add text. The example in Figure 5-5 shows a combination of various techniques used to composite two images. The top two images were composited to create the final image at the bottom. However, the process involved four sub-procedures performed in Photoshop.

- *Repainting areas in both images*—This was done to remove the subject on the right in the first image and the background in the monkey image.
- *Creating alpha channels for compositing*—An alpha channel was created for the second image that consisted only of the monkey (foreground object).
- *Adjusting color channels for color to match better*—The monkey's original color distribution does not match the color distribution of the first image mainly because one is an indoor image and the other is an outdoor image. The RGB color channels were manually adjusted for better color matching.
- *Compositing*—The two images were finally composited using the alpha channel of the second image.

You can see that the process is not so involved when using authoring tools such as Photoshop, which is the application used here. However, the process does not scale up when extending the same effect to a video. It is cumbersome to modify every frame in a video sequence and manually composite it. This and other issues with video are discussed next.

3.2 Intramedia Issues Related to Video

Video provides engaging informational and entertaining content. Video images are traditionally captured by an analog/digital camcorder or created using a variety of software animation tools. Video captured by a camcorder often needs to be edited appropriately to fit the needs of production. This is because you might have recorded content in which sections have poor filming and image quality, or sometimes mistakes have

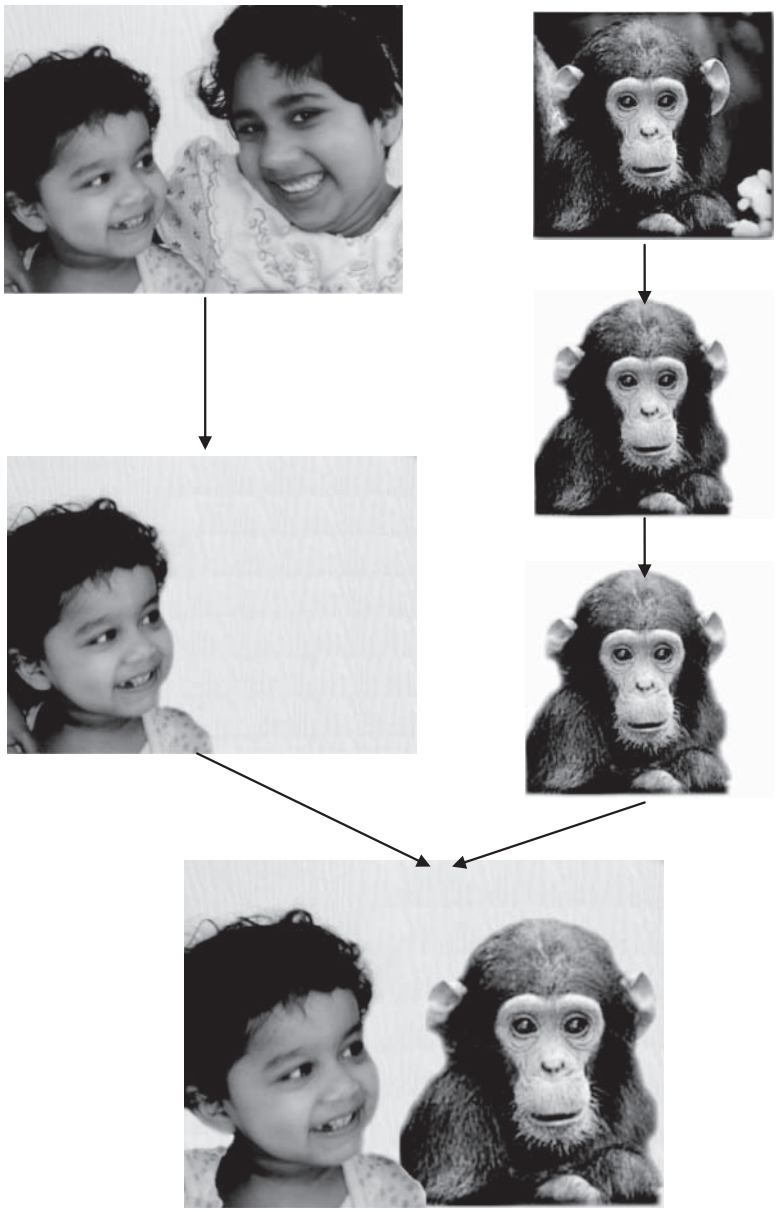


Figure 5-5 Image compositing example. The images depict the sequence of processes involved. The upper-left image is retouched to remove the person on the right side. Correspondingly, the right image is retouched to get the foreground element, whose color channels are adjusted to visually match the color distribution in the first image. Finally, both the altered images are composited to produce the bottom image. See the color insert in this textbook for a full-color version of this image.

occurred while recording. Newer models of digital video camcorders now have onboard video-editing features to help you enhance the look of your recorded content. However, these special editing features are fairly basic and have to be added while you are filming your content, which makes the whole in-device editing process rather cumbersome. Normally, software video-editing tools are used to edit video. A number of commercial video-editing software tools are available. The high-end tools include Adobe Premiere, Apple Final Cut Pro, Avid Media Composer, and Pinnacle Liquid (now part of Avid). Low-end, consumer-level editing tools include Windows Movie Maker and Apple QuickTime. Some standard operations performed during editing are as follows:

- Changing the video properties, such as width, height, interlacing properties, and even the number of frames per second (also known as video retiming). Standards-based changes frequently require a frame rate change, for example going from NTSC to PAL.
- Cutting, editing out, and combining parts of video into a tight, cohesive single video with smooth or abrupt transitions between sections. Sections of the same or different video are combined together here, thus making use of only the well-recorded and important parts of the recording.
- Creating titles that can scroll on and off screen.
- Creating transitions by fading or using dissolves or wipes.
- Using filters to adjust exposure or color-filming errors, such as over- or underexposure or to make minor color correction.
- Using video overlays to superimpose a graphic over the video. This graphic might be a company logo or an advertisement.
- Synchronizing audio to video—strictly speaking, this functionality goes beyond the intramedia video issue, but is often provided by most video-editing tools to provide background music or to ensure lip synchronization.
- Compressing video for a required bandwidth.

Digital video editing has now exploded with useful editing tools that provide for technical and creative compiling. Most video-editing applications have a built-in capture feature that can access footage on a digital video camera and download it to hard disk memory for editing. Different video editors have different capabilities and workflows, but most make use of the "timeline" to deal with the temporally changing data. An example of timeline-based editing is shown in Figure 5-6. Here, the original video is cut into clips, filtered, formatted, and, finally, aligned on a timeline. The timeline shows a temporal flow of data, which might consist of individual clips joined together with any transitions inserted between them. The output of the authoring program renders a new video file, which contains the result of transitions, filtering effects, overlays, and so on. Depending on the complexity, this can require large amounts of computing power and time. The exporting process might also need to be in compressed formats, such as MPEG, compressed AVI, and requires knowledge of what compression parameters are and how to control them for required bandwidths. A sample video-editing session from Adobe Premiere is shown in Figure 5-6. Here, a few clips of video have been joined together with transitions. At the same time, a background music track plays and a set of titles comes toward the end.

Figure 5-6 Example of a video-editing session in Adobe Premiere. Various video clips have been put together with transitions. There is a background music track and, finally, toward the end a title track consists of images playing for constant small durations.

3.3 Intramedia Issues Related to Audio

Similar to video, capturing digital audio can require significant post processing. Normally, an audio track is captured with the video when using a digital camcorder. In this case, the audio and video are synchronized, but the audio recorded might not have a high quality because of the actors' or subjects' distance from the microphone. For high-quality productions, audio is recorded separately by stage hanging mikes or small mikes attached to the actor. Audio-video synchronization needs to be done in such cases.

Most audio-processing issues relate to cleanup and editing. For example, digital signal processing filters are always applied for noise reduction or to produce smaller, down sampled and compressed versions. Editing techniques are also required to remove unwanted audio, create audio fade-ins and so on. For example, a TV commercial, introduction, or a closing audio sequence might need to be played and blended in with increasing amplitude while the original audio signal is still playing. An audio-editing tool is shown in Figure 5-7 where sound data is organized as samples. Sections of the data can be selectively edited according to the effect desired.

3.4 Intramedia Issues Related to 2D/3D Graphics

Animations are now commonplace in multimedia today. Examples include simple 2D animations to showcase information in a presentation, advertisements, or more engaging content, such as full-feature movie animations and video games. The authoring and production process has become a good blend of art and science. The scope pertaining to authoring here is very large, especially in the realm of 3D graphics. Here, we discuss authoring processes commonly used to create 2D animations.

After the creation of static images on a computer, animations are created to generate a perception of motion. This is achieved by creating a sequence of images at which

Figure 5-7 An audio-editing tool—In Depth. Shown are highlighted sections of stereo audio samples in the sound data file. These samples can be edited out, filtered, attenuated, and so on depending on the desired effect.

the object(s) are continuously changing their properties such as spatial location, color shading, or deformation. When these images are played successively at frame rate, as in a video, you get the effect of animation. The first kind of animation was created by drawing images on celluloid—called cell animation. Here, the different hand-drawn celluloid images when moved in succession created an animation. Now with computers, software tools are used to create images of objects in motion.

Graphical animations are categorized mainly into two different types—raster animations and vector animations. The ultimate display format is the same—a sequence of images that are shown at a frame rate. But the file formats for raster and vector graphics are remarkably different. Rasters consist of images that simply display in the multimedia player as a video, whereas vector animations need a *rendering engine* that takes the vector format and converts it to a raster on the fly and does the animation. For example, if you built a 3D model of a toy in Alias Maya or AutoDesk 3D Studio Max, the model is represented as a vector format consisting of points, lines, and polygons. The model can be moved around in 3D space and looked at from different angles, which creates a controlled animation. Once the author is happy with the animation sequence generated, it can then be exported from the 3D authoring tool to other movie formats, such as AVI, a sequence of JPEGs, or even to Flash format. This export operation turns the 3D vector description at each frame into a rendered 2D image. The sequence of images can then be viewed in an AVI player or a Flash Player, where the sequential rendering of the image frames simulates 3D movement.

Many authoring tools today allow you to create animations in a specific format, and many multimedia players allow you to view the animations created. Common file formats used for animations are SWF (Shockwave Flash), SVG (Scalable Vector Graphics), and PNT (USAnimation). These formats are supported by a variety of players, which can be stand-alone or embedded in a Web browser—like Flash, Swift, or Viewpoint—or incorporated into a standard player, such as DVD, MPEG-4, and so on. Most players support both vector and raster animations.

4 INTERMEDIA PROCESSING

Once the individual media types are available and production ready, they are brought into a multimedia-authoring tool. Here, all the individual media elements are assembled together to form a production. All authoring tools and processes ultimately publish the content in a specific format for viewing in a specific type of multimedia player. These formats and players might be based on proprietary technology or collective industry-determined standards. Proprietary formats and players include Adobe Flash Player, Windows Media Player, Viewpoint Experience Technology, or even interactive 2D/3D game engines, such as Quake, Nintendo platforms, and SONY PlayStations. Standards-based multimedia formats and players include DVD players and MPEG-4 players. The authoring tools that support exporting to these formats include commercial tools, such as Macromedia Director, DVD Studio Pro, Tribework iShell, iVAST iAuthor and iEncode, and so on.

It is impractical to create a single authoring tool that will universally fulfill the needs of all the multimedia content and applications for various platforms used in different markets, for example, Web-based advertising, entertainment, gaming, education, enterprise. Every authoring tool is designed to serve a specific market and platform, and, hence, has specialized workflows and capabilities. However, some common requirements of intermedia authoring tools are needed to give the author a freedom of expressiveness, such as the following:

- Spatial placement control
- Temporal control
- Interactivity setup

Depending upon a variety of controls given for each of these, you could arrive at specific properties and workflows exposed by an authoring tool. In Figure 5-8, we suggest a categorization of commonly available authoring tools based on these functionalities. An *authoring volume* is shown in the form of a cube using these three properties. Three color axes, RGB, show the three properties, respectively. The color in the representation is meant to better depict the volume areas.

An ideal authoring tool would allow good control over specifying spatial, temporal, and interactivity among the media elements and, hence, resides on the farthest corner of the cube. The power of an authoring tool goes up as it gets closer to the ideal authoring tool. Intramedia-based authoring tools, such as Adobe Photoshop and Premiere, have their respective positions at the spatial and temporal ends of the volume. Most word processors end up in the spatial corner if they do not provide for any interactivity

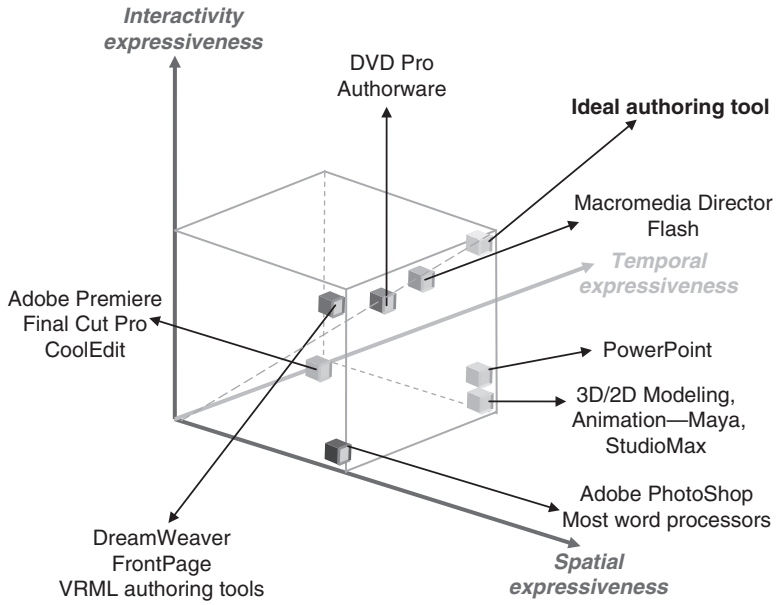


Figure 5-8 The authoring volume: Categorization of authoring tools based on spatial, temporal, and interactivity expressiveness. Commonly used authoring tools are placed in this representation as shown when compared with an ideal authoring tool. See the color insert in this textbook for a full-color version of this image.

setup. However, tools that allow for spatial as well as interactivity setup, such as Dreamweaver, and Microsoft FrontPage reside in the spatiointeractive corner (shown as magenta). Tools that allow temporal as well as spatial setups without interactivity publishing, such as graphics animation tools like Illustrator, are shown at the spatiotemporal corner of the volume. High-end authoring tools that do provide more complete capabilities, such as Macromedia Director, are shown closer to the ideal authoring tool.

4.1 Spatial Placement Control

This requirement stems from a document model, and involves providing the ability to place the imported media types at various spatial locations. These media types can be video, audio, images, virtual animations, and even programs or applets. The media objects differ in nature, and their content can be delivered either instantaneously or continuously. However, they need to have defined positions on a screen, where they will appear while a viewer is experiencing the multimedia presentation. A good example to illustrate this is the workflow used in Macromedia Dreamweaver or Microsoft FrontPage, used to publish HTML content. Here, you can import image elements and place them at various locations. Text can be inserted at required places and, finally, the document can be published in an HTML format to be viewed by any HTML browser. Apart from images, media files, which contain video and audio, need to be

imported and positioned at specific locations where the video will play. The authoring tool needs to provide an easy way to move and place media objects.

One generic purpose where good placement control is needed is in specifying interactivity interfaces, which can also be user interfaces. These interfaces are normally set up as images of buttons, which have to be placed in positions that make the multimedia experience viable. For example, multimedia applications showing a video have controls placed at the bottom or the side. Additionally, the spatial layout can involve just one screen or multiple screens. Sometimes, predefined layout templates are used. Templates are useful for real-time authoring environments.

In most formats, the spatial relationship between media elements and their placements are often explicitly or implicitly captured in the form of a scene layout graph. A scene graph typically captures a lot more than the layout information, such as dependencies between the media types, and by itself might not even be exposed by the authoring tool.

4.2 Temporal Control

Just as multimedia elements have a spatial layout in any presentation, multimedia documents incorporate time; hence, any multimedia authoring tool should provide a manner to temporally lay out and control, or schedule, media elements in a presentation. Media types, such as sound, video, and graphical animations, are dynamic by definition and change over time. Any authoring tool that involves such media types should provide metaphors for temporal editing and control. Additionally, when considering intermedia issues, for example, putting together video and graphical animation that are both changing temporally, the temporal controls should provide more flexible ways to express media interaction temporally (just as they do spatially). For example, while watching a DVD, you normally do not get to the main movie presentation directly. Prior to getting to the main DVD menu, the content is designed to show previews of other movies, advertisements, and so on. Another example might be that while watching a presentation, you often see a spatially placed image, which is an advertisement. This image might be changing at a fixed temporal rate and cycling between different images or advertisements, whose rate is different from the presentation rate.

Temporal control in an authoring tool is required to coordinate the multimedia presentation of media objects with one another in time. Static media items such as text and images, unlike video or audio, have no inherent notion of time, but the author might want to impose a temporal element, which dictates when they appear in a presentation and/or how they change. This temporal schedule can be set up on a global level, where the entire presentation is meant for a passive experience such as in a movie, or on a more micro level, where individual segments of the presentation have a temporal schedule. The individual segments could then get triggered depending on the interactive elements that are set up. The authoring tool, thus, needs to provide a methodology to temporally lay out and trigger media elements. Abstractly, a temporal layout is a sequence of entries, each of which contains a time and an event. The temporal layout is used to control a document's presentation.

During a presentation, the resulting temporal trace of events might differ from the intended temporal layout. This is dependent on the platform that the presentation is running and the mode of delivery. Often, media delays occur for a variety of reasons, including queuing for resources or bandwidth limitations and variations.

4.3 Interactivity Setup

The spatial and temporal layouts in multimedia documents and presentations have a linear flow and, as such, are paradigms of passive experience. As the usage of multimedia documents becomes more and more common, these documents are starting to contain increasing application-specific functionality. The application-specific functionality of multimedia documents is characterized by the way users interact with presentations and how the presentations are designed to react to the interaction. Let us consider an example of a computer-based test in some domain of expertise where the user is shown questions and pertaining media elements (images, video, animations) to that question. Here, the test itself is a multimedia presentation that has been authored for a specific set of people. The presentation of the next question and media elements depends on whether the user's answer was right or wrong. Another example is interactive television. An interactive TV program can be looked at as a multimedia presentation when it is enriched with additional features, such as more information related to the currently viewed program, which the user could access. This might include information about the actors or a hyperlink to shopping Web sites to buy some of the props or apparel being used in the content.

Setting up these types of interactivity involves complex event handling that results in a variety of actions. An event can be a mouse click or keyboard press by the user on a media item or media item part, such as part of an image or a video frame. An action needs to be executed on that event. The action might be starting, stopping, pausing, or entirely changing the media item being displayed, for example pausing a video or changing a channel. It might be executing a hyperlink where the presentation might jump to a Web site, another part of the same media item, or another media item. For example, while browsing the Web, clicking on an image might show you a high resolution of the image or take you to another Web page. In the DVD experience, a "chapter forward" executes a hyperlink into the same video media at a different time.

Thus, you can see that to set up an interactivity element, you need to know what event to look for and what action to execute. The combinatorial complexity of specifying a choice of all desired actions is not amenable. Consequently, authoring tools have to make a choice of the actions they want to execute. Figure 5-9 shows a snapshot of DVD-Lab Pro used to author DVD movies. The flowchart shown depicts the interactivity setup for the DVD navigation.

5 MULTIMEDIA AUTHORING PARADIGMS AND USER INTERFACES

An authoring paradigm or an authoring metaphor can be referred to as the methodology by which an authoring tool allows an author to accomplish creation of content. From a multimedia perspective, an authoring tool does provide the use of one or a few paradigms that are used to accomplish the media authoring tasks. Although there are many authoring paradigms, some of the commonly used ones such as the timeline metaphor, the scripting metaphor, and so on are described next.

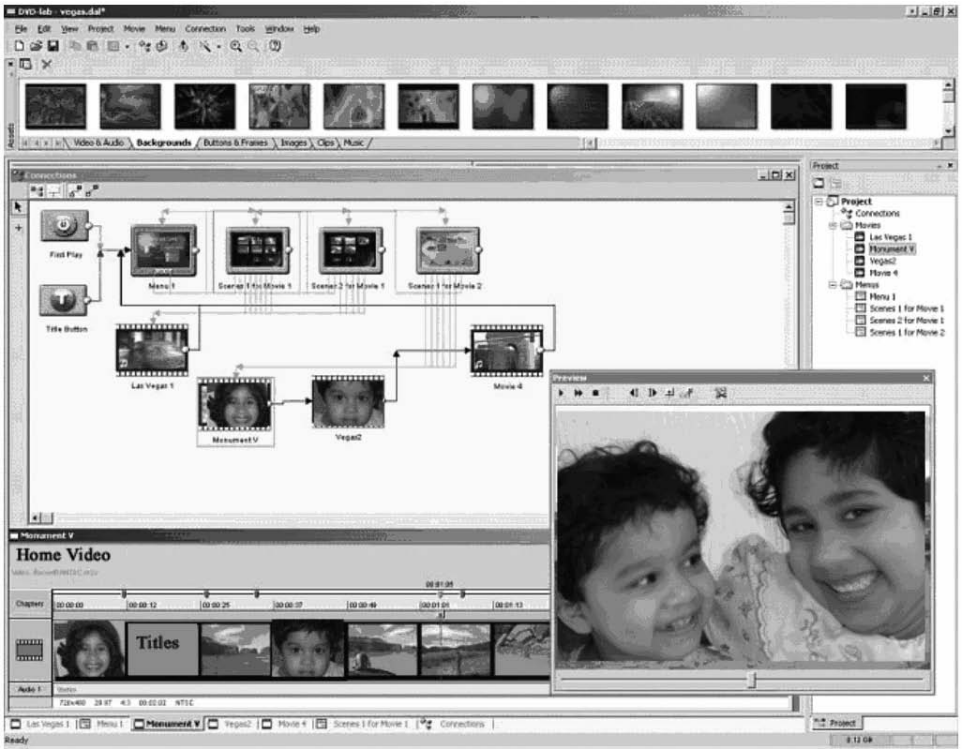


Figure 5-9 A snapshot of authoring a DVD using DVD-Lab Pro. The graphical layout shows an instance of the flowchart in the DVD interactive navigation.

5.1 Timeline

This is the paradigm used by Macromedia Director for Flash content, and also by Adobe Premiere and Adobe After Effects, where you are trying to associate media properties temporally by setting them on a timeline. In Adobe Premiere, this paradigm is easy to visualize because you are in control of when the video frames of video 1 appear, when the video frames of video 2 appear, and how video segments transition in a temporal manner. Adobe Premiere deals with audio in a similar way, either in conjunction with the accompanying video or separately from the video. Along with video and audio, a multimedia presentation might need the temporal control of graphical objects and animations. An example of this is illustrated in Figure 5-10. Macromedia Director provides mechanisms to take all these media types and have a temporal layout. Additionally, you can set other properties of media elements, such as spatial coordinates, color, and so on, that change with time. This is helpful to produce animations using media entities, such as images, graphics, video, and so on.

In Macromedia Director, as the name suggests, the author works in a way similar to directing a movie set or a theater production. The author is the director and controls "props," which are media elements. The props can be brought onto the stage and placed at different locations or at different points of time (controlled by a timeline).

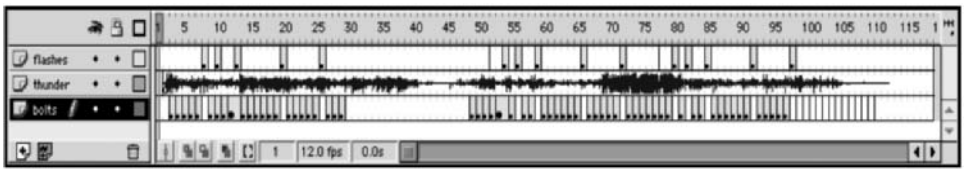


Figure 5-10 An example of a timeline where media elements are organized temporally. Shown in the figure are three media types temporally laid out in the creation of a thunder-and-lightning scene. The first line shows animated flashes occurring at specific time instances that last for one second. The second line shows a simultaneous audio track that plays the thunder sound. The last line describes the video image of bolts.

Macromedia also provides simple metaphors to manage your media elements on its production stage, which is always open and is behind other windows. These metaphors include the following:

- The *cast* window—This window stores the cast that will take part in the show. As such, it contains all the media elements, which are video, audio, image files, and graphical objects used for your production. These elements can be imported from outside file formats or represented in Director's native vector and bitmap formats.
- The *score* window—This is where you would compose the behavior of your media elements. It is here that the author sets up behavior temporally. Additionally, media elements can be combined into different channels that have similar temporal properties. The score window allows fine controls for timing media elements, transitions, and animations. You can also give media elements a prebuilt behavior, such as fading, roll over buttons, zoom, and so on.
- The *script* window—This is where you can set up interactivity and insert links.

Additional temporal control is also given from the control panel that allows you to play, stop, and jump to a specific temporal frame in your presentation. This can be used for viewing or additional fine-controlled temporal editing.

5.2 Scripting

The scripting paradigm works just like traditional computer programming, which involves the execution of commands in a sequential manner. Thus, it requires the understanding of programming concepts and sequential execution and, hence, is not commonly used by production artists. But, at the same time, it can be a very powerful metaphor that allows the author a freedom of expression that is not provided by the exposed user interfaces in any authoring tool. The paradigm normally involves the use of a scripting language to specify the spatial and temporal properties of multimedia elements, sequencing, hot spots, setting up synchronization, and interactivity.

Just like any software programming task that involves programming and scripting, the overall authoring process to create content in this case does take longer than using graphical user interfaces. But because scripting allows the author to script actions, events, transitions, and so on, which are generally not provided by the user interfaces, more powerful effects and interactivity are generally possible, depending

on the level of control exposed by the scripts interfaces. Most good authoring tools provide a scripting interface to give more power to authors if necessary. Examples of scripting languages specific to proprietary authoring environments include the Lingo scripting language of Macromedia Director and Assymetrix OpenScript for ToolBook. There is MEL scripting used in Autodesk Maya for creating 3D graphical content. Today, a variety of powerful open standards-based scripting languages are used to program multimedia setups, for example, JavaScript, HTML, SMIL, VRML, and XMT. JavaScript and HTML need no formal introduction; however, it is worth talking about the other newer programming/scripting standards.

SMIL (pronounced ''smile'') stands for Synchronized Multimedia Integration Language and is a simple but powerful markup language for timing and controlling streaming media on the World Wide Web. SMIL has very high-level constructs, unlike HTML or JavaScript, and is used for rich multimedia presentations that integrate streaming video, audio with images, and text. Figure 5-11 describes a simple SMIL script that specifies two adjacent laid-out regions, displaying an image in one region for the first 10 seconds and a video in the other for the next 10 seconds. Similar language constructs are used in VRML (Virtual Reality Modeling Language) to display and interact with 3D data, which has been extended in an XML format in XMT (Extensible MPEG-4 Textual format) to specify scene elements and interaction among them.

```
<smil>
   <head>
      <layout>
         <region id="Region1" top="5" left="20" />
         <region id="Region2" top="100" left="20" />
      </layout>
   </head>
   <body>
      <image region="Region1"
         src="http://www.mysite.org/logo.jpg"
         dur="10s" />
      <video region="Region2"
         src="http://www.mysite.org/myvideo.avi"
         clipBegin="10"  clipEnd="20">
      </video>
   </body>
</smil>
```

Figure 5-11 Example of a SMIL script that specifies a layout and displays video and image media elements

5.3 Flow Control

In this paradigm, content creation and sequence flow are achieved much like a flowchart. The author or user drags preprogrammed icons and organizes them into a flow line from a palette. Parameters are set for each icon typically using dialog boxes. An icon here represents a media element, each having properties such as spatial location, duration, and so forth. This paradigm offers very rapid prototyping setups to generate sophisticated content having sequential flow. However, the functionality tends to be limited by the number of icons provided. This paradigm is commonly used in Macromedia Authorware and a sample authoring session is shown in Figure 5-12. In the figure, a presentation about a school is being set up.

5.4 Cards

Card-based workflows are useful for spatial organization of media elements on a screen and the changes in your scene during a presentation. Commonly used tools that support the card paradigm are Apple's HyperCard[1] and the SuperCard by Solutions Etcetera. Cards represent discrete screen displays, onto which buttons and graphics are added. Jumping to a new card in the stack refreshes the display, erasing previous elements and functions implicitly. Simple navigation is similar to a slide show. More sophisticated interaction is added through a scripting interface.

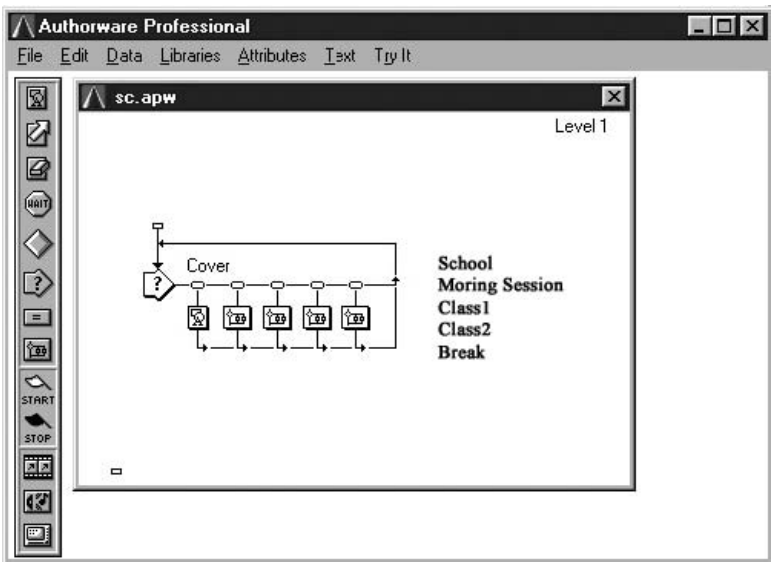


Figure 5-12 Example of the Macromedia Authorware environment to depict the flow-control paradigm

[1] The HyperCard has now been discontinued by Apple

6 ROLE OF USER INTERFACES

To work with interactive content, the user has to be able to access and control how to interact with it. User interfaces integrate the processing and consumption of heterogeneous media objects like text, graphics, video, and audio in a manner that enhances the user experience. We have talked about various metaphors used to author multimedia content, including spatial metaphors and temporal metaphors. An essential necessity of interfaces for multimedia information is, then, to support these space-variant and time-variant metaphors.

As multimedia presentations get complex, the role that hardware/software user interfaces play starts to get important. For example, watching a single-channel video broadcast, where the user sees only one video/audio channel, does not need complex UI design. The functionality that a user might desire is play, pause, stop, and perhaps forward and rewind, all of which can be accomplished with a few buttons. However, as the presentation gets more complex with multiple channels and the user exercising a choice to browse through channels, the interface design can get challenging. Possible interfaces might be choosing a channel one at a time until you get what you like (a standard implementation today) or seeing all channels displayed as thumbnail videos, which can further be categorized, while at the same time seeing your current viewing channel

The importance of user interfaces to access and interact with multimedia data should be obvious. There are various kinds of interfaces, for instance command-line interfaces and text user interfaces, which are text based. A user has to type in a text string to interact with the system or content. Because multimedia normally contains visual media types, any interfaces that are used to interact with multimedia presentations will inherently have to be visual or graphical in nature. Generic visual design principles that related to large desktops have been well studied in literature. These principles are designed to place users in control by using I/O devices such as a keyboard or mouse, providing informative iconic images to navigate, using judicious modal dialogue, maintaining a consistent motif of interaction, (similar) visual aesthetics, and so on, and all the while giving immediate feedback. It is not our intent to describe general user interface principles that you might already know or can get informed by reading relevant articles on user interfaces. Rather, in this section, we discuss user interfaces to enhance user experience when consuming and interacting with complex interactive media presentations involving a large number of objects or when the content has to be consumed on devices where the spatial capabilities are seriously limited, for example, a cell phone when compared with a desktop.

6.1 User Interfaces on Mobile Devices

The user experience that interfaces have to provide increases even further as the devices on which multimedia information is consumed get smaller and smaller. Small portable devices such as cell phones, iPods, PDAs, and so on have been very popular and have become practical for some time now. But, in recent times, the increase in their processor speed, onboard memory, bandwidth of communication, along with the availability of features such as GPS (Global Positioning System) have turned them into powerful mobile units with an expectation that you should be able to use them in ways more akin

to the desktop. One thing, though, that has not changed is the small size of the device. The small device size introduces area limitations on how user interfaces are designed, displayed for interaction, and used. Devices such as Apple's iPhone, Research in Motion's BlackBerry, and Google's G1 phone all have smaller screens but aim to preserve the same complexity of media interactivity and consumption that you would get from their larger counterparts, such as desktops and televisions with set-top boxes or even game consoles. For example, when browsing the Internet on your desktop, the interfaces/metaphors used in your Web browser are well suited for informative exchange on the desktop. However, when the same content is to be consumed in a similar way on your handheld PDA, the metaphors available are strikingly different and a user interface design needs considerable thought to bring the same experience and satisfaction.

A commonality in the evolution of mobile devices that have area limitations has been the use of touch screen technology. The touch screen is one of the easiest to use and most intuitive of all interfaces and is becoming a popular interface with its affordability. Small devices have less real estate for traditional interfaces like the mouse and keyboard. In such cases, the screen itself, a touch screen, serves as the user interface where the user touches the screen and the contact location is conveyed to the application. Three basic systems are used to build touch screen interfaces:

- *Resistive*—In this case, the normal glass panel on top of a regular CRT or LCD monitor is covered with two layers, one conductive and the other metallic with an electrical running through them. When a user touches the screen, the two layers make contact, which changes the electrical field around the contact point. This is sensed by the device to compute the coordinate where contact occurred.
- *Capacitive*—Here, layers that store a small electric charge (functioning like a small capacitor) is placed on the glass panel of the monitor. With a user's touch, some of the charge is discharged onto the user. The change in charge is measured by sensors to interpolate a coordinate.
- *Acoustic*—As the name suggests, acoustic waves are communicated through layers on the glass screen in the x and y directions. Transducers are placed to measure the incoming and outgoing waves. In case of a touch, the change in the acoustic amplitudes is detected by the transducers, which can then interpret where the touch occurred.

Most touch screens could measure only one touch, but recent advancement in both the sensitivity and ability to do multiple touches have allowed touch screens to act as powerful user interfaces.

6.2 Multiple Devices as User Interfaces

Future computing paradigms are likely to involve a combination of portable and personalized devices. Multiple coordinated computing devices will be used together in various configurations, from different geographical locations, perhaps over different times. For instance, you might use your cell phone or PDA to communicate and browse information during the day, but when you come home at the end of the day, you want to turn it into a remote for controlling your home entertainment system.

Taken a few steps further, you might want your video content that is stored on your home digital video recorder to be streamed to you on your cell phone at some later time to view, where you start using your cell phone as a television. All such media-related applications on distributed devices offer new design challenges to the field of human-computer interaction. Although this area is yet to be explored, some guidelines are as follows:

- Allow for the distribution of information across the appropriate devices. Information devices have different strengths. For example, a television is appropriate for pictures, video, and audio output. Cell phones might be more appropriate for display of text and some graphics. Cameras are appropriate for taking pictures. An application that includes all three might be where a user wants to take a photograph and send it home to be viewed on the home television while people at home watch. The user might also send a text message that is also seen or heard at the same time. To make such applications achieve their purpose, the devices must be able to easily share their media information, either using a wireless or wired connection.
- On the same token, the information-shared state must be synchronized and congruent. Because multiple devices are dealing with the same task, all involved devices should have the same state of shared information. In the preceding example, you might be talking about a photograph you took that is still yet to be displayed on the home television, leading to the potential of confusing the consumers.
- The devices should be combined in ways such that collectively they are more powerful. The only reason why a user would desire combining devices together is that the applications on two (or more) devices cooperate and enhance the task that needs to be achieved. An example to illustrate this might be a real estate application, where a map of an area on a cell phone and pictures of houses on a desktop are individually quite useful, but if the map on the cell phone can be used to visually see the location of the house and also move around virtually in the house using photographs, all this while you are in the neighborhood driving in your car, it is probably a more worthwhile application.

7 DEVICE-INDEPENDENT CONTENT AUTHORING

In recent years, there has been a proliferation of different networks besides the Internet, such as cable networks, wireless networks, and satellite networks. Correspondingly, there is now a plethora of different types of devices, platforms, and access mechanisms or protocols that go beyond the conventional PC. These access mechanisms range from computers, to Web tablets, to televisions, to other appliances in the home, to mobile devices including cell phones and PDAs, to kiosks, and to access mechanisms and devices for the physically challenged. This expansion has also standardized the expectation among consumers about the availability and consumption of the same or similar types of content on the different platforms. The concept of designing content that can be viewed and interacted with similarly on different

devices and platforms is known as device-independent authoring. Device-independent authoring, thus, defines a "universal access" for content that is irrespective of networking access and infrastructure, hardware and software platforms, and can even be extended to include independence from geographical location, language, and culture. For example, you might want to access Web information while at home using a PC that connects to the Internet via a DSL line or a cable network. However, if away from home, you expect to access the same information using another device such as a cell phone or a PDA connected through a mobile phone network. Device-independent authoring is usually addressed by publishing content in a well-known and accepted standard that is supported by multiple devices. For example, HTML and HTML-based content is supported not only on a computer, but also on other devices such as cell phones, PDAs, WebTV, and so on. But authoring using a universally supported format is not sufficient by itself. Along with end devices being able to parse standards-based content, capabilities also need to be defined by the content, which tell the end device how to format and interact with it, which might change depending on viewing screen size, resolution, and devices available to interact.

Device-independent authoring also goes hand in hand with another device-agnostic authoring process—bandwidth scalability. A news producer might want to design content that involves image, text, video, and audio that is to be streamed at different bit rates on distribution networks. Each network supports a different maximal bandwidth. Compressing content and making a different file of each network easily produces an explosion in the number of assets used for different bandwidths and becomes an asset management nightmare. Ideally, you would want to create one content file, which is scalably designed and serves multiple devices on different bandwidths. Consequently, the multimedia authors have to develop content that is universal and can be delivered for viewing and interacting on the different devices supported on a variety of networks. Designing such *device-independent* or *universal* content requires more design analysis up front. For example, all the capabilities required of content might not be available on different devices, causing the content to not be viewed/interacted with properly. Additionally, bandwidth requirements impose budgeting bits intelligently both in the spatial, as well as temporal domain.

This is not trivial, and it is probably unrealistic to expect an authoring tool to create the same user experience for different delivery contexts and on different devices. However, when supporting multiple devices, a standards-based format that supports relevant profiles is generally chosen. Profiles can be visual profiles, audio profiles, graphics profiles, scene profiles, or even systems-level profiles. Profiles describe the complexity needed by the content in terms of the spatial size of screen, the bandwidth requirements, and the number of streams of images, video, and audio that can be simultaneously dealt with. So, when a device that is designed for profile X receives content of profile X or a profile simpler than X, it can decode and display the content. However, if it receives content that is authored for a higher profile Y, it might choose to disregard and not parse the bit streams, reporting the appropriate error to the user. This is often a better recourse than "choking" in the middle while the content is consumed. In the multimedia world, the ISO standards such as MPEG-2, MPEG-4, and WAP have made a big leap in attempting to harmonize device independence by making use of profiles. Example profiles in MPEG-4 include the *simple profile* and *advanced simple profile* for video and audio.

8 DISTRIBUTED AUTHORING AND VERSIONING

Many projects in various disciplines typically involve the services of a variety of people who have similar or complementary skills. All these people have to share, understand, modify, and update a common knowledge base, which can consist of physical and digital data. With multiple people working on the same set of data, it becomes imperative to have protocols and use processes that maintain and combine everyone's work into one product. A common example of such a process occurs in the software industry where software code is developed by multiple software engineers who might even be at different geographic locations and need to collaborate on the development of a software product. They need a system that maintains a central repository of code and allows them to independently check out code, work with the code and when done, to merge their extensions with the current state of the code in the repository. Systems such as Revision Control System (RCS), Concurrent Versioning System (CVS), and Subversion (SVN) allow software engineers to concurrently collaborate on software development. These systems merge code at the data level by comparing changes between the individual bytes that represent the code. Code conflicts occur when two people changed the same lines of code. The system tries to merge these based on comparing the individual data bytes; however, if it cannot do so, a conflict is signaled and the authors work toward a resolution. At the same time, versions of the software code are maintained along the way as people update the shared code base, so that a previous version can be arrived at with relative ease.

In the multimedia world, there are similar needs when multiple media data elements are worked on by different people concurrently. For example, let us say a group of people work together to make an instructional video. After the initial camera work, all the shots are loaded on a digital form in a media repository where they can be accessed by different people. The postproduction task consists of editing, combining different media clips, inserting titles, creating the necessary special effects, and so on. The goal is to create a final video according to a script that each one has. Although everyone's tasks might be well known, it is possible that while working concurrently, the person responsible for editing and creating one time segment of the video is the same time segment that another person is adding special effects to. Working on such sections simultaneously could cause conflicts. You could use the above-mentioned systems used in software code development, but these work at a data level without any temporal or other visual semantics. In the video example, the temporal aspects need to be resolved. However, for a normal multimedia presentation, there are also spatial and interactivity aspects that might need merging and conflict resolution. Writing tools similar to CVS used in code development are not easy when it comes to multimedia productions. Nevertheless, there are tools, applications, and emerging standards that are used to help collaborative multimedia creation. Microsoft NetMeeting has aspects of collaborative authoring where you can communicate images and graphics to other viewers and authors. You can also view online gaming as a collaborative authoring medium where multiple users are affecting the state of a 2D or 3D environment while interacting with it. A common paradigm used to concurrent authoring is a distributed client/server system implemented over a network that facilitates the collaborative manipulation and assembly of shared media elements. Such systems normally need to have the following:

- *High network responsiveness*—Media objects such as images are changed and shared in real time, putting a load on the network.
- *Maintenance of concurrent states across clients*—All clients see the same state of the production. The current state of the multimedia production is broadcast to every author. Everyone sees the same state. Any change that an author makes is broadcast to all authors.
- *Using consistency protocols*—These are necessary to ensure lack of conflicts. The previous point to maintain and control concurrent states requires the use of consistency protocols which maintain consistency among multiple instances of media objects. Commonly used protocols are locking and serialization of the media objects.
- *Real-time awareness*—Media elements used in a collaborative setup might include real time captured elements. For example, a number of authors in different locations are collaborating a live recording session to create live content that streams to users. In such cases, the authoring system needs to put a priority in communicating real-time elements.

Standards are also being set for collaborative authoring. WebDAV, or simply DAV, is a Web-based distributed authoring and versioning standard that defines HTTP extensions to make Web authoring an interoperable and collaborative medium. There are also other proprietary applications based on CAD and virtual graphic environments, which aim to have multiple authors collaborate at various levels from simple markups to directly manipulating content.

9 MULTIMEDIA SERVICES AND CONTENT MANAGEMENT

Authoring tools create content, which is delivered to consumers using a variety of media forms, such as DVD and networks such as cable, satellite, and DSL. People are now used to one or more forms of the following—touch screen kiosks, interactive digital television, digital radio, Web browsers on PCs, cell phones, and so forth. Along with the delivery of all the digital content, there is also a growing expectation of high-quality multimedia services constructed from multimedia content that is timely, relevant, accurate, personalized, and cheap. These services are used to browse, collate, search, choose, and even automatically create customized multimedia content. These services, which help manage content, are related to authoring.

For example, multimedia application video streaming projects and kiosks can scale up gracefully with automated content management. Automated content management refers to processes and practices that eliminate the rote work of organization and let the author(s) concentrate on the individual bits of content. For example, multimedia kiosks placed all over downtown Los Angeles serve identical information about local services, news, weather, maps, and so on. If these kiosks are to be placed in San Francisco or moved nationwide, then specific information must be targeted to each kiosk, or group of kiosks, to satisfy local customer needs. The information also needs to be automatically updated as the local businesses or roadways change. This dynamism and change of previous authored content can greatly benefit from

automated services. Another example is Google News. Google collates information from various Web sites and automatically puts together information Web pages with audio, video, and text, with appropriate formatting and hyperlinks. This automated content management process allows for information to be constantly up to date.

Content management systems normally work by managing a piece of content from conception to destruction. There are many ways to organize data elements in their life span. Sample ways of organizing data include nomenclature rules to name data elements consistently, collections of data elements as records that can be indexed in a database, or even annotating the media elements with textual descriptions to which semantics can be attached. The latter point, which amounts to creating metadata, is central to creating multimedia services. After being organized, multimedia elements can be included in various applications and service offerings, held in content archives. Additionally, they can even be modified, creating multiple versions of the content—all of which need to be managed. Content management systems can also be distributed across many different organizations—content providers, application builders, network service providers, and even the end consumer. By definition, the system needs to be scalable addressing large pieces of elements. Also, lack of creating customizations helps increase the scalability of the content—if content is customized to a specific service provider, or specific end client, it is harder to reuse it for other markets.

10 ASSET MANAGEMENT

Software authoring tools and paradigms are becoming more sophisticated. With this, the ability and need to create expressive and longer multimedia content have also increased. It is a common trend now for multiple users and production companies to put together digital media elements and create longer, more interactive and voluminous content. Small productions make use of a few media elements but, as the size of the multimedia production increases, so do the number of multimedia elements used, the storage space required, and the number of people involved. One of the major organizational issues here is managing the various digital media elements, known as multimedia assets, which are bound to grow on a production. They are frequently revised and moved from production to production, or shared between productions and even simultaneously worked on by different people. The protocols used to help automate the control, distribution, versioning, storage, and retrieval of media elements are known as digital asset management. Although not strictly used to author content, it is a requirement to properly manage large-scale productions.

Many production companies and production units face the challenge of managing digital multimedia assets that exist as a combination of unstructured and structured data. The multimedia objects themselves, including still image, text, video, and audio, often reside as multimedia files in various file systems and on distributed servers. Additionally, metadata describing or referencing these assets is often stored in databases because of its highly refined support for queries, integrity, backup, and scalability.

Moving multimedia objects into a digital asset management system can make it difficult to use existing tools; keeping two copies, one in the file system and the other in the repository, doubles storage requirements while introducing significant synchronization challenges.

11 EXERCISES

1. **[R03]** In the text, we have talked about multimedia. Another term that is commonly used to describe interactive content is hypermedia, much in the same way that hypertext is used to convey "linked" text. We also know about static and dynamic media types. Answer the following questions about hypermedia:
   - Extrapolating from the understanding of text and hypertext, explain how the multimedia and hypermedia differ.
   - What does hypermedia mean in terms of static media and what does it mean in terms of dynamic media? Give examples.
2. **[R04]** Authoring involves intramedia and intermedia issues.
   - What is the difference between them?
   - Briefly discuss what kinds of operations are needed to support a wide range of large-scale multimedia applications.
3. **[R07]** This is an extension exercise 1. Video can be hyperlinked to other media items (hypervideo) just as normal text is hyperlinked on a Web page today.
   - Describe a few applications where hypervideo can be a useful application.
   - If only using hypervideo, and based on your examples above, what do you think are some of the common requirements an authoring tool should provide for this purpose?
   - How would you go about designing a dedicated authoring tool for hypervideo?
4. **[R06]** One of the first tasks in any multimedia production is *storyboarding*. This is where people working on the project get a rough but firm understanding of what the multimedia content is supposed to look like, how the flow of content is to be designed, and how to interact with it. A typical example of this is in moviemaking, where images of characters and environments are drawn sequentially to show the flow and the mood of a movie. Storyboarding, thus, provides an invaluable initial organization process to help understand what the finished content should look like.
   - Explain what kinds of elements you would need to depict a digital storyboard.
   - Elaborate on how you would show the interactivity elements on a storyboard.
   - Any authoring tool can be viewed as an evolving storyboard—explain.
5. **[R05]** The text has mentioned different authoring metaphors or paradigms that are commonly used in multimedia authoring software to create multimedia content.
   - What is an authoring paradigm?
   - Briefly describe any two authoring paradigms.
   - You have been asked to create a multimedia presentation from a set of multimedia video and audio files that is distributed over the Internet. Your task is to sequence a series of videos available at one site over a series of audio files that are at another site. You can assume that the lengths of the audio and video sequences match. Which multimedia authoring paradigm is best suited to provide the best solution for this kind of task?
   - Illustrate your answer with suitable fragments of pseudo code.

6. **[R06]** In multimedia application, multimedia data is often combined together in an authoring process.
   - Why is integration of multimedia data a potential problem for multimedia systems? Give examples of some problems.
   - Briefly describe how these problems are addressed in such systems.
7. **[R06]** Although e-mail is prevalently used, you want to create an application that can provide a purposeful use of media in e-mail. This is not the same as e-mailing images and video files as attachments, but rather you want to use other media to facilitate communication in an e-mail like system.
   - What media should be supported in such a mail system?
   - What practical problems do you see with such multimedia e-mail communications systems?
   - How should an application facilitate assembly, delivery, and reading of the mail?
   - How is authoring content like this different from using some of the standard authoring tools?
   - Explain how the media is dealt with in this application compared with other authoring applications.
8. **[R09]** One aspect that we have not touched on in multimedia authoring is the distributed nature. The content might not be located at a single location, but at multiple locations. A distributed authoring environment will need to collect all that information appropriately, which can then be viewed by local and distributed clients.
   - Although the authoring requirements discussed in the chapter will not change, what other additional requirements will be imposed by the distributed nature of an application?
   - Can you think of a few multimedia distributed applications where the content to be served is changing?

   One area that might benefit from a good distributed authoring tool is the news world. Here, multiple different sources are constantly collecting news media—video, audio, graphics, text, and so on. This is sent at different times to a central news hub, which might be a TV studio. The TV studio has to author content that needs to be broadcast to all its viewers in different regions, each having variously imposed constraints depending on the area, demographics, and other personalized and/or targeted requirements. You can easily see the complexity involved in authoring such dynamic, distributed, and personalized content.
   - Visualize and explain a snapshot of how this system is working and the flow of data.
   - The central TV station and its distributed affiliates have the responsibility of putting together the content. Design a block-level architecture for how such a system should work. Ignore the bandwidth/compression requirements and concentrate only on the needs to author.
   - Remember there is no interactivity so far, but should you want to impose interactivity, how and where will you impose it?

9. **[R05]** In the text, we have mentioned how software version control systems (such as CVS and SVN) are often used to manage code development on software projects and maintain a concurrent and collaborative development of software programs.
   - What qualitative features do such systems have that enable the concurrent development and management of software code among multiple authors, who in this case are software engineers?
   - Can such systems be used for collaborative use of text documents, such as a simple .txt file or a Microsoft Word document? Why and where will this setup break, if at all?
   - Can we use the same system for multimedia authoring that involves audio, video, images, and text? Mention a few scenarios where this will not work or will be cumbersome to work with.
   - What additional features would you expect that such systems provide to make it useful for multiple users to author multimedia content collaboratively and concurrently?
10. **[R05]** All operating systems nowadays, for example Windows, Mac OS, Linux, and other smaller ones for handheld devices, all make use of multimedia elements. These are primarily used to show information—such as images (icons)—and are capable of displaying and controlling video, audio, and graphics as desired by a user.
    - From an operating system standpoint, what issues do you see in creating multimedia data that can affect performance?
    - If you wanted to design a futuristic "multimedia" operating system where graphical animations, environments, videos, and audio can be used to visualize, interact, and store data, what features would you design to enhance your current usage of the computer (whether desktop or networked)?

PROGRAMMING ASSIGNMENTS

11. **[R06]** This is a practical exercise in which you will make a multimedia presentation of yourself. At the end of it, you should be familiar with the production process by using recording instruments, software programs, such as Adobe Premiere and Photoshop, and multimedia authoring tools, such as Macromedia Dreamweaver and/or Director. You might want to do this over a few days to compile content.
    - Write a small text description about yourself and your achievements.
    - Next, you want to capture three to four media items that relate to some of your hobbies. For each hobby, take a digital picture (or find a digital picture) to represent the hobby. Bring all images into Adobe Photoshop and create a thumbnail image of size 160×120. Perform any appropriate image editing you deem necessary.
    - Capture using a digital camcorder (or find) a digital video to explain each hobby. This could be you performing the hobby or any other video explaining it.
    - Find your choice on music, either a WAV, MP3, and so on.

- Bring each video into Adobe Premiere (should come with accompanying audio, if there). Learn how to use the timeline in Adobe Premiere. For each hobby movie, edit out unwanted sections, and add your music track in the background (so you can have yourself speaking at times with the music in the background). Export out an AVI at 29.97 fps having a 4:3 aspect ratio.
- Now that you have all the intramedia processing done, you next want to use all these media elements to create a Web page, or make a Flash file with added animations. To set up a Web page, import your text and thumbnail images in Dreamweaver. Organize it to have your textual description above and thumbnail images below. Link each image to the appropriate video and publish an HTML file. If you know HTML scripting, you might choose to do all this in a text editor yourself without Dreamweaver.

12. **[R08]** This programming exercise combined with the next one are designed to give you a good understanding of the authoring and publishing process. In this question, you will write an authoring tool that will import video, audio, and images, assemble them together using a timeline metaphor, and set up simple (but specific) interactivity. The authoring tool will ultimately publish a presentation. The next programming exercise deals with designing a simple multimedia player that can play video and audio and show the presentation as it was authored. In the authoring tool, you want to provide functionality for the following:

    - Importing a video and audio clip and one or more images onto an "authoring area."
    - Positioning video and images at some position in your scene.
    - Scaling the visual media types so that frame size or image size can be adjusted. Adjusting a frame size corresponds to scaling all the frames of video accordingly.
    - Tracking some object or area (graphically specified) in your video so that you can click on the area and take some interactive action.
    - Publishing a file in a media format, which can be loaded in a media player (designed in the next programming exercise).

    For setting up interactivity, you can make use of a timeline. The timeline allows you to jump to any frame and define (or display if defined) interactive properties that have been chosen. You are required to provide three kinds of functionality on a selected media object type.

    - Start/Stop/Pause playing video and audio—for example, clicking on an object (which acts like a button). This way, you can define your own user interface look.
    - Go to a specific Web site.
    - Display an object (such an image) for a set frame range.

    Figure 5-13 shows a simple, but functional setup for such an authoring tool. You are free to design your own.
    The markings in Figure 5-13 are explained next:

    A) The loaded video in the authoring tool is shown in the center. Moving the bottom indicator should allow you to scrub to any frame in the video, at which you can define interactivity elements.

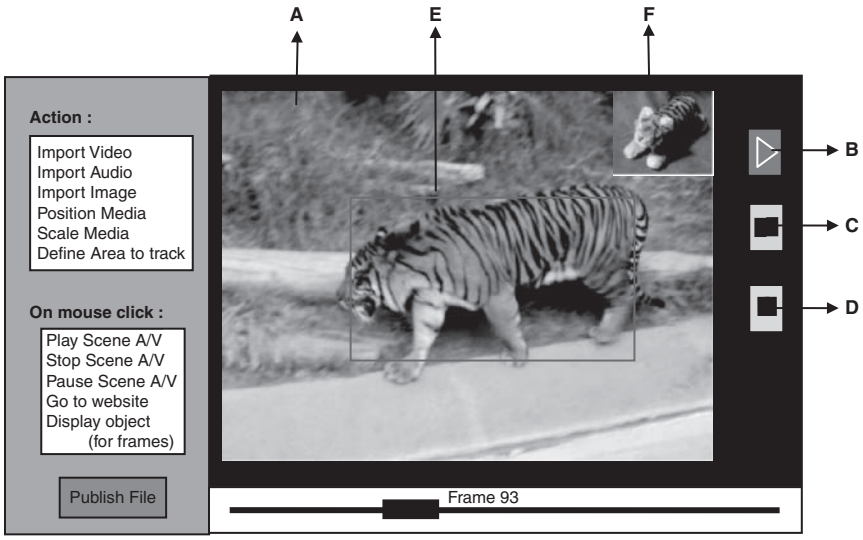


Figure 5-13

B) Shows an image object. For this object (which could be literally any media type), you might choose to play the scene's video (if any) and audio (if any) content.

C) Same as B), but you can choose to pause the A/V play.

D) Same as B) but you can choose to stop the A/V play.

E) Shows a graphical rectangular area. This shows a region of interest, which might change from frame to frame. You can assume that you interpolate the regions in intermediary frames if two or more such noncontinuous frames are specified.

F) Shows an image which allows you to click and go to another sponsored Web site. The image might show up on a defined set of frames or all the frames—depending on how it has been authored.

The publish button in the figure should allow you to save the interactive content. The format of the published file is up to you and your design, but instead of resaving the video/audio/image content, you should just keep pointers to it with the appropriate scaling, positional, and interactive properties. Content such as this can be described in various ways—for example, a simple version would define it frame by frame, giving properties of each media type for each frame, or you might even define a slightly more complex but efficient multimedia language using scene graphs.

13. **[R08]** This programming exercise goes hand in hand with the previous one. Here, you are asked to design a multimedia player that will load the published file with the interactivity that you have defined. The interactivity should allow you to play, pause, and stop on mouse-clicking objects (for example, image buttons). If defined, it should allow you to go to a Web site or display an object for a certain frame range. Here are snapshots of a sample player:

- *Snapshot 1*—Figure 5-14 shows frame 20. Keeping in line with the description of the previous example, clicking on objects A, B, and C should allow you to play, pause, and stop the video.

Figure 5-14

- *Snapshot 11*—Figure 5-15 shows frame 95. Keeping in line with the description of the examples so far, clicking on objects A, B, and C should allow you to play, pause, and stop the video. Object D shows up for a defined frame range as specified during authoring. Clicking on it should take you to the Web site that you defined during authoring time. Also, clicking in area E (a rectangular area defined for the frame or obtained as a result of interpolating areas between two end frames) should allow you to take the necessary action defined during authoring—like going to a Web site, printing informative content, and so on.

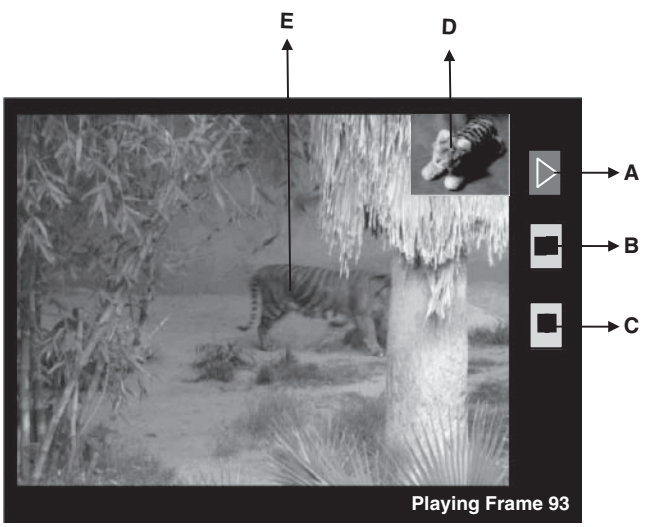


Figure 5-15

CHAPTER 6

Overview of Compression

The amount of digital media data that is produced in the form of text, video, audio, 3D graphics, and combinations of these media types is extraordinarily large, and the rate of creation increases every day. This growing mass of data needs to be stored, accessed, and delivered to a multitude of clients over digital networks, which have varying bandwidths. Whereas fiber-optic networks can provide a bandwidth upwards of 1G bits/second, others, such as cell phone networks, are limited to a range of 20–300 kbps. The third generation (3G) cell phone standard provides transmission bandwidths up to 5 Mbps. On the consumer side, people are expecting to receive multimedia data in the form of video, audio, and graphics on a variety of devices, from televisions, set-top boxes, and desktop computers to handheld portable devices such as PDAs and cell phones. The existence of voluminous data, from creation to storage and delivery, motivates the need for compression.

Data compression is a well-understood area with practical standards in place for compressing and communicating individual data types—text, images, video, audio, and graphics. This second part, Part II, of the book has five chapters dedicated to explaining media compression theory, media compression algorithms, and the compression standards that are now in place. In this chapter, which is the first chapter of the compression part, we give an overview of the issues in compression theory, discuss its theoretical limits, analyze generic techniques, and discuss implications that regularly come up in practical systems, such as real-time needs, compression ratios, time complexities, and so on. The subsequent chapters in Part II explain how these generic techniques are adapted to compress a specific media type by exploiting signal redundancy in that media type along with human perceptual understanding.

This chapter begins by explaining the need for compression in Section 1 and explaining the current state of the art in compression. Section 2 presents a formal definition of information and describes theoretical limits associated with it. Section 3 gives a taxonomy of commonly used compression techniques. These are divided into lossless compression techniques, described in Section 4, and lossy compression techniques, described in Section 5. Finally, Section 6 provides a discussion on practical issues that relate to building encoders and decoders.

1 THE NEED FOR COMPRESSION

The need to compress media data is motivated by both storage requirements and transmission requirements. In earlier days of the information age, disk space was limited and expensive, and bandwidth was limited, so compression was a must. Today, inexpensive storage in a variety of forms is readily available, and high-bandwidth connections (DSL, fiber optics and T-1) are becoming more popular. However, the amount and type of information we consume has also increased many fold, with the use of images, audio, video, and graphical animations. Furthermore, modern applications, such as high-definition video, require even larger bandwidths. This is coupled with the fact that other devices on wireless networks still do not have the bandwidth that a connection-oriented network provides. Let us take the example of HDTV. The interlaced HDTV (1080i) format has a frame size of 1920×1080. With YUV 4:2:0 subsampling, every color pixel is represented by 12 bits. This brings the required bandwidth to $1080 \times 1920 \times 12 \times 30 = 745$ Mb/s. Home network connections do not support this type of bandwidth and the digital signal will need to be compressed for delivery. HDTV 1080i is considered to be acceptable when compressed down to 18 Mb/s.

Storage devices are now more affordable and more compact than ever before. But, although multigigabyte hard disks have become commonplace for usage with desktop and networked computers, so has the use of large-scale digital media in the form of videos, high-definition images, documents, and so on. Furthermore, many smaller devices do not provide the options of carrying around large memory cards. For example, most popular consumer digital cameras now produce 5-megapixel images. This amounts to 15 MB in raw, uncompressed form per image, assuming 24 bits of RGB per pixel. The storage needs alone are very demanding, and the information contained in one such uncompressed image takes a long time to write to disk. Therefore, captured images are compressed prior to writing them to the memory devices. These two concepts of storage and transmission of data might seem like two different areas, but the data economics are the same. Each bit, whether stored or transmitted, has a cost associated with it. Compression of media is, therefore, a necessity to make many applications work with respect to storage and networked delivery. To compress multimedia data, we look at data as containing *information*. Compression, then, can be considered as a science that reduces the amount of data used to convey the same information. It relies on the fact that information, such as an image or video, is not a random collection of pixels but exhibits order and patterns. If this coherence can be understood or even modeled, the information can often be represented and stored/transmitted with a lesser amount of data. The next section introduces notions of information theory to be used in the following sections.

2 BASICS OF INFORMATION THEORY

The fundamental advances in information theory were developed in the 1940s at Bell Labs by Claude Shannon. When transmitting information from a source to a destination, information theory concerns itself with the *efficiency* and *reliability* of the transmission.

- *Efficient transmission*—How efficiently can a source transmit information to a receiver over a given channel and, thereby, communicate the information in the least amount of bits?
- *Reliable transmission*—How reliably can a source transmit information to a receiver over a given channel, knowing that the channel is noisy and can corrupt the information during transmission.

The first point relates to compression. To formalize the problem of communicating information from a source to a destination, Shannon proposed his model of inserting an encoder that encodes data produced by an information source and a decoder that decodes the information received by a receiver. Shannon's model is illustrated in Figure 6-1. Here, the source is assumed to be stochastic by nature and produces information symbols that can be described with probabilistic variables. This model eliminates the semantic role of information in communication. The encoder provides a method of efficiently coding information symbols produced by the source, which are decoded by the decoder. This process is known as s*ource coding* and involves formalizing information and compression. The basic terminology used to represent information and coding is described in the next sections.

The second point deals with ensuring reliable delivery of data across a noisy channel by adding redundant bits to the coded symbols. With added redundancy, the receiver is able to detect transmission errors and even correct them. This analysis is broadly termed as channel coding and is beyond the scope of this chapter.

To compress information, we need to formalize what *information* means, and thereby have a metric to measure the amount of information a sequence of bits and bytes contains. Information theory allows us to describe the process to compress data without losing its *information content*. The data could represent virtually anything—simple text, documents, images, graphics, and even binary executables. In this chapter, we treat all these varied data types as generic data represented digitally in a binary form by a sequence of 1s and 0s.

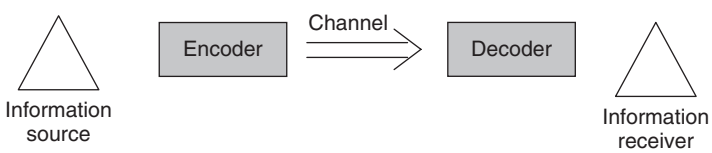


Figure 6-1 Shannon's communication model. Information produced by a source is encoded in binary form and decoded at the receiver.

2.1 Information Theory Definitions

To understand compression, it is necessary to understand the information content of a message. Information relates to the organization in the data as a sequence of symbols. If the sequence changes, so does the information. For example, you might have a binary data stream that consists of 80,000 bits. These bits might not need to be treated individually, but might instead be grouped into symbols. For instance, if it is known that bits represent a gray-intensity image with each pixel represented by 8 bits, then we have 10,000 pixels or symbols. Furthermore, if width and height of the image are both 100, the bit stream might be interpreted as a gray image of size 100×100. Here, each symbol is represented by 8 bits, and there can be $2^8 = 256$ different possible gray levels or symbols. Also, the arrangement of the symbols is important. An image representing a scene carries more information than a black image, where all the pixel or symbols are zero. The following sections define terms and concepts that will be useful to understanding information and compression.

2.1.1 Alphabet and Symbols

Information can be thought of as an organization of individual elements called *symbols*. An *alphabet* is defined as a distinct and nonempty set of *symbols*. The number or length of symbols in the set is known as the *vocabulary*. We can define an alphabet of symbols as $S = \{s_1, s_2, s_3, s_4 \ldots s_n\}$. Though n can be very large, in practice, an alphabet is limited and finite and, hence, has a well-defined vocabulary.

In the previous example involving a gray image, each pixel is represented by 8 bits and can have one of 2^8 or 256 unique values. Here, vocabulary consists of 256 symbols, where each symbol is represented or coded by 8 bits. The English alphabet has 26 unique letters and, hence, has a vocabulary of 26 symbols. In binary representation, each symbol will need 5 bits ($\log_2 26 = 4.7$, but because bits are represented as a whole, each symbol will need 5 bits). CD audio data is digitized at 16 bits/sample. Because each sound sample is a 16-bit number, the alphabet has a vocabulary of $2^{16} = 65{,}536$ symbols. An RGB color image is represented by 3×8 bits per sample. In this case, the vocabulary consists of $2^8 \times 2^8 \times 2^8 = 16{,}777{,}216$ or nearly 17 million colors. In the next chapter, we study JPEG compression, where the algorithm breaks the image into blocks of size 8×8. Because each block forms the basic unit of information, it can be considered as a symbol. Each block has 64 pixels, with each pixel represented by 8 bits (assuming a one-channel image). Thus, each symbol requires $8 \times 64 = 512$ bits. Hence, the vocabulary in this case is $2^{512} = 10^{77}$ unique symbols.

2.1.2 Sequence

A series of symbols of a given alphabet form a *sequence*. For the alphabet $\{s_1, s_2, s_3, s_4\}$, a sample sequence is $s_1\, s_2\, s_1\, s_2\, s_2\, s_2\, s_1\, s_2\, s_3\, s_4\, s_1\, s_2\, s_3\, s_3$. A sequence of symbols is also termed a *message* produced by the source using the alphabet, and represents information. A change in the sequence corresponds to a change in the message and, thereby, a change in the information. Each symbol has a binary representation and, hence, a message translates digitally to a bit stream. An image, a video, and text are all examples of binary messages.

When looking at a sequence, some symbols might occur more commonly than other symbols. For example, in the preceding sequence, the symbol s_1 occurs more frequently than the others. In the English language, the letter *e* occurs more frequently than the other letters. The frequency of occurrence of a symbol is an important factor when coding information represented by the symbols. The frequency is also known as probability and is defined in the following section.

2.1.3 Symbol Probability

The different organization of symbols captures the information in a message. The number of times a symbol occurs in the entire message, on the average, measures the likelihood, or probability of that symbol. The probability of occurrence is defined by the ratio of the number of occurrences of that symbol over the length of the entire message.

$P_i = \frac{m_i}{N}$, where m_i is the number of times symbol m_i occurs in the message of length N.

For analysis, the probabilities can be measured for a message alone, but it might make more practical sense to measure probabilities based on a large sample set. For example, when dealing with the English alphabet, it is better to measure the usage of the letter *e* in the language by measuring how many times it occurs over a number of sentences, rather than a single word or just one sentence.

Most coding algorithms make extensive use of symbol probabilities to obtain optimal codes for compression. Later on in this section, we see how probabilities play an important role in computing the fundamental limit of compression that can be attained. Here is an example to illustrate the importance of symbol probabilities. If a source produces four symbols s_1, s_2, s_3, and s_4, then two bits ($\log_2 n$, where n is the number of symbols) are required to represent each symbol in a binary code, assuming no further information of their probabilities of occurrence. Let us say that in a message of 100 symbols, s_1 occurs 70 times, s_2 occurs 5 times, s_3 occurs 20 times, and s_4 occurs 5 times. With each symbol represented by 2 bits, this coding method produces 200 bits for the message. This is illustrated in the first table of Figure 6-2. However, if we distribute the code representation unequally, so that the more frequently occurring symbols have shorter code words, the entire message could be represented by fewer bits, as the other tables show.

2.1.4 Uniquely Decodable Codes

A code *C* for a message set *S* represented by symbols is a mapping from each symbol to a bit (binary) string. This mapping can be defined as $\mathrm{C} = \{(s_1, c_1), (s_2, c_2) \ldots (s_n, c_n)\}$. Codes are used in computer science on an everyday basis. For example, the ASCII code maps every printable character to a 7-bit representation. For compression, however, a code needs to be implemented such that more-frequent symbols are given shorter code words than less-frequent symbols. These are called variable-length code words, which have a potential problem. If the code words are not properly chosen, the encoding from symbols to bit codes makes it ambiguous and, thus, hard or even impossible for the decoder to detect where one code word finishes and the other one starts. The decoder receives the binary codes and has to uniquely define the set of symbols that the binary string maps to. If this is possible, the code is called uniquely decodable.

Symbol	Code	Number of occurrences (probability)	Bits used by each symbol	
s_1	00	70 (0.7)	$70 \times 2 = 140$	
s_2	01	5 (0.05)	$5 \times 2 = 10$	200
s_3	10	20 (0.2)	$20 \times 2 = 40$	
s_4	11	5 (0.05)	$5 \times 2 = 10$	

Symbol	Code	Number of occurrences (probability)	Bits used by each symbol	
s_1	1	70 (0.7)	$70 \times 1 = 70$	
s_2	001	5 (0.05)	$5 \times 3 = 15$	140
s_3	01	20 (0.2)	$20 \times 2 = 40$	
s_4	000	5 (0.05)	$5 \times 3 = 15$	

Symbol	Code	Number of occurrences (probability)	Bits used by each symbol	
s_1	0	70 (0.7)	$70 \times 1 = 70$	
s_2	01	5 (0.05)	$5 \times 2 = 10$	110
s_3	1	20 (0.2)	$20 \times 1 = 20$	
s_4	10	5 (0.05)	$5 \times 2 = 10$	

Figure 6-2 The left column of tables shows different code assignments that can be used for symbols s_1, s_2, s_3, and s_4 in the same message. The top table has equal code length for all symbols, whereas the bottom two tables have variable-length codes. The boxes on the right show the total number of bits used for the 100 symbol message. The code representation affects the overall bits used. The first two tables have uniquely decodable codes, but the third does not.

In the example illustrated in Figure 6-2, the first two tables have uniquely decodable codes. Any sequence involving s_1, s_2, s_3, and s_4 can be coded such that the resulting bit stream can be uniquely decoded by the decoder to reconstruct the original symbol sequence. This is not the case with the code in the third table. The code in the third table maps the message "$s_1\ s_1\ s_2\ s_3$" to a binary sequence "00011". This can be decoded as "$s_1\ s_1\ s_2\ s_3$" or as "$s_1\ s_1\ s_1\ s_3\ s_3$". The third table represents a code that is not uniquely decodable.

2.1.5 Prefix Codes

A prefix code is a special kind of uniquely decodable code in which no one code is a prefix of another code. For example, C = $\{(s_1, 1), (s_2, 001), (s_3, 01), (s_4, 000)\}$ is a prefix code. In general, prefix codes do not have to be binary. Prefix codes can be used in ternary representations as well. However, for our purpose, we limit our discussion and encoding to binary prefix codes. It is necessary that the codes generated by the encoder for each symbol be prefix codes, and, hence, uniquely decodable by the decoder.

2.2 Information Representation

Say a source has a vocabulary of N symbols $\{s_1, s_2, \ldots, s_N\}$. It produces a message consisting of M symbols, which are emitted at frequency of $1/T$ Hz, which is one symbol every T seconds. Furthermore, let each symbol s_i be emitted m_i times so that the frequency of s_i in the message of M symbols is m_i. Also let the length of each symbol s_i be given by l_i. Then, we have the following relationships:

- The message of length M contains m_i instances of symbol s_i.

$$M = \sum_{i=1}^{i=N} m_i$$

- The total number of bits that the source has produced for message M is given by

$$L = \sum_{i=1}^{i=N} m_i l_i.$$

- The average per symbol length is given by

$$\lambda = \left(\sum_{i=1}^{i=N} m_i l_i\right) \Big/ M.$$

- Additionally, in terms of symbol probabilities, λ may be expressed as

$$\lambda = \left(\sum_{i=1}^{i=N} m_i l_i\right) \Big/ M = \sum_{i=1}^{i=N} \left(\frac{m_i}{M}\right) l_i = \sum_{i=1}^{i=N} P_i l_i$$

  where P_i is the probability of occurrence of symbol s_i.

Given a set of symbols represented by s_i for all i in an alphabet having probabilities P_i, the goal of compression then becomes that of finding a binary prefixed code $\mathrm{C} = \{(s_1, c_1), (s_2, c_2) \ldots (s_n, c_n)\}$ where each code word c_i has code length l_i such that the average code length λ is minimized. In other words, compression algorithms attempt to find a C, such that the average number of bits required per symbol is the minimum. Since λ is a function of the product of two quantities, namely the probabilities of symbols P_i and their associated code lengths l_i, as P_i increases, l_i should decrease and vice versa. The next logical question is how low can the value of λ be? This was proved by Shannon to be measured by the entropy.

2.3 Entropy

Shannon's information theory borrows the definition of **entropy** from physics to quantify the amount of information contained in a message of symbols given their probabilities of occurrence. For source-producing symbols, where each symbol i has a probability distribution P_i, the entropy is defined as

$$H = \sum P_i \log_2\left(\frac{1}{P_i}\right) = -\sum P_i \log_2 P_i.$$

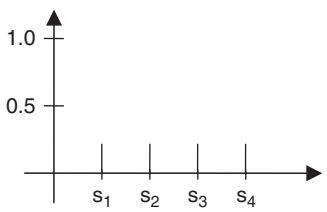
For the symbol s_i having a probability P_i, Shannon defined the notion of self-information of the symbol given by $log_2(1/P_i)$. This self-information represents the number

of bits of information contained in the symbol and, hence, the number of bits used to send that message. When the probability is high, the self-information content is low and vice versa. Entropy, then, becomes the weighted average of the information carried by each symbol and, hence, the average symbol length.

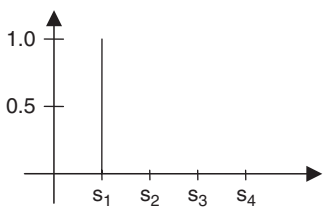
The entropy depends only on the probabilities of symbols and, hence, on the source. It is highest when the source produces symbols that are equally probable (all P_i have the same value) and reduces as some symbols become more likely to appear than the other. The computations in Figure 6-3 show sample entropies given the probability distributions of four symbols $\{s_1, s_2, s_3, s_4\}$.

Information entropy is very similar to the entropy discussed in Physics: The second law of thermodynamics states that entropy measures the amount of randomness in a system—entropy increases with randomness. In our system of information, randomness corresponds to the arbitrary distribution of symbols. This occurs when

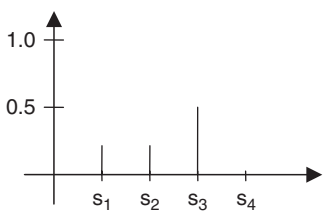
$$P_i = \{0.25,\ 0.25,\ 0.25,\ 0.25\}$$
$$H = -(4 \times 0.25 \times \log_2 0.25)$$
$$H = 2$$



$$P_i = \{1.0,\ 0.0,\ 0.0,\ 0.0\}$$
$$H = -(1 \times \log_2 1)$$
$$H = 0$$



$$P_i = \{0.25\ \ 0.25\ \ 0.5\ \ 0.0\}$$
$$H = -(2 \times 0.25 \times \log_2 0.25 + 0.5 \times \log_2 0.5)$$
$$H = 1.5$$



$$P_i = \{0.125\ \ 0.125\ \ 0.25\ \ 0.5\}$$
$$H = -(2 \times 0.125 \times \log_2 0.125 + 0.25 \times \log_2 0.25 + 0.5 \times \log_2 0.5)$$
$$H = 1.75$$

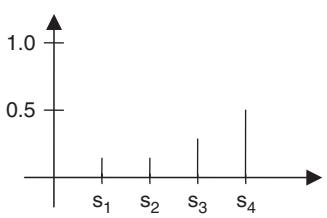


Figure 6-3 Entropy. Four examples of probability distribution for the symbols $\{s_1, s_2, s_3, s_4\}$ and the corresponding entropies are shown. The first distribution with equal probability assignments for all symbols has the highest entropy.

probabilities of symbols are equal. When the entropy is higher (or the information content is higher), more bits per symbol are needed to encode the message. On the other hand, when symbols are not random but seem more predictable (for example, a long strings of symbol s_1 in a message), some symbols have a higher probability than the others, which results in a lower entropy, hence, in lower information content. In this case, fewer bits per symbol are needed to encode the message. To illustrate this, consider the two messages shown in Figure 6-4. Each message is a black-and-white image made up of symbols (s_1, s_2), which are either black or white. The first message is a completely black image. Here, P_1 is 1 and P_2 is 0. H in this case evaluates to zero, suggesting that there is no information in the image. The second picture has equal distribution of black and white pixels. Here, P_1 and P_2 are both roughly 0.5. H in this case evaluates to 1, which is exactly the minimum number of bits needed to represent the two symbols ($s_1 = 0$, $s_2 = 1$).

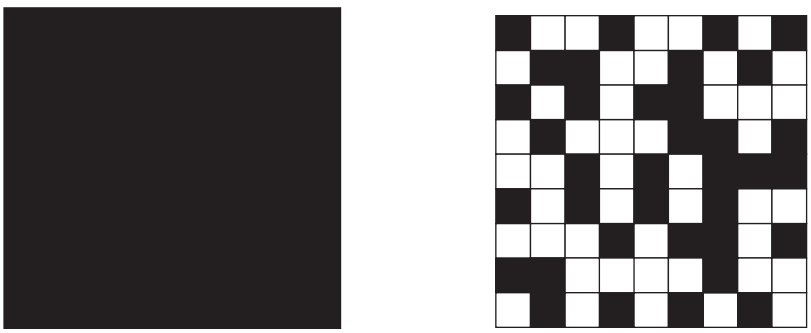


Figure 6-4 Two images are shown. The left image contains all black pixels and the information content is lower resulting in zero entropy. The right image contains roughly equal distribution of black and white pixels. The information content is higher in this case and the entropy evaluates to 1.

2.4 Efficiency

Given an alphabet of symbols, the goal of any coding system is to transmit the information contained in a string composed of these symbols in the most efficient manner, that is, in the least number of bits. The coding system needs to evaluate the frequency of occurrence of each symbol in the string and then decide how many bits to code each symbol so that the overall number of bits used for that string is the least. The previous section explained that the optimal average symbol length is given by the entropy. Thus, any coding system's goal is to assign binary codes to each symbol in a manner so as to minimize the average symbol length. In such cases, *efficiency* is an important metric to evaluate the coding system's ability of compression and is defined as

$$Efficiency = \frac{Entropy}{Average\ Symbol\ Length}.$$

In the next sections, we see how these probabilities are utilized in obtaining codes for compression.

3 A TAXONOMY OF COMPRESSION

A variety of data compression techniques compress multimedia data. They fall into either the lossless or lossy categories. Lossless compression, as the name suggests, results in a compressed signal, which produces the exact original signal when decompressed. In lossy compression, the compressed signal when decompressed does not correspond to the original signal. Lossy compression produces distortions between the original and the decompressed signal. Figure 6-5 shows a chart that categorizes lossless and lossy compression techniques. Both techniques can be used separately, or in a combined manner for various applications and have resulted in a variety of standards used to compress text, images, video, audio, and so on.

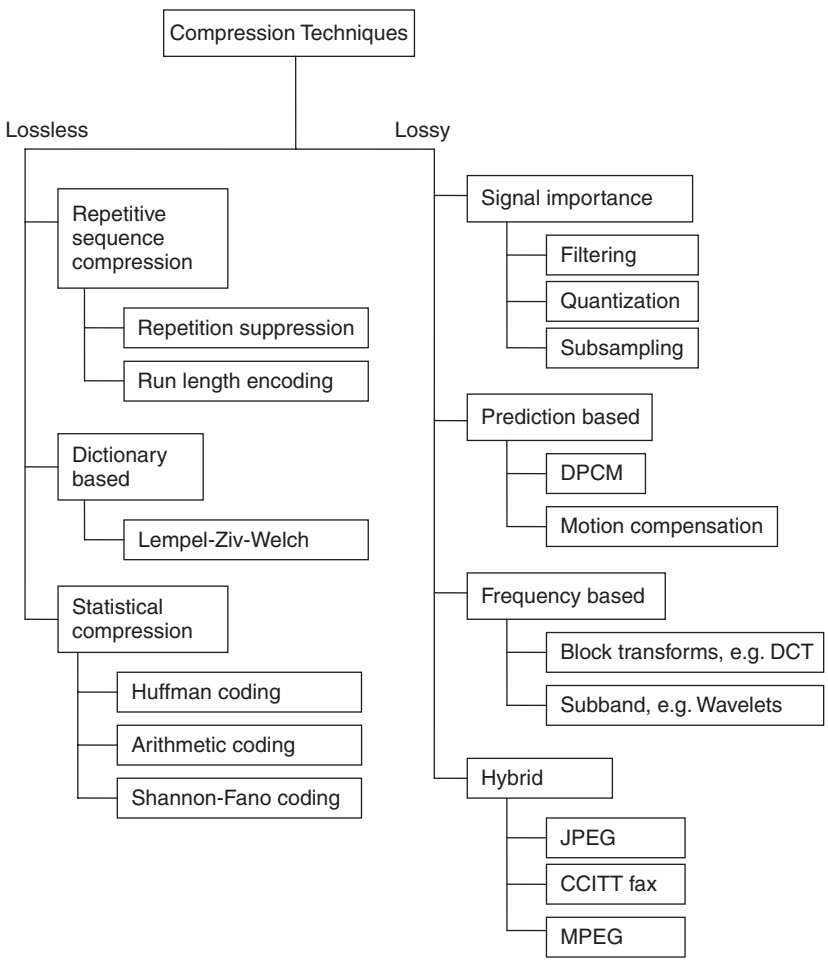


Figure 6-5 Taxonomy of compression techniques. Lossless or entropy-coding techniques are shown on the left; lossy techniques are shown on the right.

3.1 Compression Metrics

To characterize compression, two metrics are commonly used—*compression rate* and *compression ratio*. Both are related, but the former relates to the transmission of data, whereas the latter relates to storage of data. Compression rate is an absolute term and is simply defined as the rate of compressed data, which we imagine as being transmitted in real time. It is expressed as bits/symbol—such as bits/sample, bits/pixel, or even bits/second. It, thus, defines the bit rate of the compressed signal. Compression ratio is a relative term and is defined as the ratio of the size (or rate) of the original data to the size (or rate) of the compressed data. For example, if an audio signal is originally represented as 16 bits per sample and is compressed down to 4 bits per sample, the compression ratio is 4-to-1.

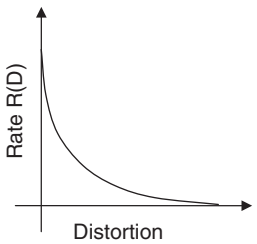
Lossless and lossy compression can be better understood in terms of the above-mentioned compression rate and compression ratio. Lossless compression normally assigns different codes to the original samples or symbols by studying the statistical distribution of those symbols in the signal. Thus, the bit rate produced by the compressed signal, or the compression rate, is bound to vary depending on how the signal's symbols are distributed. This produces a variable bit rate for the compressed signal. This might not pose a problem for some applications involving offline compression, but variable rates are undesirable when delivering or streaming real-time data. In such cases, lossy techniques are used, where the compressed signal can be distorted to achieve a constant bit rate and are useful for streaming video and audio and for real-time delivery of media across networks.

3.2 Rate Distortion

In lossy compression, compression corrupts the information; therefore, the reconstructed signal is not the same as the original signal. An important metric of such lossy techniques is the amount of distortion caused by compression. To measure distortion, a distortion criterion has to be established. In most cases, statistical criteria that measure the difference between the original and reconstructed data are used. A simple example of such a criterion when dealing with image data might be to take the sum of pixel-by-pixel differences. However, this might not correspond to the perceived distortion. When evaluating distortion with media such as images, video, and audio, perceptual distortion measures are often used. For example, if an image is simply shifted to the left by a single pixel, the perceived distortion is negligible, but the measured one is very large.

To better formalize distortion measurements, let us assume that an original signal y is compressed, and then reconstructed as $\hat{y}$. The distortion or error in this case can be measured by the $error = f(y - \hat{y})$, where f is some function that measures the difference. One popular function is the mean square error. However, others, such as signal-to-noise ratio (SNR) and peak signal-to-noise ratio (PSNR) are also used. These measure the error relative to the signal and are often more informative than a simple mean square error.

The graph in Figure 6-6 illustrates that lossy compression is always a trade-off between rate and distortion, which is represented in the form of a rate distortion function $R(D)$. If D is the amount of distortion that is acceptable, $R(D)$ gives the lowest rate at which the original signal can be compressed, while keeping the distortion bounded above D. When $D = 0$, $R(D)$ is highest and this corresponds to lossless compression. The



$$\sigma_d^2 = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

$$SNR = \frac{\sigma_{\hat{y}}^2}{\sigma_d^2}, \quad SNR(db) = 10 \times log\left(\frac{\sigma_{\hat{y}}^2}{\sigma_d^2}\right)$$

Where $\sigma_{\hat{y}}$ is the averaged squared value of the output

$$PSNR(db) = 10 \times \log\left(\frac{\sigma_{peak}^2}{\sigma_d^2}\right)$$

Where σ_{peak} is the squared value of the peak output

Figure 6-6 Rate distortion. The top figure shows a typical rate distortion function R(D). As compression increases, the number of bits used or rate decreases and this increases the distortion. The bottom formulae show typical error measurements—mean square error, signal-to-noise ratio, and peak signal-to-noise ratio.

function $R(D)$ establishes the theoretical performance limits for lossy data compression, and is frequently used to evaluate the performance of various compression algorithms.

4 LOSSLESS COMPRESSION

Lossless compression techniques achieve compression by removing the redundancy in the signal. This is normally achieved by assigning new codes to the symbols based on the frequency of occurrence of the symbols in the message. As mentioned earlier, more frequent symbols are assigned shorter codes and vice versa. Sometimes the entire message to be coded is not readily available and frequency of symbols cannot be computed a priori. This happens when a source is "live," for example, in real-time communication systems. In such cases, a probabilistic model is assumed and the source symbols are assigned a model probability. The lossless compression algorithms make use of the probabilistic models to compute efficient codes for the symbols.

In most practical cases, probabilistic models have to be assumed. Although the source will produce symbols in accordance to the model over large time intervals, it is not unnatural for a source to produce a sequence that does not conform to the probabilistic model over shorter intervals. For example, in the English language *e* is the most frequently occurring letter—13% on average—and *z* is the least frequently occurring letter—0.07% on average. However, in a certain sentence "Zoos display zebras from Zambia, Zimbabwe, and Zaire," there are more occurrences of *z* than *e*. In this case, using the probabilistic model would not result in an efficient compression, and sometimes might even do the opposite. Thus, lossless coding techniques based on

a probabilistic model might not necessarily guarantee compression when the distribution of symbols in the message does not conform to the probabilistic model used. The next few subsections describe the commonly used lossless compression processes.

4.1 Run Length Encoding

Run length encoding is the simplest form of redundancy removal. It removes redundancy by relying on the fact that a string contains repeated sequences or "runs" of the same symbol. The runs of the same symbol are encoded using two entities—a count suggesting the number of repeated symbols and the symbol itself. For instance, a run of five symbols *aaaaa* will be replaced by the two symbols 5*a*. A sample string of symbols is as follows:

$$BBBBEEEEEEEECCCCDAAAAA.$$

It can be represented as

$$4B8E4C1D5A.$$

Here, *BBBB* is represented as 4*B*, EEEEEEEE is represented as 8*E*, and so forth. The original 22-byte message (assuming a byte for each symbol) has been reduced to 10 bytes, giving a compression ratio of 2.2. Although this is the general framework of run length encoding, it needs to be more sophisticated in practice: The preceding example will not work when a string contains numbers—how would you know whether the number in the encoded string was the length of a run or part of the string content? Also, it is not practical to encode runs of length 1, so how would you tell when a run starts and when a literal sequence starts? The answers to these questions and others related to run length encoding are addressed in exercise 3 and 4 at the end of this chapter.

Run length encoding performs best in the presence of repetitions or redundancy of symbols in the information. This corresponds to a low entropy case and does result in efficient compression of the data. However, if there is no repetition or the seeming randomness in the symbols is high, run length encoding is known not to give any compression and, in fact, depending on implementation flavors, it could very well result in increasing the size of the message.

Run length encoding is used in a variety of tools involving text, audio, images, and video. For example, when two people are having a telephone conversation, there are gaps represented by zero values when nobody is speaking. It is also not uncommon in images and video to have areas, which have the same pixel distribution, such as in the background. Run length encoding has also become a part of compression standards—it is supported by most bitmap file formats such as TIFF, BMP, and PCX and a variant of it is even adopted in the JPEG compression pipeline, as discussed in the next chapter.

4.2 Repetition Suppression

Repetition suppression works by reserving a specific symbol for a commonly occurring pattern in a message. For example, a specific series of successive occurrences of the letter *a* in the following example is replaced by the flag ψ:

$$Abdcbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa$$

is replaced by

$$Abdcb\psi.$$

Repetition suppression can be considered as an instance of run length encoding, where successive runs of specific characters are replaced.

4.3 Pattern Substitution

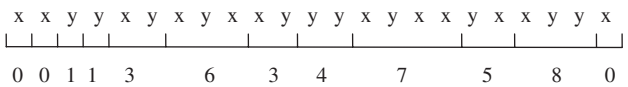
This class of compression algorithms is based on the dynamic creation of a dictionary of strings. Each string is a subsequence of symbols, as they appear in the message. It was derived in the late 1970s by Lempel and Ziv and later refined by Welch. It is now popularly known as the LZW compression algorithm. The compression is based on a simple idea that attempts to build a group of individual symbols into strings. The strings are given a code, or an index, which is used every time the string appears in the input message. So, the whole message consisting of symbols is broken down into groups forming substrings, each substring having an index. Its simplicity resides in the fact that no preanalysis of the message is required to create a dictionary of strings, but rather the strings are created as the message is scanned. Compression is achieved when a single code (or index) is used for a string of symbols rather than just the symbol. Here is the process:

- Initialize the dictionary to contain all initial symbols. The vocabulary forms the initial dictionary entries. Every entry in the dictionary is given an index.
- While scanning the message to compress, search for the longest sequence of symbols that has appeared as an entry in the dictionary. Call this entry **E**.
- Encode **E** in the message by its index in the dictionary.
- Add a new entry to the dictionary, which is **E** followed by the next symbol in the scan.
- Repeat the process of using the dictionary indexes and adding until you reach the end of the message.

The algorithm is also described in the following pseudo code.

```
InputSymbol = Read Next input symbol;
String = InputSymbol;
WHILE InputSymbol exists DO
        InputSymbol = Read Next input symbol;
        IF  (String + InputSymbol) exists in DICTIONARY then
             String = String + InputSymbol
        ELSE
             Output dictionary index of String
             Add (String + InputSymbol) to DICTIONARY
             String = InputSymbol
       END // end of if statement
END // end of while loop
Output dictionary index of String
```

Figure 6-7 illustrates a running example. Here the dictionary is first initialized with x and y (the initial symbol set) having indexes 0 and 1. The string is shown at the top of the figure and along with it the iterative output of indexes produced by the algorithm. The coded output produced by the algorithm is "0 0 1 1 3 6 3 4 7 5 8 0." The middle table shows the process at each iteration. This includes the current longest string that can be replaced using the current dictionary, and the next new string that is added to the dictionary. The bottom table shows the constructed dictionary of strings for the input message.



Iteration number	Current length symbol(s)	Dictionary code used	Next symbol in string	New dictionary code generated	New dictionary index generated
				x	0
				y	1
1	x	0	x	x x	2
2	x	0	y	x y	3
3	y	1	y	y y	4
4	y	1	x	y x	5
5	x y	3	x	x y x	6
6	x y x	6	x	x y x x	7
7	x y	3	y	x y y	8
8	y y	4	x	y y x	9
9	x y x x	7	y	x y x x y	10
10	y x	5	x	y x x	11
11	x y y	8	x	x y y x	12
12	x	0	END		

Index	Entry	Index	Entry
0	x	7	x y x x
1	y	8	x y y
2	x x	9	y y x
3	x y	10	x y x x y
4	y y	11	y x x
5	y x	12	x y y x
6	x y x		

Figure 6-7 Dictionary-based encoding. The top illustration shows the input sequence and the dictionary indexes while encoding. The middle table depicts the step-by-step iteration while dictionary codes and indexes are generated. The bottom table shows the final set of indexes and dictionary code words.

Decompression works in the reverse manner. Given the encoded sequence and the dictionary, the original message can be obtained by a dictionary lookup. Note that transmitting the entire dictionary to the decoder is not necessary. It is important to realize that from the way the algorithm works, the dictionary can be re-created from the coded message itself, based on the initial vocabulary (x and y, in this example). The decoder knows that the *last* symbol of the most recent dictionary entry is the *first* symbol of the next parse block. This knowledge is used to resolve possible conflicts in the decoder.

Although simple in its working, practical implementation needs to handle a few issues for efficient running. For example, how long should the dictionary grow? Clearly, as the dictionary size increases, so does the size of the strings that form the dictionary and consequently the compression gets better. However, along with the dictionary growth, the number of bits required to represent the index also grows (bits required $b = log_2 N$, where N is the dictionary size). It is impractical to have a large N, and this value is normally limited to 4096, which gives 12 bits per index or it might also be an input parameter to the algorithm. Once the limit is reached, no more dictionary entries can be added. In practice, the LZW algorithm works well only when the input data is sufficiently large and there is sufficient redundancy in the data. Many popular programs, such as the UNIX-based compress and uncompress, gzip and gunzip, and the Windows-based WinZip program, are based on the LZW algorithm. It is also used while compressing images using the GIF format.

4.4 Huffman Coding

When data is initially sampled, each sample is assigned the same number of bits. This length is $\log_2 n$, where n is the number of distinct samples or quantization intervals. This is true of audio data (8 bits or 16 bits per sample), images (24 bits per pixel), text ASCII data, and so on. Assigning an equal number of bits to all samples might not be the most efficient coding scheme because some symbols occur more commonly than others. The goal, as mentioned earlier, is to have more frequent symbols that have smaller code lengths and less frequent symbols that have longer code lengths.

This variable-length representation is normally computed by performing statistical analysis on the frequency of occurrence of each symbol. Huffman coding provides a way to efficiently assign new codes to each symbol, depending on the frequency of occurrence of the symbol. The process starts by computing or statistically determining the probability of occurrence of each symbol. The symbols are then organized to form the leaves of a tree, and a binary tree is built such that the most frequent symbol is closer to the root and the least frequent symbol is farthest from the root. This can be outlined as follows:

- The leaves of the binary tree are associated with the list of probabilities. The leaves can be sorted in increasing/decreasing order, though this is not necessary.
- The two smallest probabilities are made sibling nodes by combining them under one new parent node with a probability equal to the sum of the child node probabilities.

- Repeat the process by choosing two least probability nodes (which can be leaf nodes or new parent nodes) and combining them to form new parents until only one node is left. This forms the root node and results in a binary Huffman tree.
- Label the branches with a 0 and 1 starting from the root and going to all intermediary nodes.
- Traverse the Huffman tree from the root to a leaf node noting each branch label (1 or 0). This results in a Huffman code representation for that leaf node symbol. Perform the same for all the symbols. Note that the symbols closer to the root and, hence, most frequent have smaller bit code lengths than the symbols farther away from the root.

This can be illustrated with an example shown in Figure 6-8. There are eight symbols A, B, C, D, E, F, G, H whose probabilistic distributions are as shown.

Huffman coding was considered to be optimal until arithmetic coding was developed in the 1970s.

4.5 Arithmetic Coding

Unlike the methods discussed in the previous sections, arithmetic coding overcomes the requirement of every symbol to be coded independently. In other words, coding every symbol was represented individually by a code, or a group was represented in case of a run, or dictionary word. Thus, a whole number of bits were required to encode a symbol (or symbol group). Arithmetic coding overcomes this constraint by mapping an entire message to a real number between zero and one. This real number representing the entire message is coded as a binary number. Arithmetic coding, thus, encodes a message entirely without assigning a fixed binary code to each symbol and, thereby, tends to produce better compression ratios.

The coding process uses a one-dimensional table of probabilities instead of a tree structure. Given an alphabet of n symbols, there are an infinite number of messages that are possible. Each message is mapped to a unique real number in the interval [0,1). The interval contains an infinite amount of real numbers, so it must be possible to code any message uniquely to one number in the interval. The interval is first set at [0,1) for the first symbol, and then partitioned according to the symbol probabilities. Depending on the first symbol of the message, an interval is chosen, which is then further partitioned depending on probabilities. The algorithm can be outlined as follows. Suppose we have a vocabulary of n symbols:

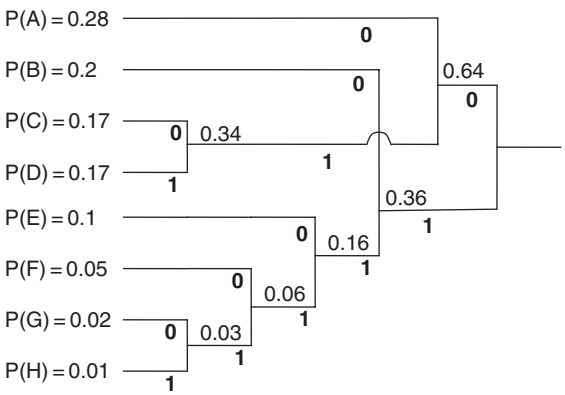
1. Divide the interval [0,1) into n segments corresponding to the n symbols; the segment of each symbol has a length proportional to its probability. Each segment i has an upper bound U and lower bound L corresponding to the start of the segment and the end of the segment ($U - L = P_i$).
2. Choose the segment that corresponds to the first symbol in the message string. This is the new current interval with its computed new upper and lower bounds.
3. Divide the new current interval again into n new segments with length proportional to the symbols probabilities and compute the new intervals accordingly.

Symbol	Probability	Binary code	Code length
A	0.28	000	3
B	0.2	001	3
C	0.17	010	3
D	0.17	011	3
E	0.1	100	3
F	0.05	101	3
G	0.02	110	3
H	0.01	111	3

Average symbol length = 3

Entropy = 2.574

Efficiency = 0.858



Symbol	Probability	Huffman code	Code length
A	0.28	00	1
B	0.2	10	3
C	0.17	010	3
D	0.17	011	3
E	0.1	110	3
F	0.05	1110	4
G	0.02	11110	5
H	0.01	11111	5

Average symbol length = 2.63

Entropy = 2.574

Efficiency = 0.9792

Figure 6-8 Huffman coding. The top table shows the symbols used with their probability distribution and original binary code length. Each symbol has the same code length 3. Also shown are the entropy and the efficiency of the code. The middle figure shows the Huffman tree with branches labeled in bold. The bottom table shows the symbol codes obtained from the Huffman tree and the corresponding code lengths. The average code lengths are decreased, thus increasing the coding efficiency.

4. From these new segments, choose the one corresponding to the next symbol in the message.
5. Continue Steps 3 and 4 until the whole message is coded.
6. Represent the segment's value by a binary fraction in the final interval.

The process is illustrated by the example in Figure 6-9 (top), where at every stage, a new unique interval is refined to a smaller range that ultimately results in a unique interval result. Shown are two symbols *a, b* having probabilities as follows $P(a) = 2/3$, $P(b) = 1/3$. To encode the message of length 4 "*abba*", we start with the interval [0,1) and divide it into two segments that correspond to the probabilities of *a* and *b*. These intervals are [0, 2/3] and [2/3,1). The first symbol is *a* and hence the interval [0,2/3] that corresponds to *a* is chosen. This interval is again broken into two segments corresponding to the two symbols *a* and *b*. At the second stage, the interval [4/9, 6/9] is chosen

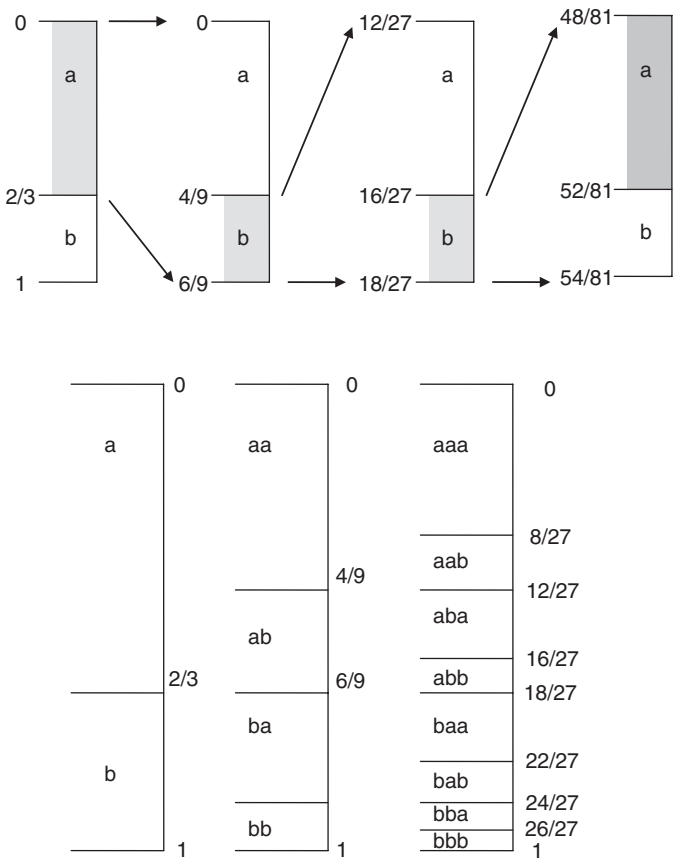


Figure 6-9 Arithmetic coding. The top picture illustrates the coding process to code the message "abba" given the original symbol probabilities. The bottom picture shows all possible intervals up to level 3 using the same probability distribution. At each stage, every subinterval is broken into two intervals in the ratio of the symbol probabilities.

because it corresponds to the second symbol *b*. This process repeats for the third symbol *b* and fourth symbol *a* till we get the final interval [48/81, 52/81]. This interval uniquely represents the message "*abba*" and thus any number in the interval can be chosen as a binary representation of the message. The number that is typically chosen is one that has the least number of bits in that interval. For the interval [48/81, 52/81], the binary representation of this range (up to eight binary decimals) is [0.10010111, 0.1010010]. This interval can be represented by 0.101, which corresponds to the decimal 0.625. Thus, 101 should suffice as a code for "*abba*". This information is enough for the decoder to reconstruct the sequence "*abba*" having knowledge of the probability distributions.

The top figure shows the mapping of the sequence "*abba*" to an interval. If the sequence changes, so does the mapped interval. The second illustration at the bottom of 0 describes this by showing all possible outcomes for the same probability distribution of symbols *a* and *b* in the example. For simplicity, all the eight outcomes up to level 3 are shown, but this could in theory go to much larger. Let us take "*baa*" as another example. This sequence maps to the interval [18/27, 22/27]. This range in binary can be represented by [0.10101010, 0.11010000]. For this, you can see that 0.11 (0.75) defines this range and hence the code 11 suffices for "*baa*". Note, any other number in the range would still be OK, but the least binary number is typically preferred.

The algorithmic implementation is very important in practice. A direct implementation of the description will have two major difficulties:

- The ever-shrinking intervals require high-precision floating-point arithmetic to find the representative binary code for each interval.
- Second, the entire message is mapped to an interval and so it will take an end-to-end scan of the message to produce a code, resulting in no output until all the symbols in the message have been processed.

Practical solutions solve the first issue by preventing the shrinking when the interval bounds get too close. In such cases, the interval is scaled and remapped around 0.5 such that the upper bound is above 0.5 and the lower bound is below 0.5. By moving the interval, we have lost the binary ability to compute the next bit, but by using this process we know that whatever this bit might be, it will have opposite signs if above or below 0.5. The second issue is solved by incrementally transmitting the entire message as a number of messages, each message dealing with a fixed block of symbols.

The last statistical-based method mentioned in the table—Shannon-Fano coding is described and further analyzed in exercise 11 at the end of this chapter.

5 LOSSY COMPRESSION

Entropy-coding or lossless compression has theoretical limits on the amount of compression that can be achieved. In certain situations, it might be necessary and appropriate to sacrifice some amount of information and thereby increase the compression obtained. For instance, human visual experiments on image perception have shown distortion introduced by image compression is still perceptually acceptable. In such

cases, the original signal cannot be recovered in its entirety and there is a loss or distortion introduced. Most lossy compression schemes introduce a distortion because of quantizing the symbol code representations. Quantization is inherently lossy. This section explains some of the commonly used lossy compression techniques.

5.1 Differential PCM

PCM data consists of digitally coded samples. Signals contain all kinds of frequencies, which cause it to change over time. When a signal does not have high frequencies in it, it changes slowly, causing successive samples to be similar in value. If successive samples are similar in value, they can be predicted using the current and past samples. The closer the prediction is to the actual value, the lesser the prediction error. So, instead of transmitting the signal values, we can transmit the errors between the predicted value and the actual value. The decoder decodes a symbol by predicting it based on a previous set of samples as used by the encoder and adds to it the error that it received from the encoder. This is the basic working of prediction-based encoders. Differential pulse code modulation (DPCM) works by doing the simplest of prediction. The simplest prediction is the signal itself. If the $(n-1)^{\text{th}}$ sample is represented by $y(n-1)$, then the n^{th} sample can be predicted to be $y(n) = y(n-1)$, causing the prediction error to be the difference between the predicted and actual samples as

$$d(n) = y(n) - y(n-1).$$

This is illustrated in Figure 6-10. Instead of sending the original signal $y(n)$, the differences $d(n)$ are computed. As seen in the figure, the differences $d(n)$ have a smaller dynamic range compared to the original $y(n)$ and consequently a smaller entropy. Hence the $d(n)$ values can now be represented using fewer bits. The encoder quantizes the $d(n)$ into $\hat{d}(n)$.

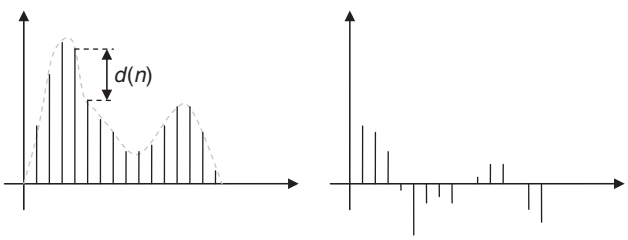


Figure 6-10 Differential pulse code modulation. The left signal shows the original signal. The right signal samples are obtained by computing the differences between successive samples. Note the first difference d(1) is the same as y(1); all the successive differences d(n) are computed as y(n) − y(n − 1).

At the encoder, DPCM works by sending quantized values $\hat{d}(n)$ of the signal $d(n)$. This is the open loop case, which is illustrated in Figure 6-11. The decoder works by recovering the lossy signal $\hat{y}(n)$ from $\hat{d}(n)$ as follows:

$$\begin{aligned} &\text{If } n = 1, && \hat{y}(n) = \hat{d}(n) \\ &\text{else} && \hat{y}(n) = y(n-1) + \hat{d}(n). \end{aligned}$$

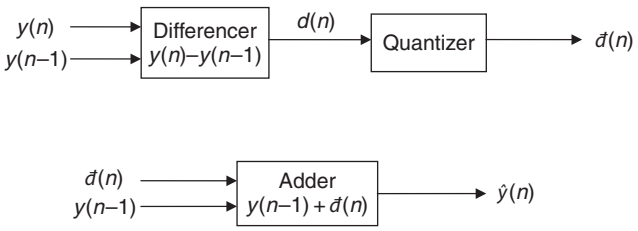


Figure 6-11 Open loop differential pulse code modulation. Encoder (top) works by predicting difference d(n) and quantizing it. Decoder (bottom) works by recovering quantized signal ŷ(n) from đ(n).

However, the previous method has one flaw. Although the decoder computes $\hat{y}(n) = y(n-1) + đ(n)$, the signal $y(n-1)$ is not available at the decoder. The decoder has the reconstructed $\hat{y}(n-1)$, which is the same as $y(n-1)$, with some quantization error e_{n-1}. This error accumulates at every stage, which is not desirable. To rectify this, a closed loop DPCM can be performed, where at the encoder, the difference $d(n)$ is not computed as $y(n) - y(n-1)$ but $\hat{y}(n) - y(n-1)$. This depicted in Figure 6-12.

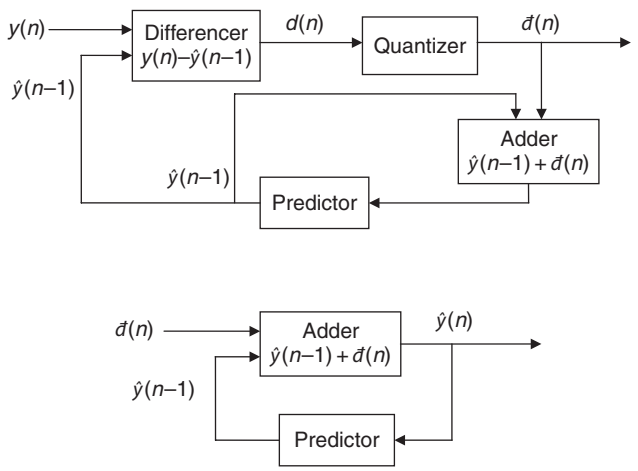


Figure 6-12 Closed loop differential pulse code modulation. At the encoder (top), the difference d(n) is computed as y(n) − ŷ(n-1). This reduces the error that accumulates under the open loop case. A decoder (bottom) is simulated at the encoder; the decoded values are used for computing the difference.

5.2 Vector Quantization

Quantization is normally performed on individual samples where the original signal value is quantized to fall in a range, and this range is represented by an index or number. Quantization causes a lossy compression and most compression techniques operate by quantizing each symbol taken individually. This is known as scalar quantization. Scalar quantization techniques have many desirable qualities, such as simple to implement, fast to quantize/dequantize, and lead to a good symbol redundancy,

which can be exploited further by entropy-coding techniques. However, this method is insensitive to intersample correlations. Normally, digital signals are not random but exhibit a high degree of correlation in the local neighborhood. Additionally, scalar quantization can also hurt correlation by quantizing each sample individually. Lower compression ratios and better qualitative results can be obtained by quantizing a *group* of samples together by largely preserving correlations. This is the general principle of operation for vector quantization and it works as follows:

- Prior to encoding, a representative set of "*code vectors*" need to be determined. These code vectors form a code book. Each code vector can be represented by a number or an index. Code vectors can be 1D (sound), 2D matrices (images), and in general *n*-dimensional. The codebook can be generated on a per-signal basis by statistically analyzing the signal. It can also be arrived at by statistically analyzing many example signals in the same domain. Irrespective of how the code book is determined, it needs to be communicated to the decoder. If the code book is determined on a per signal basis, it is normally communicated along with the compressed signal.
- The encoding process proceeds by partitioning the input signal into correlated blocks or vectors. Each input signal vector is then coded by an index, which represents the best-matched codebook vector that minimizes the distortion among the choice of code vectors. Compare this with replacing a single sample by an index representing the quantized interval. For all input vectors, we now have a list of indexes, each of which represents a code vector.
- The decoding process is a simple table lookup. Each index can be replaced by the corresponding code vector from the codebook to reconstruct the original signal.

Although the general algorithm is simple, there are a few issues that need attention. One of the foremost is the design of the codebook:

- How should these code vectors be decided?
- How many codebook vectors should there be (codebook size)?
- What is the size of each code word (code vector lengths)?

The code vectors to be chosen should be the best representative vectors for the given input data set, given its statistical properties and maximal distortion tolerated. The most common algorithm used for this purpose is the LBG (Linde, Buzo, and Gray), which is based on a training sequence from a number of similar signals. The size and structure of the codebook is an important aspect in the codebook design. A larger number of code vectors allows for representing more features and better distortion minimization, leading to better reconstructed signals after decoding. However, larger codebooks causes more storage and/or transmission, which affects the overall bit rate. Typical codebook values for image compression range from 2^7 to 2^{11}. The code vector size exploits the intersample correlations. The local correlation depends on the signal and the code vector sizes have to be chosen accordingly.

During encoding, the input data set is organized as vectors. The encoder searches for the closest code vector from the codebook. The closest vector is chosen such that it has the minimal distortion using mean square error computed as

$$MSE = \sum_{i=1}^{n} (y_i^c - y)^2, \text{ where } y_i^c \text{ is the codebook vector and } y \text{ is the input vector.}$$

The index i that corresponds to the least error is chosen as the output index. All the output indexes are communicated to the decoder. The decoder works by outputting the appropriate code vector from the codebook given the index. For decoding, the decoder has to have the codebook for that input set available. A block diagram illustrating how vector quantization works is shown in Figure 6-13.

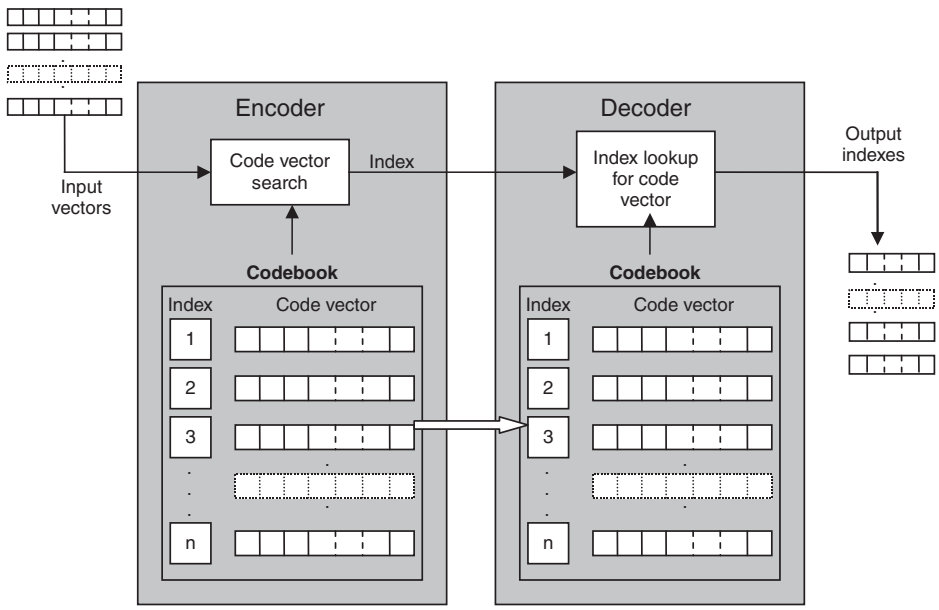


Figure 6-13 Vector quantization. On the left, input data is organized as vectors. In the encoder, a search engine finds the closest vector from the codebook and outputs a codebook index for each input vector. The decoding process (right) proceeds as an index lookup into the codebook and outputs the appropriate vector. Along with the coded indexes, the encoder needs to communicate the codebook to the decoder.

One of the simplest vector quantization schemes is used to compress color images. Each color sample is represented by three bytes for R, G, and B. When compressing the image of a scene, where most colors are shades of green (trees) and blue (sky), you can select a relatively small subset (for example, 256) of RGB colors and then approximate every vector RGB value to one of these colors. Here, instead of 3 bytes per pixel, 1 byte is used. Similarly, vector quantization can be used to compress a variety of media, such as audio, images, and video. In Chapter 9, we will see how the CELP speech compression standard uses codebooks to quantize excitation vectors obtained by linear prediction.

5.3 Transform Coding

Transform coding techniques work by performing a mathematical transformation on the input signal that results in a different signal. The transformation must be invertible so that the original signal can be recovered. The motivation for the use of such techniques is that the transformation changes the signal representation to a domain, which can result in reducing the signal entropy and, hence, the number of bits required to compress the signal. Transform coding techniques by themselves are not lossy; however, they frequently employ quantization after a transform, which results in loss of data. This process is shown in Figure 6-14.

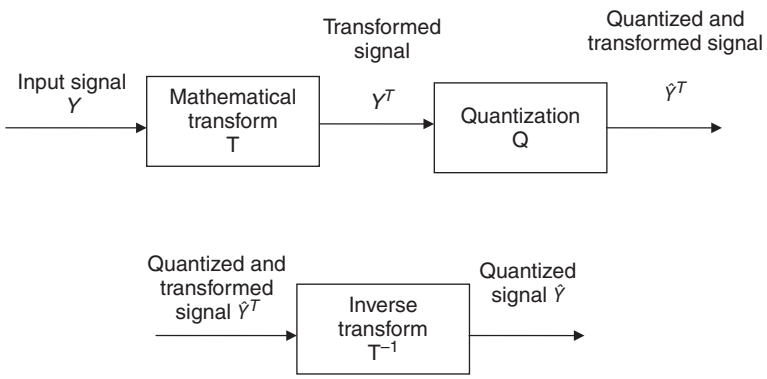


Figure 6-14 Transform coding. The top figure shows a block diagram at the encoder. The input signal Y is transformed to Y^T and then quantized to $\hat{Y}^T$. The bottom figure shows the decoder where $\hat{Y}^T$ is inverse transformed to produce the signal $\hat{Y}$.

Transform techniques are very well suited to both 1D signal (sound) and 2D signal (images) coding, thus a large number of such approaches have been proposed. These can be grouped as follows:

- *Frequency transforms*—Discrete Fourier transforms, Hadamard transforms, Lapped Orthogonal transforms, Discrete Cosine transforms
- *Statistical transforms*—Karhunen-Loeve transforms
- *Wavelet transforms*—While similar to frequency transforms, these transforms work more efficiently because the input is transformed to a multiresolution frequency representation.

The commonality of the first group of transforms deals with converting the signal from the sample domain, which might be the time domain for audio, the spatial domain for images, or the frequency domain. All the methods in the first group represent the signal by weighted coefficients of the fundamental frequencies, as explained in section 6 of Chapter 2. However, the methods differ in how the energy gets distributed among the coefficients in the frequency domain and the region of influence each coefficient has in the reconstructed picture. The Discrete Fourier and Cosine transforms have well-distributed coefficients when working on smaller signals,

but tend to become computationally unwieldy when dealing with larger images, and result in a large number of coefficients. The Lapped Orthogonal transform is a special case of these frequency-based transforms where the frequency coefficients do not influence the whole image or block, but only local adjacent neighborhoods.

The second group of transforms is based on statistical analysis of the samples in a signal or group signals. Unlike the first group, where the fundamental frequencies are used as a basis, the Karhunen-Loeve transform finds an optimal set of basis functions for the given input signal(s) by performing principal component analysis (PCA). PCA is a commonly used mathematical process that converts the original description based on the correlated signal dimensions into a different (smaller) number of dimensions. Eigen vector analysis is frequently employed in the conversion process. Unlike the first group of transforms, because the choice of the new basis computed depends on the signal content(s), the Karhunen-Loeve transform is bound to provide an optimal sense of energy compaction for the specific input set analyzed. However, the basis computed by statistical analysis on one set of signals might not work well for a different set; therefore, these methods are harder to adapt for a general-purpose standard.

Compression is obtained by quantizing the coefficients of the basis frequencies, thus reducing the number of bits used by all the coefficients. Quantization causes a loss and data cannot be recovered. However, the coefficients quantized have to be properly ordered so as to cause the least distortion. Computationally, it is not feasible to compute transformations for the entire media signal, for example an entire image or a few minutes of audio. Most standard techniques use frequency transforms by breaking the signal into blocks and transforming the blocks individually to keep the computation manageable. For instance, JPEG breaks up the image into 8×8 blocks and performs a Discrete Cosine transform on each block. This block-based transformation technique is illustrated in Figure 6-15. During encoding, the input signal Y is broken into different blocks Y_1, $Y_2 \ldots Y_n$. Each of the blocks is individually transformed and then quantized. The decoder receives the quantized blocks, which are then inverse transformed to get the reconstructed signal.

In the next chapters, we provide details of the workings of block-based discrete cosine transform (DCT) transformation in connection with image, video, and also audio compression.

Although block-based transform coding is frequently employed when compressing media data, another issue worth mentioning is that each data block's transformed coefficients can be quantized independently. This means that each block is not compressed by using the same quantization levels, but each can be compressed differently, in such a way that the overall distortion of the combined signal is minimized. This is an important consideration when attempting to control bit rates. For example, if you have B bits per frame for a video frame or audio segment, how do you distribute the B bits among the transformed blocks Y^T_1, $Y^T_2 \ldots Y^T_n$ such that you get the best quality (least distortion)? For best results, channels with higher entropy are allocated more bits.

$$B = \sum_{i=1}^{i=n} b_i$$

The third group of transforms, Wavelet transforms, deals with multispectral descriptions of a signal and how each spectral band is compressed differently. This is described in the next section on subband coding.

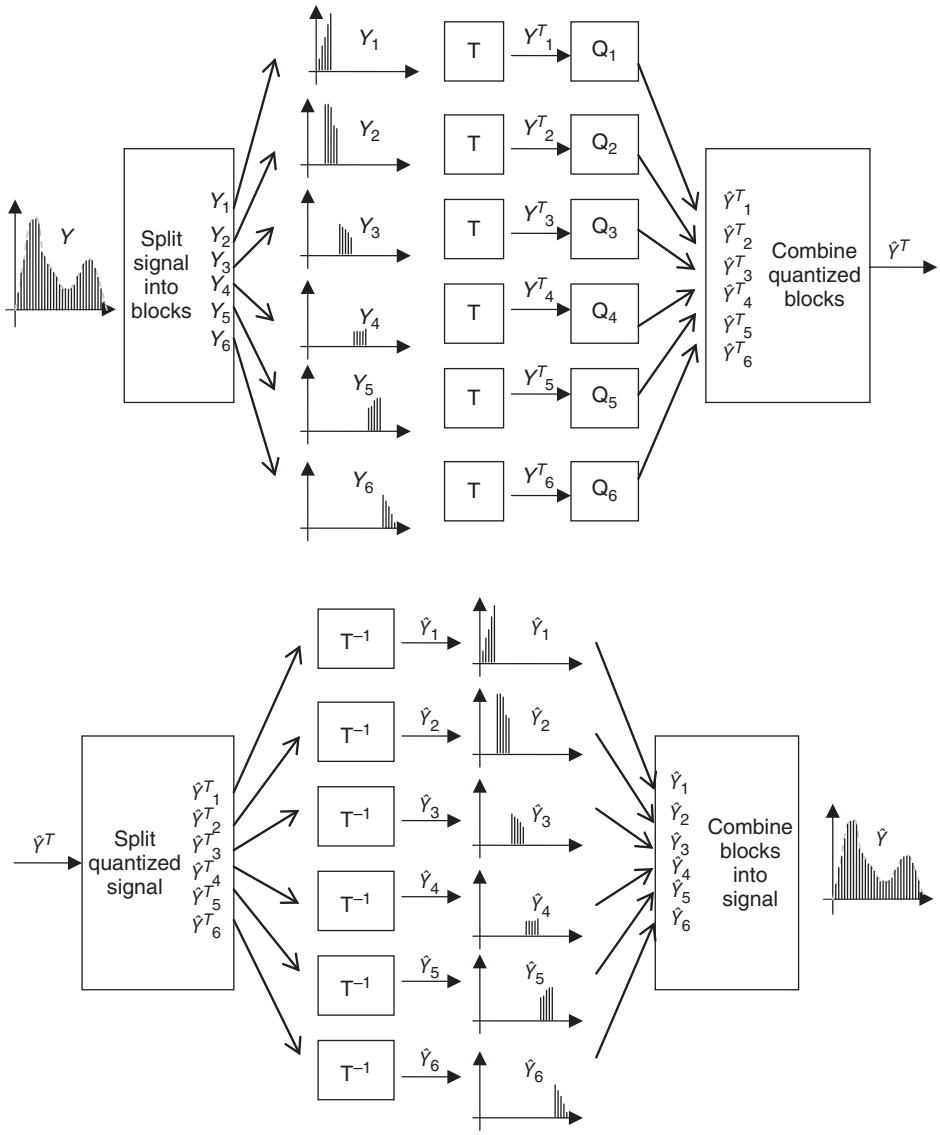


Figure 6-15 Block-based transform coding. The top figure shows the encoding process. The input signal Y is split into blocks Y_1, Y_2 . . . Y_6, each of which are transformed using the same transformation T into Y^T_1 Y^T_2 . . . . Y^T_6. Each of these are in turn quantized separately by quantizers Q_1, Q_2 . . . Q_6 and combined to produce the resultant $\hat{Y}^T$. The bottom figure shows the decoder where $\hat{Y}^T$ is split into its constituent blocks and inverse transformed to produce the signal blocks $\hat{Y}_1$, $\hat{Y}_2$ . . . $\hat{Y}_3$, which are then combined to produce the decoded signal $\hat{Y}$.

5.4 Subband Coding

To understand subband coding, we need to understand band descriptions of a signal. The basic underlying notion in subband techniques is successive approximation or multiresolution, where a signal can be decomposed into a coarse (low frequency) version with added details (high frequency) at various resolutions. In other words, the input signal can be described in stages, each stage giving a finer description of the signal. An example of such a description is shown in Figure 6-16. Any media signal

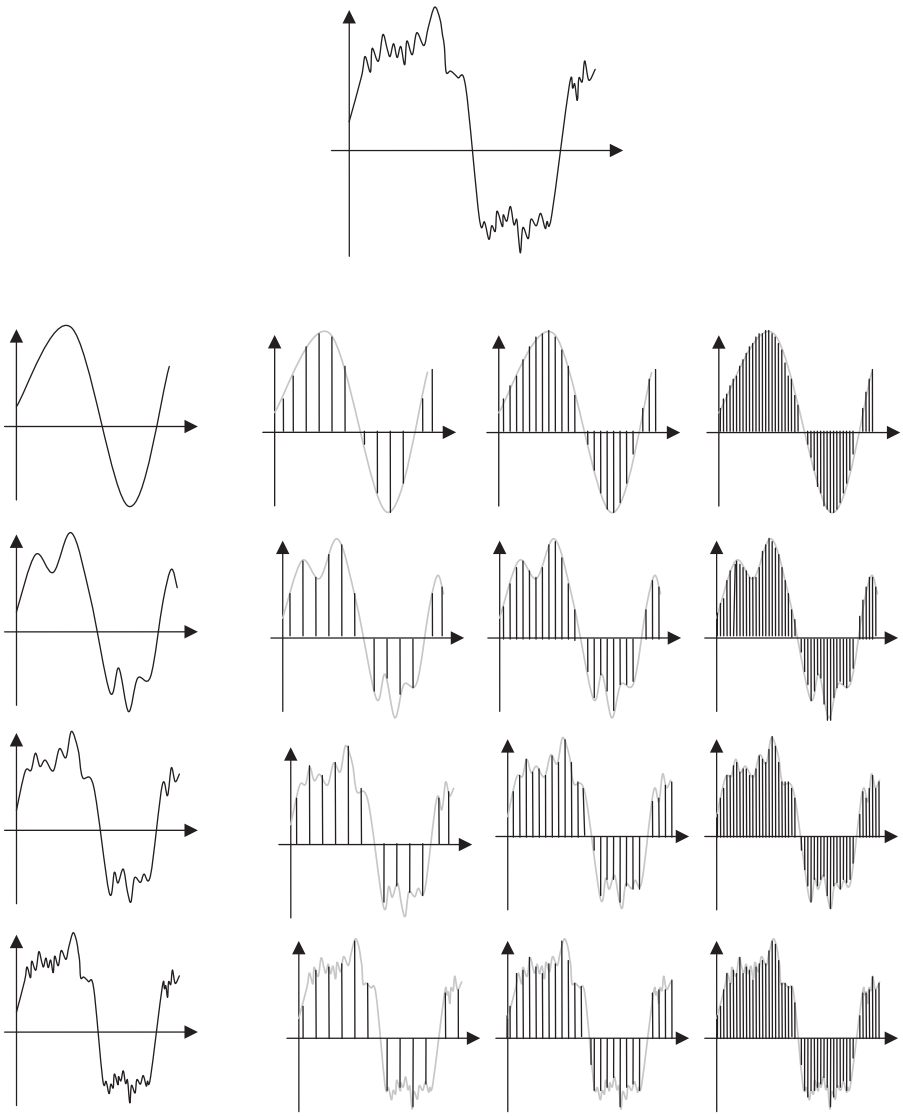


Figure 6-16 Multiresolution signal description. The top figure shows a signal with multiple frequencies. The left column shows the signal at various frequency resolutions. Each row shows a frequency resolution sampled at different time resolutions.

can be broken down into its constituent band descriptions. The motivation for doing so is that each band can now be dealt with differently depending on its importance and relevance in human perception. For example, if it is known that a certain band-x is more important than another band-y, the processing of band-y can involve more quantization than band-x. Therefore, you have the ability to distribute your bits among different bands according to importance. This is frequently done in audio coding, where frequency bands are perceived differently.

A multiband description of an input signal is shown in Figure 6-16. The input signal on the top is shown at different frequency resolutions in the left column. For each signal at a resolution, digital sampled versions of the signal are also shown. You can see that at the lower-frequency descriptions, the time resolution does not add much information and so a coarsely sampled version of the signal is sufficient to describe the low-level frequency resolution. However, as the frequency resolution increases, so should the time resolution to capture the contained frequencies.

Next, we need a way to break up a signal into different bands. This is normally done using band filters with different frequency cutoff limits. Each filter allows only the signal frequencies lying in the band to pass through. An input signal is broken down into different bands using a filter bank. For example, in audio processing, the input signal is typically broken down into 32 different bands. For wavelet-based compression, the signal is passed through a concatenation of low- and high-pass filters, as shown in Figure 6-17. At each frequency resolution, the low-pass signal is downsampled to reduce the time domain resolution. The process might be repeated recursively to get to the desired levels of frequency resolutions. Each band is quantized differently.

When compared with transform coding, subband techniques are similar in that they also perform spectral analysis, but it is important to understand how they differ. The difference is mainly in how the signal is described prior to compressing it. In transform coding, the signal as a whole (or in blocks) is taken and transformed to the frequency domain approximating it with the fundamental frequencies. The subband coding process first describes the signal by a group of signals at different spectral resolutions. Each of these band signals can then be processed individually by a transform. Alternatively, you can also apply subband coding to each block instead of a frequency transform on each block.

5.5 Hybrid Compression Techniques

The main goal of compression, whether lossy or lossless, is to minimize the data used to represent information. In the case of lossy compression, the process is subject to a distortion evaluation so that the signal perception remains relatively unaffected when evaluated by the human perceptual system. In the case of lossless compression, there is no distortion, but the method is subject to theoretical limits of how much compression can be achieved. All the various basic methodologies discussed so far in this chapter have their own advantages and disadvantages. Hybrid methods combine the good properties of these various processes.

Hybrid methods share characteristics of both methods—the lossy part is subject to the same distortion evaluation, whereas the quantized (or distorted) output is subject to the same information limits to achieve compression.

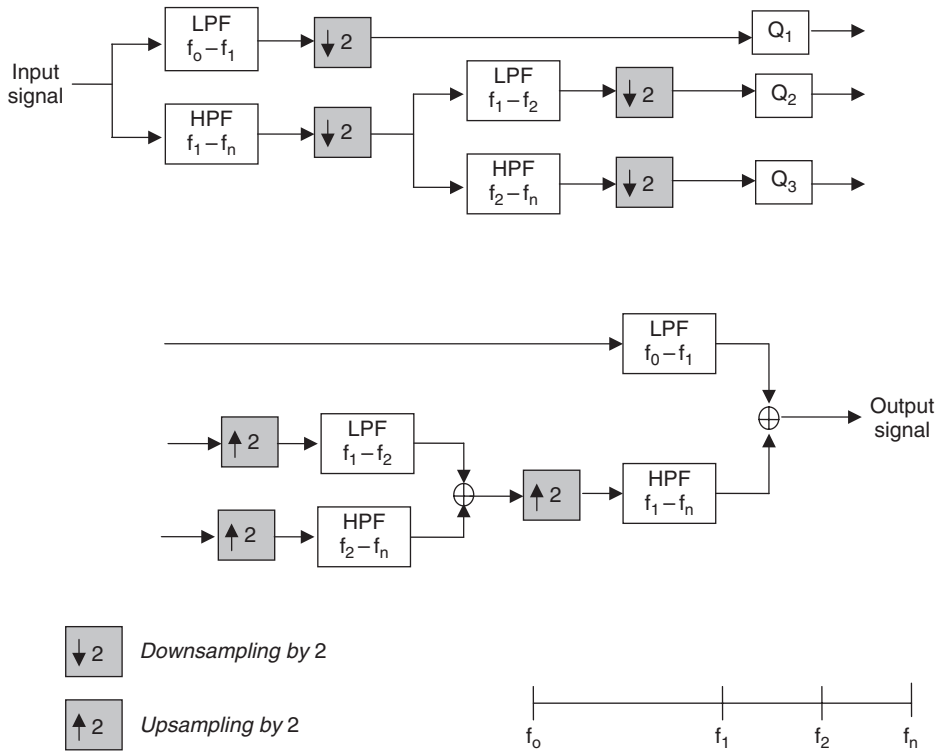


Figure 6-17 Subband coding. The top figure shows a subband encoder till level 2—the input signal is passed through low- and high-pass filters, with the low-pass outputs downsampled. The resulting signals are quantized differently. The bottom figure shows the decoder, which adds together the different filtered signals to produce the final output.

This practice of combining lossy and lossless techniques is frequently used to get better compression. Lossy compression techniques cause a loss by quantizing either the input samples of the signal itself, or the differences between the successive samples (DPCM) or frequency domain coefficients obtained by transformations (DCT, Wavelet). The quantization, which causes a loss in the information, also reduces the signal's entropy. This is because quantization effects reduce the variation in the signal. Thus, the quantized signal can be further compressed by using entropy-coding techniques. For example, the JPEG image compression standard works by transforming input blocks of data using the Discrete Cosine transform and quantizing the frequency coefficients. The quantized coefficients are then DPCM and run length ordered followed by Huffman coding to get better compression performance. At the decoder, the compressed stream is decoded using Huffman tables to generate the quantized frequency coefficients, which are then inverse transformed to obtain the image data. The detailed description of such hybrid techniques and the respective ISO standards that pertain to the different media are the subject matter of the remaining chapters in this second part of the book.

6 PRACTICAL ISSUES RELATED TO COMPRESSION SYSTEMS

The previous sections thus far have explained generic compression algorithms by categorizing them into lossless and lossy algorithms. Although this might be a useful taxonomy for overall explanation, practical communication systems take other categorizations into consideration in their design and architecture. Among these are the speed of encoding, the time/space complexity of encoding, real-time needs, rate constraints, symmetric/asymmetric coding setups, adaptive/non adaptive coding, and so on. The following sections explain these issues from a generic standpoint, but the remaining chapters in the media compression part discuss the design issues in more detail as it pertains to the individual media type.

6.1 Encoder Speed and Complexity

The speed at which an encoder needs to compress and its overall complexity is a central design issue when building end-to-end communication systems. Some factors that influence this design are as follows:

- *Performance requirements*—These requirements address the speed at which the encoder needs to perform. In a real-time scenario, the encoder needs to produce a compressed bit stream at the rate of the incoming data, whereas for offline cases, there is more time available to encode the data.
- *Data available for analysis*—The amount of signal that an encoder has access to prior to encoding also influences how the algorithm extracts statistics, sets up coding lookup tables, performs prediction-based searches, and minimizes distortion during encoding. The complexity of encoding in such cases increases compared with cases where little or no data is available.

The first point influences the design of the encoder from the data capture stage, the encoder stage to the transmission stage. In real-time design, the encoding algorithm does not have the luxury to explore exhaustive strategies that might otherwise result in better compression. For instance, encoding an offline movie with little or no time constraints allows the encoder to perform a better predictive search compared with real-time cases. As such, the speed at which the encoder has to encode is inversely proportional to the quality of compression. The second point addresses how an encoder might produce better compression by taking advantage of available data at the time of encoding. To explain this better, remember that lossy encoding will invariably produce distortion and the amount of distortion produced is directly proportional to the number of bits used to produce the compressed bit stream at that instant. Given that there is a bit budget that the encoder has to work with, an educated statistical choice of how to distribute bits unequally over different time frames during compression is important to the overall compression quality. For instance, in the case of encoding a movie, either for archival or streaming, the encoder might search through all the video frames and decide a priori how many bits to use per frame so that the overall quality is maximized. In such cases, higher

entropy frames are assigned more bits while lower entropy frames are assigned less bits. However, this is not possible if all the frames are not available, such as in the case of a live encoding.

6.2 Rate Control

In traditional media communication applications, such as digital TV broadcast, looking at digital photographs over a network, listening to an MP3 compressed audio file, video-on-demand, and so on, the signal is compressed *offline* and stored on disc. A server then serves or streams this data through a network to the end viewer. In this case, the major constraint for the encoding/decoding system is the storage space generated by the encoded files and transmission bandwidth available to serve the end client. The overall storage space is the total number of bits of output from the encoder after compression—and that has to be minimal, of course. But more important is the rate at which the encoder is generating the bits. Ideally, you would want this rate to be constant. All practical media encoding systems will have a lossy encoding scheme, followed by lossless entropy-coding. Entropy coders do not necessarily maintain a constant bit rate and the lossy distortion module of an encoder has to work together with the entropy encoder to ensure that an amenable output bit rate is maintained at all times. Thus, the overall goal in this type of encoding system design is to optimize the media quality under a rate constraint.

Media quality and bits required are two clearly competing and interdependent factors. Compressed media needs more bits for better quality. To solve this problem, rate distortion theories, sometimes also referred to as rate control, have been developed to model the relationship between the coding bit rate and signal distortion. The rate distortion theory describes the performance limit of lossy data compression and attempts to compress the source data at a given or defined distortion level using the minimum number of bits. Rate distortion theories have been a topic of active research focusing on theoretical performance bounds ever since Claude Shannon introduced information theory. However, the theoretical analysis has taken a new dimension in the presence of sophisticated media systems and standards that compress images (JPEG, JPEG-2000), video (MPEG-2, MPEG-4 AVC, H.264), and audio (MP3). This is because, unlike 1D signals such as text, whose compression characteristics can be either captured or modeled easily, 2D and 3D media objects have complex models that involve spatial and temporal frequency distribution, time varying motion patterns, arbitrary camera motion, and so on. This is analyzed for each media type in the remaining chapters of this part of the book.

6.3 Symmetric and Asymmetric Compression

Symmetric compression refers to the case when the decoder's execution is the exact reverse of the encoder's execution. Such compression methods make sense when applications are constantly encoding and decoding files. Conversely, an asymmetric compression method is one where either the encoder or the decoder is more complex than the other, consequently having unequal encoding and decoding times. Although it might seem computationally convenient to have the same complexities in the encoder and decoder, there are many applications where

unequal complexities are not necessarily bad. For example, a compression method where the encoder executes a slow, complex algorithm and the decoder is relatively faster in comparison is a natural choice when media files are compressed. Many compression algorithms for compressing media types such as video are based on complex search strategies which are time consuming. It takes considerably more time to encode a movie to create a DVD, although it can be quickly decoded and viewed. The opposite case, where the encoder is simple while the decoder is complex and slow is useful in applications where files are constantly updated, backups need to be made, and the backed up files are rarely used. The decoder here can be slow because the backup files are rarely or never used.

6.4 Adaptive and Nonadaptive Compression

Another differentiating factor among encoders is whether it is capable of adapting its encoding technique depending on the data that is produced by the source. Most simple and direct encoding techniques either use fixed statistics based on an overall average of many data sets to decide how to encode a specific data stream, or make two passes over the message, where the first pass is used to gather statistics and the second pass is used to encode the message using the extracted statistics. These statistics, among other things, help decide encoding tables that the encoder uses to encode symbols in the message. In both these cases, whether fixed or two pass, the statistical information that the encoder is using does not change and, hence, these techniques are known as *static* or *nonadaptive*. Static techniques, though used in many media compression standards, are unsatisfactory for general-purpose data compression because they cannot adapt to unexpected data. Whenever unexpected data is produced by the source, it often causes longer bit streams generated by the encoder and does not lead to compression.

Adaptive techniques adjust the decisions taken by the encoder based on the history of data produced by the source. At each step, the next set of symbols is coded based on a code constructed from the history. This is possible because both the encoder and the decoder have access to the history. For instance, in case of Huffman coding, the codes generated can be constructed using the history of data for that given signal rather than generating them based on a static setup. Such techniques are called *adaptive* encoding techniques.

7 EXERCISES

1. [**R02**] Entropy compression or lossless coding is used to compress data without losing the information content.
   - What does *information content* mean?
   - If all the samples in a signal have the same value, what is the information content of this message?
   - What kind of signal would have the maximum information content?
   - Why does entropy compression produce a variable bit rate?

2. [**R03**] You are given a data stream that has been compressed to a length of 100,000 bits, and told that it is the result of running an "ideal" entropy coder on a sequence of data. You are also told that the original data consists of samples of a continuous waveform, quantized to 2 bits per sample. The probabilities of the uncompressed values are as follows:

   00 1/2
   01 3/8
   10 1/16
   11 1/16.

   What (approximately) was the length of the uncompressed signal?

3. [**R05**] Run length encoding results in compression by describing the data in the form of runs.
   - Does it always lead to compression? When would it not lead to compression? Is such cases, does the message have high entropy or low entropy?
   - Which image will result in better run length encoding among the ones shown in Figure 6-18? Assume all images are of the same size and list them in the order of best achieved compression using RLE.

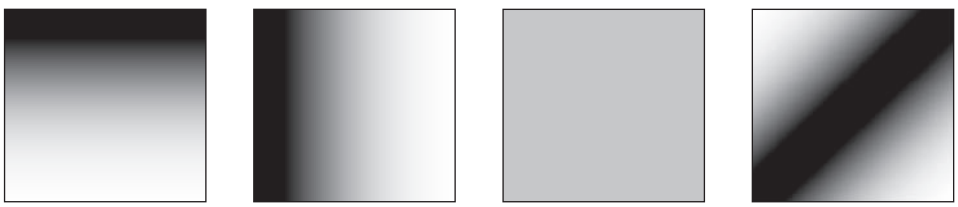


Figure 6-18

   - Suppose you had a color image with each pixel represented as 24 bits—there are 2^{24} possible symbols. You write a program that takes each pixel as a symbol and performs RLE to compress the sequence. Is there a more efficient way to use RLE in this case?
   - How does quantization affect RLE? If you were to quantize the images in Figure 6-18 before you perform RLE, would quantization help or hurt?
   - RLE is known to perform poorly when there is no repetition as suggested in the text. So while AAABBBCCCDDD compresses, ABCDEFGH does exactly the opposite, and AABBCDEF is the same. Suggest a scheme that would ensure a better performance irrespective of how the symbols occur.
   - How does your scheme perform in the best-case and worst-case scenarios?

4. [**R04**] Consider a video sequence of images with an interlaced scanning format at 60 fields per second and each image having dimensions of 960 × 960. Each image is represented by, at the most, eight colors (R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8). When the rates of occurrence of the pixels in the video stream are measured, we get the following statistics: R_1 occurs 5,250,000 times, R_2 occurs 1,500,000 times, R_3 occurs 840,000 times, R_4 occurs 15,450,000 times, R_6

occurs 75,750,000 times, and R_8 occurs 6,450,000 times. Furthermore, it is known that R_5 and R_8 occur at the same rate.

- If each color is represented by 3 bits, what is the bit rate?
- Given the statistical color distribution over five seconds, compute the frequency or probability of occurrence for each color.
- Design a Huffman coder for lossless encoding of each pixel.
- What is the average bit rate using this encoder?
- What is the efficiency of your Huffman encoder?

5. [**R06**] In a certain exam, here is a grade distribution for students:

   Grade A = 25%
   Grade B = 50%
   Grade C = 12.5 %
   Grade D = 10%
   Grade F = 2.5%.

   Using this distribution, answer the following:

   - If you do not take probabilities into effect, how many bits are required per letter grade? If probabilities are taken into account, what is the minimum number of bits required per symbol in theory?
   - Construct a Huffman code for this distribution of grades and give a bit code to each grade. What is the average bit length per symbol and the efficiency of your code?
   - For the sake of efficiency, we decide to represent grades of A, B, and C as "pass" and grades of D and F as "no pass." Clearly, 1 bit is required to represent a pass or a no pass, but what is the entropy of the information?
   - Can you design a Huffman code to represent the pass/no-pass information more efficiently than using 1 for "pass" and 0 for "no pass"?

6. [**R04**] Arithmetic coding is known to perform better on average than Huffman coding and although compression standards provide both as a usable option, Huffman coding is predominantly used.

   - Mention a few drawbacks of arithmetic coding.
   - Why is 0 not chosen as a code to map all strings in the first range?
   - Practical implementations break down code into blocks of symbols (symbol blocking) and send arithmetic codes of each block separately. What problems to you see with this, if any?
   - For the example shown in Figure 6-10 (bottom) write out the binary number that best represents each length 3 code.

7. [**R06**] Consider two symbols, A and B, with the probability of occurrence of 0.8 and 0.2, respectively. The coding efficiency can be improved by combining N symbols at a time (called "symbol blocking"). Let N = 3, which would mean that you are grouping together three symbols to form a symbol block. Assume that each symbol occurrence is independent of previous symbol occurrences).

   - How many types of different outcomes are there and what are their probabilities?

- Show the arrangement of symbols on the unit interval [0, 1] and determine the arithmetic code for the three-symbol sequence.
- What is the average code word length? Is it optimum?
- How many bits are required to code the message "ABABBAABBAAABBB"?
- How could you do better than the preceding code length?

8. [**R07**] In the text, we dealt with data sources that have a finite vocabulary. For theoretical reasoning, let us say you have a source that has an infinite vocabulary $x_1, x_2, x_3 \ldots$. Let each symbol be indexed by the letter k. Furthermore, this source follows a statistical model such that the probability of occurrence for a symbol x_k is given by a probability distribution function described by the following formula for some given value of p.

$$P(x_k) = p(1 - p)^k - 1$$

- What is the entropy of the information produced by the source?

9. [**R07**] Consider a communication system that gives out only two symbols *A and B*. Assume that the parameterization followed by the probabilities are $P(A) = x^2$ and $P(B) = (1 - x^2)$.
   - Write down the entropy function and plot it as a function of x.
   - From your plot, for what value of x does the entropy become a minimum?
   - Although the plot visually gives you the value of x for which the entropy is a minimum, can you now mathematically find out the value(s) for which the entropy is a minimum?
   - Can you do the same for the maximum, that is can you find out value(s) of x for which the value is a maximum?

10. [**R06**] Consider a source that emits an alphabet with three symbols A, B, C having probability distributions as follows—P(A) = 0.625, P(B) = 0.125, P(C) = 0.25.
    - Construct a Huffman code and calculate its average code length.
    - For this three-symbol case, how many Huffman codes are possible? What are they?
    - In general, for N symbols, how many Huffman codes are possible? Justify your answer formally.
    - Is the code you have defined optimal? Give reasons. If not, what can you do to improve upon this? Show a solution by an example computing the average code length.

11. [**R08**] In this example, you will use an entropy-coding algorithm to code the English alphabet and then compare your code with a more commonly used code—the Morse code. To code the English alphabet, you will use the *Shannon-Fano* entropy-coding technique which works as follows:
    a. Arrange all the symbols in decreasing order of probability. Now split them into two groups with approximately equal probability totals. Note, it is not always possible to break them into equal probability groups, but we can partition them in the closest possible way.
    b. Assign 0 as an initial code digit to the entries in the first group and 1 to those in the second.

c. Now recursively perform the same technique for each group ensuring that the symbols are kept in the same order. Proceed recursively until all the groups are divided.

The following table in Figure 6-19 shows an example containing eight symbols with the original binary code and the computed Shannon-Fano code.

Symbol	% probability	Binary code	Shannon-Fano code
A	25	000	00
B	25	001	01
C	12.5	010	100
D	12.5	011	101
E	6.25	100	1100
F	6.25	101	1101
G	6.25	110	1110
H	6.25	111	1111

Figure 6-19

Now you are asked to compute a Shannon-Fano code for the alphabets of the English language. The table in Figure 6-20 represents the frequency of occurrence for the letters of the English alphabet. The first column gives the letter, the second column gives the frequency of occurrence on average, and the third column gives the equivalent Morse code for that letter. Using these

Letter	% probability	Morse code	Letter	% probability	Morse code
E	13.1	.	M	2.5	--
T	10.4	-	U	2.4	..-
A	8.1	.-	G	1.9	--.
O	8.0	---	Y	1.9	...
N	7.0	-.	P	1.9	.--.
R	6.8	.-.	W	1.5	.--
I	6.3	..	B	1.4	-…
S	6.1	...	V	0.9	…-
H	5.2	....	K	0.4	-.-
D	3.7	-..	X	0.1	-..-
L	3.3	.-..	J	0.1	.---
F	2.9	..-.	Q	0.1	--.-
C	2.7	-.-.	Z	0.07	--..

Figure 6-20

statistics and the Morse code description, answer the following questions. Note: Statistics for English alphabet occurrence and Morse code can be found at a variety of sites, such as *www.askoxford.com/asktheexperts/faq/aboutwords/frequency* and *www.learnmorsecode.com/*.

- There are twenty-six letters, which requires $\lceil \log_2 26 \rceil = 5$ bits of binary. Compute entropy of English text coded as individual letters (ignore spaces between words and other grammatical details such as punctuation, capitalization, and so on).
- Using the previous distribution, compute the Shannon-Fano code for English letters. How many bits do you need for each letter in your derived code need?
- If you were to use 5 bits per letter, how many total bits would be required to send the following English message as binary? How many bits would it take using the Shannon-Fano code?

"*This book is meant to give a comprehensive understanding of multimedia*."

- What is the efficiency in both of the previous cases?
- Repeat the preceding analysis for the following translation of infrequent symbols.

"*six xmas jazzzzzzz max wax buzz xerox xeno bmw zq*"

- Now look at the Morse code, which is also based on a similar statistical distribution. Assuming a "." corresponds to the bit 0 and a "-" corresponds to the bit 1, compare the lengths of your obtained code with the Morse code. Which is better? Can you get close to the Morse code using some other entropy-coding technique?
- What is the average code length of Morse codes? Is this greater or less than the entropy? Can you explain why it is so?

12. **[R06]** The following sequence of real numbers has been obtained sampling a signal:

5.8, 6.2, 6.2, 7.2, 7.3, 7.3, 6.5, 6.8, 6.8, 6.8, 5.5, 5.0, 5.2, 5.2, 5.8, 6.2, 6.2, 6.2, 5.9, 6.3, 5.2, 4.2, 2.8, 2.8, 2.3, 2.9, 1.8, 2.5, 2.5, 3.3, 4.1, 4.9.

This signal is then quantized using the interval [0,8] and dividing it into 32 uniformly distributed levels.

- What does the quantized sequence look like? For ease of computation, assume that you placed the level 0 at 0.25, the level 1 at 0.5, the level 2 at 0.75, the level 3 at 1.0, and so on. This should simplify your calculations. Round off any fractional value to the nearest integral level.
- How many bits do you need to transmit it?
- Suppose you need to encode the quantized output using DPCM. Compute the successive differences between the values—what is the maximum and minimum value for the difference? Assuming that these are your range, how many bits are required to encode the sequence now?
- What is the compression ratio you have achieved?
- Instead of transmitting the differences, you use Huffman-coded values for the differences. How many bits do you need now to encode the sequence?
- What is the compression ratio you have achieved now?

13. [**R04**] In the text, we explain lossy coding techniques such as transform coding, subband coding, vector quantization, and so on. Answer the following question regarding lossy techniques:
    - Mathematical transformations are normally irreversible, yet the transform coding techniques are categorized as lossy. What causes the lossiness?
    - Why is block-based transform coding used compared with transforming the entire signal?
    - What problems do you see with block-based coding, if any?
    - Subband coding and subsampling lead to compression, but how do the two processes differ?

14. [**R07**] In this exercise, we will attempt to use vector quantization and compute the optimal code vector length for a minimum bit rate, given that your codebook length size is fixed. Consider an image of size $N \times N$ pixels with r bits per pixel. Assume we construct a codebook with N_c code vectors. Let each code vector be a uniform $p \times p$ matrix. Answer the following questions to prove that the optimal value of p for minimum bit rate is

$$\mathrm{p} = [(N^2 log N_c)/(rN_c)]^{1/4}$$

    - What is the size of the codebook?
    - How many vectors are there in the image? What is the size of the compressed image?
    - The data that needs to be sent to the decoder includes both the compressed image and the codebook. What is the bit rate R generated using vector quantization here?
    - Now compute the value of p for the minimum R. The condition of this is $\mathrm{dR/dP} = 0$.
    - If you have an image of 512×512 with 24 bits per pixel, plot a graph for *Nc* with *p*. *Nc* is on the x-axis. This will show what p should be as your codebook size increases. Conclude that for powers of 2 codebook sizes, it is normally optimal to have 4×4–8×8 code vector sizes.

15. [**R05**] Differential PCM works by computing differences between successive samples and transmitting the differences rather than the signal itself. Furthermore, DPCM is deemed a lossy scheme when the differences are quantized prior to transmitting them. Each quantized difference introduces an error $e(n) = d(n) - \hat{d}(n)$, where $\hat{d}(n)$ is the quantized value of $d(n)$, and $d(n) = y(n) - y(n-1)$.
    - Considering open loop DPCM, if the error at each stage n is $e(n)$, how does $e(n)$ relate to the errors in the previous stages? That is, compute $e(n)$ in terms of $e(n-1)$, $e(n-2)$ .. $e(1)$.
    - Repeat the same for closed loop DPCM. That is, compute $e(n)$ in terms of $e(n-1)$, $e(n-2)$ .. $e(1)$ in terms of the closed loop case.
    - If the differences $d(n)$ were not computed as $d(n) = y(n) - y(n-1)$, but rather as shown here, how do your results change?

$$d(n) = y(n) - A, \text{ where A is the average value of all } y(i)$$

    - What practical problems do you see with using the average values?

PROGRAMMING ASSIGNMENTS

16. **[R06]** In this programming exercise, you will implement a simple run length coder that works on image data or any 2D array of values by first quantizing the input values and then applying run length coding on them. The quantization will increase the runs of consecutive same values. Your program should take as input a basic gray image (or RGB color image), and a quantization level value. You can use/modify the sample display source code given and start from there. The main idea of the encoding process here can be described in three steps:

    - Read in the image and quantize the input array using the input quantization value.
    - Run length encode the array of quantized pixels by recording the first place where a new color is encountered and only registering information when a new color is seen.
    - Organize the resulting structure and store it efficiently.

    Assume that you are following a row order scan. For example, if we have the following 5×5 gray-level input image block with a quantization value of 10 (10 intensity levels correspond to one quantization step), where each number represents a gray-level value, then we have the following quantized and encoded outputs as shown in Figure 6-21.

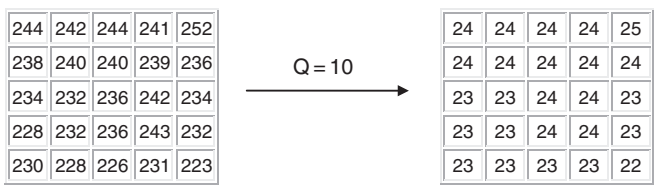


Figure 6-21

    Answer the following questions regarding the exercise.

    - For the sample input images, what compression ratios do you get on average?
    - Do your compression ratios increase or decrease with the quantization value?
    - Suppose you were to store the compressed outputs and later read them and process them. You can, of course, decompress them and represent them as an $m \times n$ image array. But, is there any other more efficient way to represent them?
    - What advantages and disadvantages do you see with your representation?

17. **[R06]** In this exercise, you will apply DPCM techniques to compress video. Although this is not the most efficient way to compress video, it should introduce you to video compression that is soon to follow in next chapters.

Video can be defined by a sequence of frames. From frame to frame there is a great deal of coherency in the data—all the pixels do not change from frame to frame, for example, the background pixels. This is the fact you will exploit in this exercise. Here, you are asked to write a program that will read a sequence of video frames as input along with a quantization parameter. Your task is to implement a DPCM like algorithm that leaves the first frame as is, but computes the differences between successive frames and quantizes the pixel value of these differences depending on the quantization value taken as input. Therefore, your program should do the following:

- Read in a video sequence.
- Compute successive frame differences—pixel for pixel.
- The differences are bound to have less entropy—quantize the values depending on the input quantization factor.

When writing out the compressed file, write the first frame as is and then the quantized pixel differences for each frame. Answer the following questions about this exercise:

- What compression ratio you are getting for your data set?
- For the same video, input all the frames to the previous program and compute the combined compression ratio. Which performs better? Why?
- Because this algorithm does do the job of compression and works very fast, what problems do you predict when using it in a real scenario? (Assume bandwidth is not an issue, that is, the algorithm is enough to compress down to the required bandwidth you have.)

18. [**R06**] Here, you are going to write an encoder and decoder to simulate a simple dictionary coding algorithm as discussed in the chapter. The program should support the following set of features:

    - As input, you should be able to take any user-defined character string from the input terminal or by reading a file. Assume that all symbols are 8-bit characters.
    - As input, you should be able to specify a dictionary size. Assume a maximum size of 256 (8 bits).
    - As output, you should get a list of indexes that corresponds to your input string and the dictionary word-index list.
    - Also output a number that shows you the compression ratio.

    You do not have to write out the compressed file, which is a little harder to do when each index is not 8 bits, for example, if it is 5 bits. In such cases, you will have to perform bit-shifting operations and concatenate the 5 least-significant bits of every index, which might be represented in your program as an int or a short. But for compression ratio computation purposes, you may assume that every index is represented by a number of bits that corresponds to the maximum entries in your dictionary (for example, 8 bits per index when the dictionary size is 256).

This page intentionally left blank

CHAPTER 7

Media Compression: Images

Visual media types such as images, video, and graphics are now ubiquitous in today's digital world. One important effect of the change is an ever-increasing demand for the access, transfer, storage, and interpretation of increasing amounts of image information, quickly and conveniently. Image compression technology is central to meeting this demand, which is manifested in many different applications. For example, the red-hot market of new and better consumer products in digital photography would not have been possible without image compression technologies. Consumer cameras touting 5 megapixels (=15 MB of raw RGB data) per image would not be available with a standard 256 MB onboard memory without the use of compression technologies. Most cameras compress images using the JPEG standard, where the average compressed size of a 5 megapixel image goes down to 300 Kb. The GIF image compression standard initially made it viable to transfer images on the Internet. Images compressed using JPEG or GIF are now pervasive in high-bandwidth applications such as digital libraries and the Internet and even low-bandwidth applications such as cellular network–based communications. Software technologies that help postprocess images and perform imaging operations are also commonplace today. All these make it easy to manipulate and transfer compressed images. In Part I of this book, we enumerated commonly used formats, and also saw the use of images in a variety of complex multimedia authored content where combinations of media types are used—such as digital images inserted as advertisements superimposed on video. Images are now routinely used in their compressed formats, such as JPEG, GIF, PNG, and TIFF for a variety of such multimedia tasks.

Furthermore, with the ever-increasing resolution of images in many applications areas—such as satellite imaging, geographic information systems, medical imaging, entertainment, and so forth—more powerful compression algorithms than the current JPEG standards will be necessary to meet consumer and industrial demands. This chapter is devoted to explaining the state of the art in image compression and the technical issues involved in compression of images. To understand this chapter, you need a thorough understanding of image representation, color, and frequency domain issues, which were presented in Chapters 2, 3, and 4. We start by discussing the transmission and storage requirements for commonly used image sizes with and without compression. Section 1 explains the redundancy in image information and how that can be exploited by compression algorithms. Section 2 describes generic classes of image compression techniques. A few of these algorithms are selected for detailed explanations because of their popular use and their adoption into the ISO/ITU standards, such as JPEG and JPEG 2000.

The need for compressing images should be clear by now. Figure 7-1 provides disk memory requirements and transmission times across different bandwidths for a variety of consumer- and industry-used image formats.

From Figure 7-1, it should be clear that compression techniques that produce compression ratios from 20:1 to 30:1 and even more are necessary to make image storage and distribution feasible for consumer-level and industry-grade applications.

Multimedia image data	Grayscale image	Color image	HDTV video frame	Medical image	Super High Definition (SHD) image
Size/duration	512 × 512	512 × 512	1280 × 720	2048 × 1680	2048 × 2048
Bits/pixel or bits/sample	8 bpp	24 bpp	12 bpp	12 bpp	24 bpp
Uncompressed size (B for bytes)	262 KB	786 KB	1.3 MB	5.16 MB	12.58 MB
Transmission bandwidth (b for bits)	2.1 Mb/image	6.29 Mb/ image	8.85 Mb/ frame	41.3 Mb/ image	100 Mb/ image
Transmission time (56 K modem)	42 seconds	110 seconds	158 seconds	12 min.	29 min.
Transmission time (780 Kb DSL)	3 seconds	7.9 seconds	11.3 seconds	51.4 seconds	2 min.

Figure 7-1 Examples showing storage space, transmission bandwidth, and transmission time required for uncompressed images

1 REDUNDANCY AND RELEVANCY OF IMAGE DATA

An overview of both lossless and lossy compression techniques was given in the previous chapter. Image compression techniques can be purely lossless or lossy, but good image compression techniques often work as hybrid schemes, making efficient use of lossy and lossless algorithms to compress image data. These schemes aim to get compression by typically analyzing the image data according to two important aspects:

- *Irrelevancy reduction*—In many cases, information associated with some pixels might be irrelevant and can, therefore, be removed. These irrelevancies can be further categorized as *visual irrelevancy* and *application-specific irrelevancy*. Information can be deemed visually irrelevant when the image sample density exceeds the limits of visual acuity given the display/viewing conditions. Application-specific irrelevance occurs when an image or image region might not be required for the application. For example, in medical and military imaging, regions irrelevant to diagnosis and analysis can be heavily compressed, or even excluded.
- *Redundancy reduction*—The image signal, like any other signal, has statistical redundancy because pixel values are not random but highly correlated, either in local areas or globally. The repeated use of pixels can be statistically analyzed by entropy-coding techniques to reduce the amount of bits required to represent groups of pixels.

The first point deals with the removal of image data using a lossy scheme due to the data's lack of relevancy either in perception or importance for the required application. This is achieved by applying quantization in the spatial and/or frequency domains. The lossy quantization effects often introduce a distortion, which must be minimized, so as to go either unnoticed or be perceptually acceptable by the human visual system (HVS). The lossy process reduces the entropy in the data, which leads to a greater amount of redundancy in the distribution. This redundancy is further reduced by lossless or entropy-coding techniques, which use statistical analysis to reduce the average number of bits used per pixel. Although there are standards such as GIF, TIFF, and PNG that use lossless coding techniques only, most commercially used standards that require higher compression use both lossy and lossless techniques in combination.

A common characteristic of most images is that the neighboring pixels are correlated. This correlation arises because the image is not a random collection of pixels, but rather a coherent structure composed of objects and similar areas. Additionally, if pixels change, the change is mostly gradual and rarely sudden. This similarity and slow change of pixels is seen more in local neighborhoods, as illustrated in Figure 7-2.

Figure 7-2 shows three magnified areas. The collective pixels in all regions together might not exhibit as much correlation, but when each local area is considered, there is less variance among the pixels. This local similarity or local

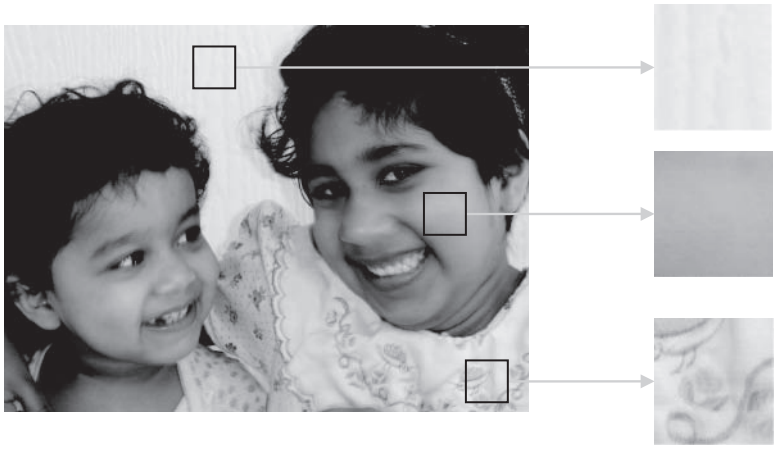


Figure 7-2 Local pixel correlation. Images are not a random collection of pixels, but exhibit a similar structure in local neighborhoods. Three magnified local areas are shown on the right.

redundancy is exploited by image compression techniques. The redundancy can be further classified as follows:

- *Spatial redundancy*—The pixels' intensities in local regions are very similar.
- *Spectral redundancy*—When the data is mapped to the frequency domain, a few frequencies dominate over the others.

There is also another kind of redundancy—*temporal redundancy*—which occurs when a sequence of images, such as video, is captured. Because this chapter focuses on still image compression only, we do not address it here, but introduce it in the following chapter on video encoding. The next section describes how spatial and spectral redundancies are exploited by image compression techniques.

2 CLASSES OF IMAGE COMPRESSION TECHNIQUES

Like all other compression techniques, image compression techniques can be generically categorized as either *lossless* or *lossy*. In lossless compression schemes, the reconstructed image, after compression, is numerically identical to the original image. The classes of lossless techniques mainly exploit spatial and statistical redundancies. The entropy-coding techniques introduced in the previous chapter are, in theory, sufficient to fully exploit the statistical redundancy. In practice, however, these techniques by themselves do not achieve sufficient compression. The lossy schemes normally operate by quantization of image data either directly in the pixel domain or in the frequency domain obtained via different transformations. An overall taxonomy is shown in Figure 7-3. Also shown in bold are popular standards based on the respective algorithms. Several selected techniques among these are discussed in more detail in the following few sections.

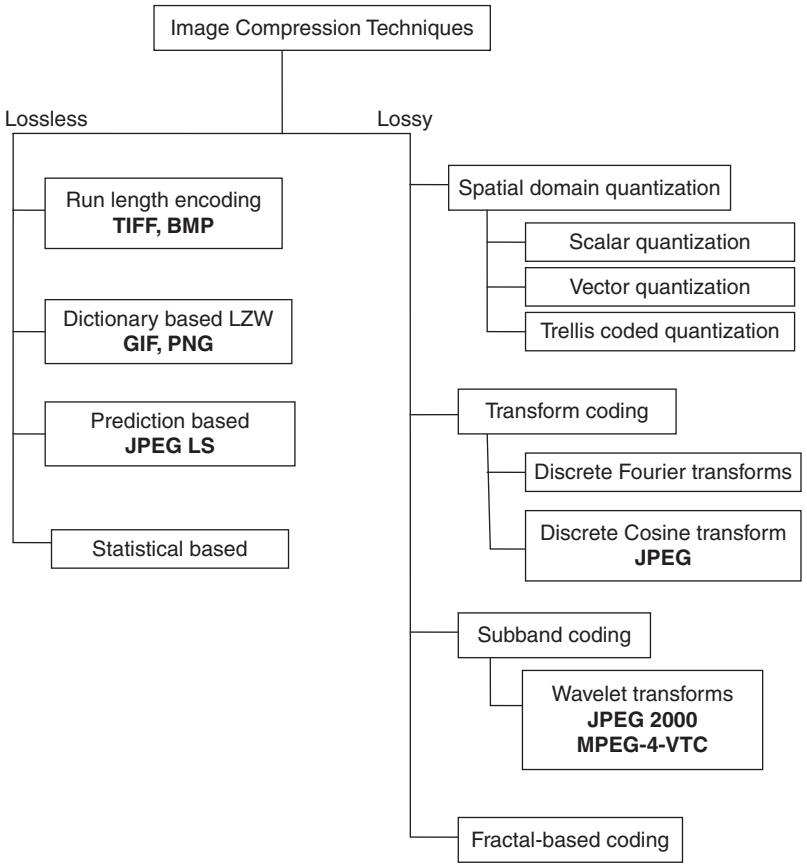


Figure 7-3 Taxonomy of image compression techniques. The left column shows lossless categories and the right column shows lossy categories. Commonly used standards based on the techniques are shown in bold.

3 LOSSLESS IMAGE CODING

The first compression schemes were designed to compress digital images for storage purposes, and were based on lossless coding techniques. Compared with lossy techniques, purely lossless compression can only achieve a modest amount of compression, so their use is somewhat limited in practice today. However, certain applications do need to store images in lossless form—for example, the film industry renders images and the original image sequences are always stored in lossless form to preserve image quality. These images are used to make films, which are distributed to theaters. The lossless compressed images can be further adopted for specific markets where the images might undergo quality degradation by a lossy scheme—such as digital distribution via DVDs and usage in digital projectors. Lossless image compression schemes correspond to the BMP, TIFF, GIF, and PICT formats, among others.

3.1 Image Coding Based on Run Length

Run length encoding (RLE) is probably one of the earliest lossless coding schemes that was applied to images, and was incorporated in the earlier formats such as BMP, TIFF, and RLA on PCs and PICT on Macintosh machines. As described in the previous chapter, this method simply replaces a run of the same pixel by the pixel value or index along with the frequency. Although better schemes than BMP and RLA exist, this is still used sometimes for compatibility of formats between various imaging software. TIFF (Tagged Image File Format) was designed to be extensible by supplying the compression scheme used in the header, so TIFF can be adjusted to make use of RLE or any other supported compression algorithm.

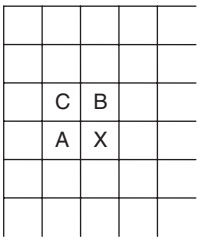
3.2 Dictionary-Based Image Coding (GIF, PNG)

Dictionary-based coding algorithms, such as LZW, work by substituting shorter codes for longer patterns of pixel values occurring in an image. A dictionary of code words and indexes is built depending on the pixel patterns occurring in the image. This technique is used by both GIF and PNG standards. However, one difference is that instead of using the LZW algorithm on the initial pixels in the image, it is performed on an indexed set of colors. When compressing using GIF, you normally supply a palette size, for example, 8 bits, which corresponds to a maximal number of 256 colors. The best 256 colors are then chosen as the palette colors and the image pixel values are replaced by indexes that best represent the original pixel values from the palette. The LZW algorithm is then applied to the indexes. The GIF format was initially developed by the UNISYS Corporation and was one of first compression formats that made image transfer on the Internet possible.

3.3 Prediction-Based Coding

Standard prediction techniques such as DPCM or its variants modified to 2D can be used to predict pixel values in the neighborhood. The errors between the actual values and their predictions can then be entropy coded to achieve compression. The lossless mode of JPEG works by using a predictive scheme that exploits redundancy in two dimensions. Given a pixel value X, a prediction mechanism based on the previously used neighborhood pixels is used to predict the value of X. The difference D(i,j) between the original and predicted value is computed. Repeating this process for all the pixels results in an *error image*, which has lower entropy than the original image and can be effectively entropy coded. The process is illustrated in Figure 7-4, where a pixel with value X is shown, and the predicted value obtained from the neighbors A, B, and C depends on a prediction rule, which might be one-dimensional or two-dimensional.

Lossy schemes are capable of achieving much higher compression rates. In most cases, the distortion caused by lossy techniques is acceptable for the desired compression ratio. Popularly used lossy image compression techniques and standards are discussed next. In almost all cases, lossy quantization is followed by a lossless step of entropy coding based on statistical schemes such as the Huffman or arithmetic codes.



Prediction index	Prediction
0	No prediction
1	A
2	B
3	C
4	A + B − C
5	A + ((B − C)/2)
6	B + ((A − C)/2)
7	(A + B)/2

Figure 7-4 Lossless coding process based on predictions in the JPEG standard. The index shows how the value of X is predicted by a combination of A, B, and C. Prediction indexes 1, 2, and 3 are 1D predictors, whereas 4, 5, 6, and 7 are 2D predictors.

4 TRANSFORM IMAGE CODING

The generic transform-based coding was illustrated in Figure 6-14 of the previous chapter. In addition, image compression operates by performing entropy coding after the transform computation and quantization of the transform coefficients. This is shown in Figure 7-5. The encoder proceeds in three steps: transform computation, quantization of the transform coefficients, and entropy coding of the quantized values. The decoder only has two steps: the entropy decoding step that regenerates the quantized transform coefficients and the inverse transform step to reconstruct the image. The quantization step present in the encoder is the main cause of the loss.

The transform must be a mathematically invertible function that takes the input spatial domain image to produce its frequency domain representation. Examples of these transforms include the Discrete Fourier transform (DFT) and the Discrete Cosine transform (DCT). The frequency domain representation is also an image whose pixels represent frequency domain coefficients. We now explain how the generic transform-based pipeline shown in Figure 7-5 has been adapted in the popular JPEG compression standard.

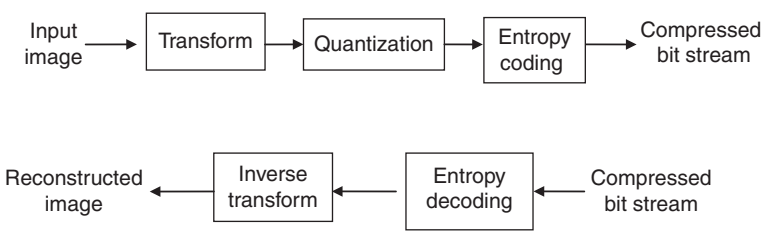


Figure 7-5 Transform-based encoding and decoding pipelines. Encoding consists of three steps: computation of the transform, quantization of the transform coefficients, and entropy coding the quantized transform coefficients. The decoding process does exactly the reverse.

4.1 DCT Image Coding and the JPEG Standard

The JPEG standard is based on a transform image coding technique that utilizes the Discrete Cosine transform (DCT). The DCT was chosen by the JPEG community because of its good frequency domain energy distribution for natural images, as well as its ease of adoption for efficient hardware-based computations. To understand how compression occurs in the JPEG pipeline, it is imperative that the DCT behavior be well understood. Section 8 explains the DCT formula and the intuitive working behind it. Additionally, Chapter 2 explains aspects of frequency domain representations for images. The entire JPEG encoder pipeline is depicted in Figure 7-6. The step-by-step descriptions show the exact data flow and evaluations that happen in *baseline mode* of JPEG compression. Besides compressing using the baseline mode, there is also the progressive mode of JPEG discussed in Section 7. All modes of JPEG compression start by transforming the image representation into a YCrCb component representation. The YCrCb color space is very similar to the YUV color space explained in Chapter 2. Each of the Y, Cr, and Cb components is treated independently by breaking it into 8 × 8 blocks. In the baseline mode, the blocks are scanned in scan line order from left to right and top to bottom, each one undergoing the DCT computation, quantization of frequency coefficients, and, finally, entropy coding.

The salient features of each step are summarized in the following list:

- Any image can be taken as input, but is always first converted to the YCrCb format to decouple the image chrominance from the luminance.
- The YCrCb representation undergoes a 4:2:0 subsampling, where the chrominance channels Cr and Cb are subsampled to one-fourth the original size. The subsampling is based on psychovisual experiments, which suggest that the human visual system is more sensitive to luminance, or intensity, than to color.
- Next, each channel (Y, Cr, and Cb) is processed independently. Each channel is divided into 8 × 8 blocks. The JPEG community chose this block size after much experimentation on natural, continuous tone images. On the average, the 8 × 8 size seems to be the most optimal area for spatial and spectral correlation that the DCT quantization can exploit. Smaller sizes increase the number of blocks in an image, and larger sizes reduce the correlation seen among pixels. If the image width (or height) is not a multiple of 8 × 8, the boundary blocks get padded with zeros to attain the required size. The blocks are processed independently.
- Each 8 × 8 block (for all the channels) undergoes a DCT transformation, which takes the image samples $f(x,y)$ and computes frequency coefficients $F(u,v)$ as explained in Section 8. The DCT computation of a sample 8 × 8 block is shown in Figure 7-7. The image $f(x,y)$ and its numerical intensity values are shown on the upper left. The middle section shows the computed DCT transform $F(u.v)$. Although the values are real numbers, the table shows the DCT values rounded to the nearest integer for compactness in depiction. The 3D plot shows the DCT values in 3D. Of these, the first coefficient, that is the coefficient in the location given by index (0,0), is normally the highest, and is called the DC coefficient. This special status for the DC coefficient is deliberate because most of the

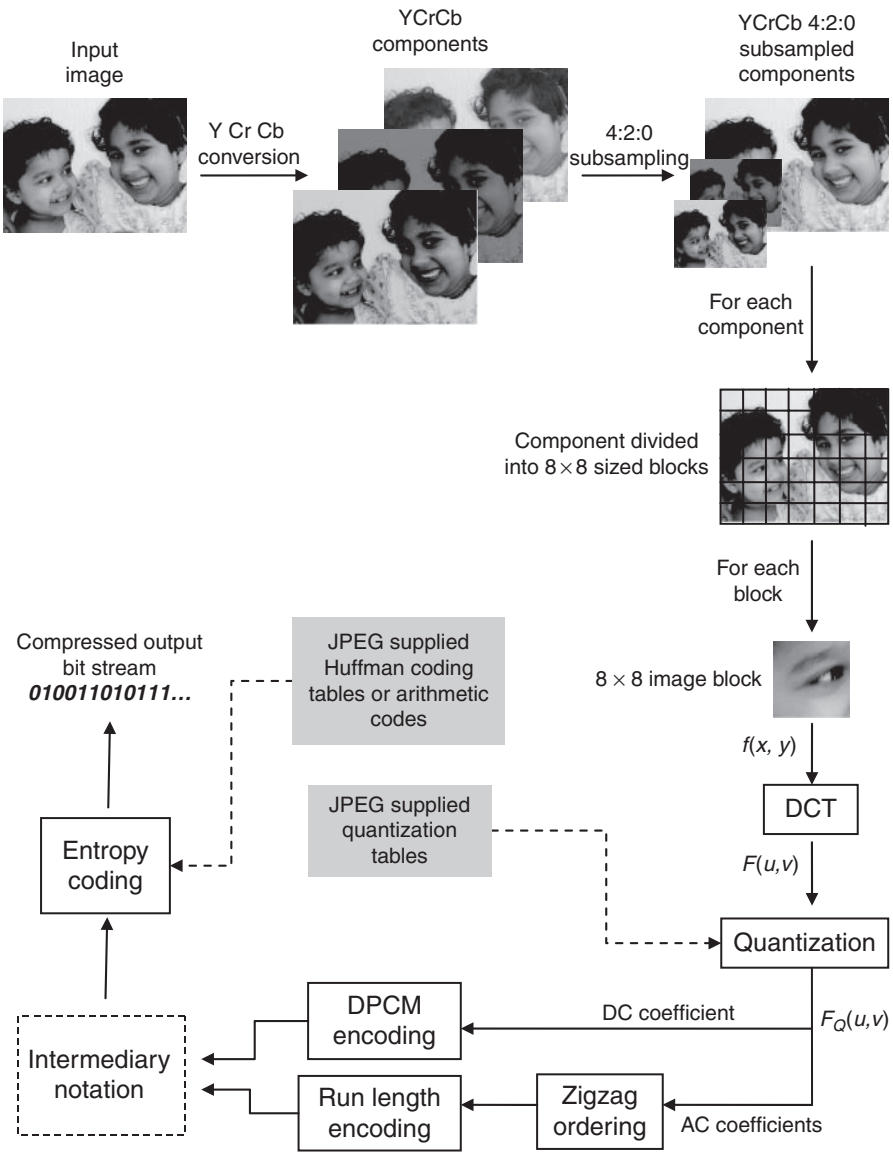
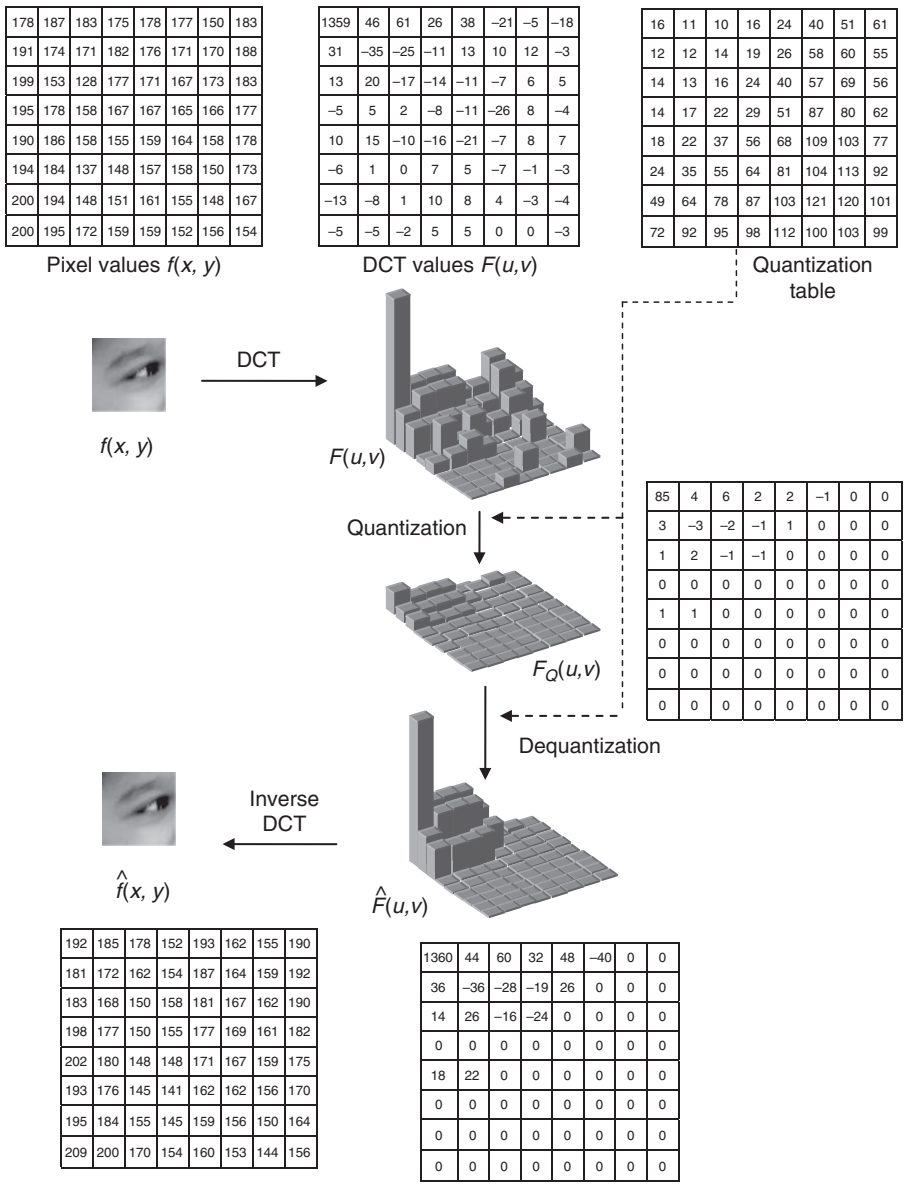


Figure 7-6 The JPEG compression pipeline. See the color insert in this textbook for a full-color version of this image.

energy in natural photographs is concentrated among the lowest frequencies. The remaining coefficients are called AC coefficients.

- Next, the DCT coefficients $F(u,v)$ are quantized using a quantization table supplied by JPEG. This table is shown in the upper-right corner in Figure 7-7. Each number at position (u,v) gives the quantization interval size for the corresponding $F(u,v)$ value. The quantization table values might appear random, but, in fact, they are based on experimental evaluations with human subjects, which have shown that



178	187	183	175	178	177	150	183
191	174	171	182	176	171	170	188
199	153	128	177	171	167	173	183
195	178	158	167	167	165	166	177
190	186	158	155	159	164	158	178
194	184	137	148	157	158	150	173
200	194	148	151	161	155	148	167
200	195	172	159	159	152	156	154

1359	46	61	26	38	–21	–5	–18
31	–35	–25	–11	13	10	12	–3
13	20	–17	–14	–11	–7	6	5
–5	5	2	–8	–11	–26	8	–4
10	15	–10	–16	–21	–7	8	7
–6	1	0	7	5	–7	–1	–3
–13	–8	1	10	8	4	–3	–4
–5	–5	–2	5	5	0	0	–3

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

85	4	6	2	2	–1	0	0
3	–3	–2	–1	1	0	0	0
1	2	–1	–1	0	0	0	0
0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

192	185	178	152	193	162	155	190
181	172	162	154	187	164	159	192
183	168	150	158	181	167	162	190
198	177	150	155	177	169	161	182
202	180	148	148	171	167	159	175
193	176	145	141	162	162	156	170
195	184	155	145	159	156	150	164
209	200	170	154	160	153	144	156

1360	44	60	32	48	–40	0	0
36	–36	–28	–19	26	0	0	0
14	26	–16	–24	0	0	0	0
0	0	0	0	0	0	0	0
18	22	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 7-7 DCT 8 × 8 block example. The arrays show the values of f(x,y), F(u,v), F_Q(u,v), and the decoded F_Q(u,v) and f(x,y). The frequency coefficient values are also shown depicted in 3D. The coefficient in the first location given by index (0,0) is normally the highest, whereas higher-frequency coefficients are almost zero.

low frequencies are dominant in images, and the human visual system is more sensitive to loss in the low-frequency range. Correspondingly, the numbers in the low-frequency area (upper-left corner of the table) are smaller and increase as you move toward the high-frequency coefficients in the other three corners.

Using this table, $F_Q(u,v)$ is computed as

$$F_Q(u,v) = \left\lceil \frac{F(u,v)}{Q(u,v)} \right\rceil, \textit{ where } Q(u,v) \textit{ is the value in the quantization table}$$

The computed $F_Q(u,v)$ values are shown in the center right table. After quantization, almost all the high-frequency $F_Q(u,v)$ are zero, while a few low-frequency values remain. For evaluation purposes, we also show the decoder process in Figure 7-7, which takes $F_Q(u,v)$ and dequantizes the value to produce $\hat{F}(u,v)$. The loss of data in the frequency coefficients should be pretty obvious when you compare $F(u,v)$ and $\hat{F}(u,v)$.

- The quantized coefficients $F_Q(u, v)$ are then encoded into an intermediary pattern. Here, DC $F_Q(0,0)$, which normally corresponds to the highest energy, is treated differently when compared with the other higher-frequency coefficients, called AC coefficients. The DC coefficients of the blocks are encoded using differential pulse code modulation. The AC coefficients are first scanned in a zigzag order, as shown in Figure 7-8. The quantization using the JPEG quantization table produces a lot more zeros in the higher-frequency area. Consequently, the zigzag ordering produces a longer run of zeros towards the end of the scan because the high-frequency coefficients appear at the tail end. This produces a lower entropy of scanned AC coefficients, which are run length encoded. Both the DPCM codes of DC coefficients and the run length coded AC coefficients produce an intermediary representation.
- The next step is to entropy code the run of DC and AC coefficients. Prior to entropy coding, these coefficients are converted into intermediary representations. For the representation of the DC coefficient, it is first DPCM coded with the previous block's DC value and the difference is represented as a 2-tuple, which

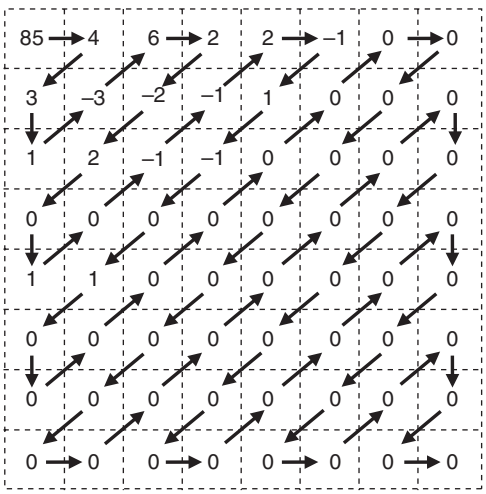


Figure 7-8 Zigzag ordering of quantized AC coefficients. The resulting sequence has a much longer run of zeros and, hence, can be more efficiently entropy coded.

shows the *size* in bits used to encode the DPCM difference and the *amplitude* of the DPCM difference. In our example, the quantized DC value of the block is 85. Assuming that the DC value of the previous block was 82, the DPCM difference is 3, which needs two bits to encode. Therefore, the intermediary notation of the DC coefficient is <2><3>. This representation is illustrated in Figure 7-9.

For AC coefficients, the intermediary notation is used only for the nonzero AC coefficients. Each nonzero AC coefficient is again represented by two symbols. The first symbol here represents two pieces of information—*runlength* and *size*. The *runlength* is the number of consecutive nonzero AC coefficients that precede it in the zigzag sequence and the *size* is the number of bits used to encode the amplitude of the AC coefficient. The second symbol encodes the amplitude of the nonzero AC coefficient. In our example, the first nonzero AC coefficient has a value of 4 and there are no zeros preceding it, so the run length is 0. The amplitude of 4 needs 3 bits (as per the JPEG codes) to encode. Therefore, the intermediary representation is <0,3> <4>. Figure 7-9 shows the intermediary notation for all the nonzero AC coefficients in our example.

- The intermediary representations are then entropy coded using codes supplied by the JPEG organization. The first symbol in the intermediary representation for both DC and AC coefficients is encoded using Huffman coding, where the Huffman codes supplied by JPEG are based on statistical analysis of the frequency of these intermediary notations. The second symbol, which is the amplitude, is encoded using variable length integer code that are not prefixed. The table in Figure 7-9 enumerates the codes for the first and second symbols used in our example. The first symbol has a Huffman code and this is a variable-length code (VLC), whereas the second symbol's code is a variable length integer code (VLI). The difference is that VLCs are prefixed and uniquely decodable, whereas the VLIs are not and, consequently, are shorter in length compared with VLCs. Note that the decoder has to decode the VLC (which is unique), and from the VLC, it can decode the number of bits that correspond to the next VLI and, hence, decode the amplitude appropriately.

Figure 7-10 illustrates results of the JPEG algorithm at different compression rates. You can see that as the compression ratios go down to more than 50:1, the distortion gets unacceptable.

4.2 JPEG Bit Stream

The previous subsection illustrated the JPEG algorithm where the image is divided into blocks and each block is encoded in a scan line manner. The resultant compressed bit representation was also derived for a sample block. The compressed bit representations of all the scanned blocks are packed and organized to form a bit stream as per the JPEG specification standard. Figure 7-11 shows a hierarchical view of this standardized representation. The first level shows a header and tail to represent the start and end of the image. The image here is termed as a *frame*. Each frame is organized as a combination of *scans* or components of the frame. Each scan is further organized as a group of *segments*. A segment is a series of 8 × 8 blocks.

DC coefficient representation:

symbol -1	symbol -2
<SIZE>	<AMPLITUDE>

AC coefficient representation:

symbol -1	symbol -2
<RUNLENGTH, SIZE>	<AMPLITUDE>

Intermediary stream

<2><3> <0,3><4> <0,2><3> <0,1><1> <0,2><-3> <0,3><6>
<0,2><2> <0,2><-2> <0,2><2> <1,1><1> <1,1><-1> <0,1><-1>
<0,2><2> <0,1><-1> <0,1><1> <0,1><-1> <1,1><1> EOB

Intermediary symbol	Binary representation of first symbol (prefixed Huffman Codes)	Binary representation of second symbol (non-prefixed variable integer codes)
<2> <3>	011	11
<0,3> <4>	100	100
<0,2> <3>	01	11
<0,1> <1>	00	1
<0,2> <−3>	01	00
<0,3> <6>	100	110
<0,2> <2>	01	10
<0,2> <−2>	01	01
<0,2> <2>	01	10
<1,1> <1>	11	1
<1,1> <−1>	11	0
<0,1> <−1>	00	0
<0,2> <2>	01	10
<0,1> <−1>	00	0
<0,1> <1>	00	1
<0,1> <−1>	00	0
<1,1> <1>	11	1
EOB	1010	

Binary Stream:

01111100100011100101001001100110010101101111100000110000001000111101O

Figure 7-9 Intermediary and coded representations of DC and AC coefficients. The figure shows four items. The first item shows the DC and AC coefficient representation format. The stream incorporating intermediary representations of all coefficients is shown next. The first two symbols in the stream correspond to the DC coefficient, whereas the rest correspond to the nonzero AC coefficients. The third item is a table illustrating the binary code for each of the symbols in the intermediary notation. Finally, the coded binary stream for the intermediary stream is shown last.

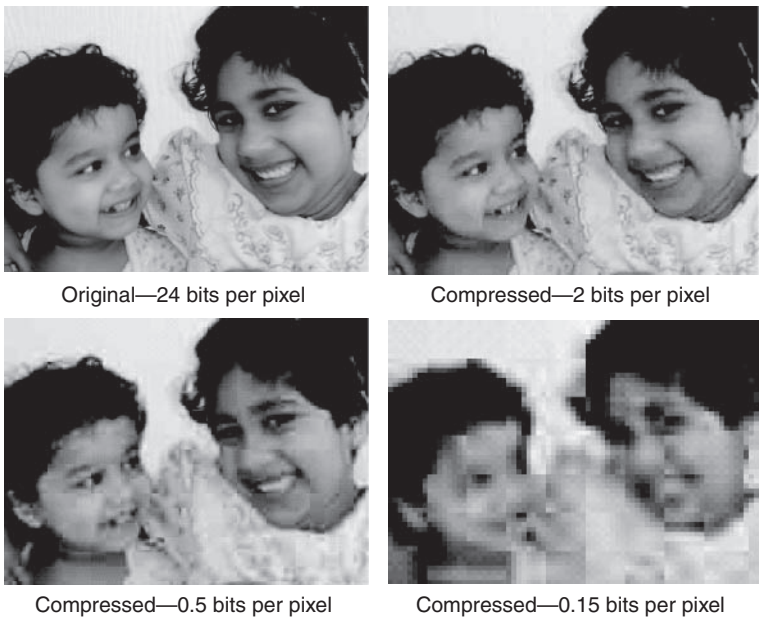


Figure 7-10 JPEG compression results. The upper-left image shows the original at 24 bits per pixel. The remaining three images show the reconstructed outputs after JPEG compression at different compression ratios. The blocky artifacts increase at lower bit rates.

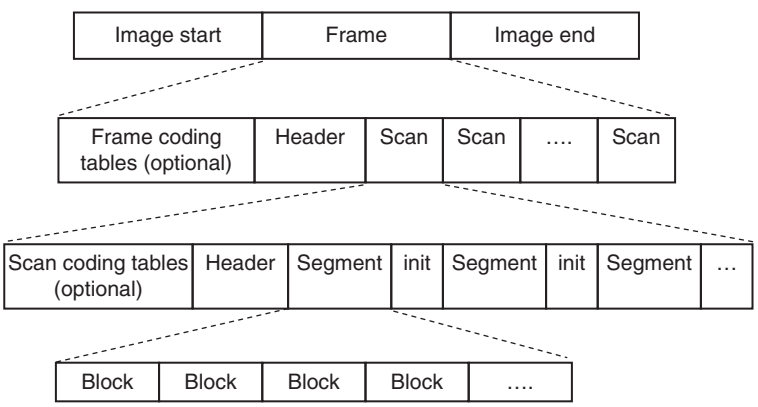


Figure 7-11 Hierarchical representation of the JPEG bit stream.

4.3 Drawbacks of JPEG

The JPEG compression standard has been successfully adopted since 1993 as a milestone in image compression standards. Besides being at the state of the art in research at that time, the adoption was mainly facilitated by its royalty-free implementation/usage, and it produced decent quality for the storage and transmission needs at the time. Also, it has an acceptable computational complexity, and the DCT computation can easily be implemented in hardware for devices that involve image capture. However, today's

needs are far greater than the provided functionality of JPEG, such as better compression requirements for larger imagery in a variety of applications, processing needs on compressed image bit streams, and so on. These drawbacks are summarized in the following list. Some of these drawbacks are because of the DCT artifacts, although others lie in the way the bit stream is put together.

- *Poor low bit-rate compression*—JPEG offers excellent rate-distortion performance in the mid and high bit rates, but at low bit rates, the perceived distortion becomes unacceptable.
- *Lossy and lossless compression*—There is currently no standard that can provide superior lossless and lossy compression in a single coded stream.
- *Random access of the bit stream*—JPEG does not allow random access because each of the 8×8 blocks is interdependent. This prevents certain application scenarios where the end user might want to decode and see only a certain part of the image (or region of interest).
- *Large image handling*—JPEG does not allow for the compression of images larger than 64K by 64K without tiling.
- *Single compression architecture*—The current JPEG standard has about 44 modes. Many of these are application specific and not used by the majority of decoders.
- *Transmission in noisy environments*—JPEG was created before wireless communications became an everyday reality; therefore, it does not acceptably handle such an error-prone channel.
- *Computer-generated images and documents*—JPEG was optimized for natural images and does not perform well on computer-generated images and document imagery. This is because JPEG is well suited to continuous tone imagery but not constant tone or slow changing tone imagery.

Consequently, the ISO community set forth a new image coding standard for different types of still images (bilevel, grayscale, color, multicomponent), with different characteristics (natural, scientific, remote sensing, text rendered graphics, compound, etc.), allowing different imaging models (client/server, real-time transmission, image library archival, limited buffer and bandwidth resources, etc.) preferably within a unified and integrated system. This is the JPEG 2000 standard, which is based on the wavelet transform, as explained in the next section.

5 WAVELET BASED CODING (JPEG 2000)

The JPEG 2000 compression pipeline makes use of the Discrete Wavelet transform (DWT) to compress images. The data flow of JPEG 2000 is similar to transform coding flow and is depicted in Figure 7-12. Central to the whole encoding process in JPEG 2000

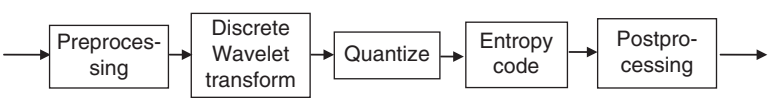


Figure 7-12 The JPEG 2000 pipeline

is the Discrete Wavelet transform, which is known to work better than the Discrete Cosine transform in the way it distributes energy among the frequency coefficients. In JPEG, the DCT works on individual 8 × 8 blocks, which results in "blockiness" at higher compression ratios. In contrast, the DWT in JPEG 2000 converts the whole image into a series of wavelets, which can be stored more efficiently than 8 × 8 pixel blocks. We now discuss each of the subprocesses in the pipeline.

5.1 The Preprocessing Step

The preprocessing stage is responsible for tiling, conversion into the YCrCb formats, and level offsetting. The preprocessing steps, all of which are not compulsory, are performed prior to applying the Discrete Wavelet transform to the image.

- The *tiling process* partitions the image into rectangular, but equal-sized and nonoverlapping blocks. Each tile is then independently processed for DWT analysis, quantization, entropy coding, and so on. The tiling process is purely optional and is done only to reduce memory requirements, as needed to deal with very large images. Additionally, because each tile is encoded independently, they can also be decoded independently. The tiling process is shown in Figure 7-13.
- The YCrCb conversion process is similar to the JPEG pipeline. It is mainly done to take advantage of the human visual system's diminished tolerance to chrominance when compared with luminance. Each of the Y, Cr, and Cb channels are independently processed afterward, with different tolerances used for Y, Cr and Cb.

Figure 7-13 Sample tiling on an image. The tiles are square with nonoverlapping areas. The tiles at the boundary might not contain all the image data. In such cases, the out-of-range area is padded with zeros.

- The level offsetting process refers to shifting the DC levels. Before the DWT can be applied to the image (or tiles), the image (or tiles) is DC level shifted by subtracting a constant value from each pixel value. This is done primarily for the DWT to operate properly, which requires that the dynamic range of the pixel values be centered about zero. For an n bit representation, the output of this stage ensures that the values range from It should be $-2^{(n-1)}$ to $2^{(n-1)}-1$. The level shifting is also used in region of interest coding, as explained in the next few sections.

5.2 The Discrete Wavelet Transform

JPEG 2000 uses the Discrete Wavelet transform (DWT) to decompose the incoming image signal (or an individual tile) into its high and low subbands at every stage. One such stage is shown in Figure 7-14, where the input signal is passed through a low-pass filter and a high-pass filter. This results in a low-band output and a high-band output. For displaying results, which explain the wavelet transform, we have converted the original image used to a square image, but the processes work just as well for rectangular images with tiling. The low-pass output has the same number of samples as the input image, but the image signal does not contain any high frequency. Similarly, the high-pass output also contains the same number of samples as the input, but without any low-frequency content. Thus, the number of samples at each stage gets doubled and, therefore, the output of each filter is downsampled by a factor of 2 to keep the total number of samples at each stage's output the same as the input.

Although Figure 7-14 shows the general process on two-dimensional data, all the wavelet transforms used in JPEG 2000 are applied in one dimension only. A two-dimensional transform is achieved by applying one-dimensional transforms along two

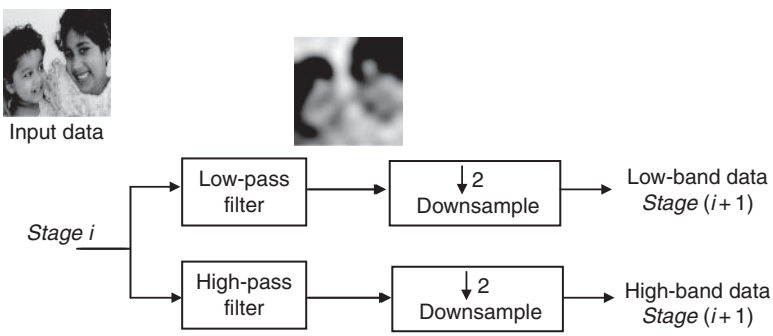


Figure 7-14 The basic Discrete Wavelet structure. Any input data at stage i is passed through a high- and low-frequency filter to produce two outputs. Each contains the same data samples as the input but has either the low or high frequencies removed. The resulting filter outputs are then subsampled to reduce the number of samples by a factor of 2. The combined samples of the outputs are now equal to the number of samples at the input.

orthogonal dimensions. This is better depicted in Figure 7-15, where first a row 1D wavelet transform is applied and the output subsampled by 2, and then the output of the first 1D transform stage undergoes a column 1D wavelet transform to create four bands as shown. All the channels are independently processed, but we show the output on the luminance channel only.

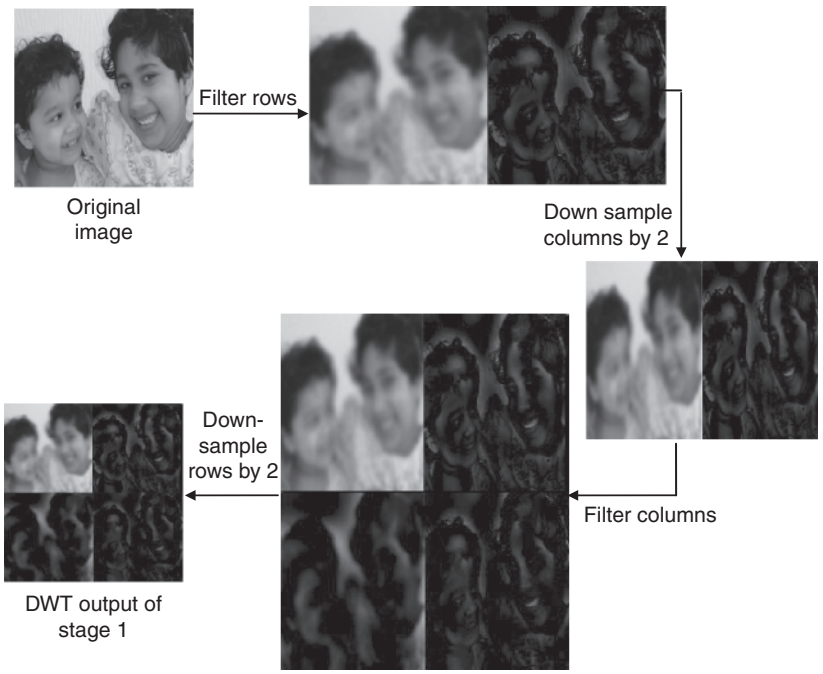


Figure 7-15 DWT process for the Y component of the sample image used.

In JPEG 2000, every original input image signal goes through multiple stages of the DWT. The number of stages performed depends on the implementation. Two stages are demonstrated in Figure 7-16; although all the Y, Cr, and Cb are independently processed, here we show the output together. The first stages output has four quadrants:

- *LL*—Low subbands of the filtering in both dimensions, rows and columns
- *HL*—High subbands after row filtering and low subbands after column filtering
- *LH*—Low subbands after row filtering and high subbands after column filtering
- *HH*—High subbands after both row and column filtering

The second stage repeats the same process with the LL subband output of the previous stage. The higher subbands (HL, LH, and HH) hardly contain any significant samples and, therefore, only the LL subband is further transformed. Although JPEG 2000 supports from 0 to 32 stages, usually 4 to 8 stages are used for natural images.

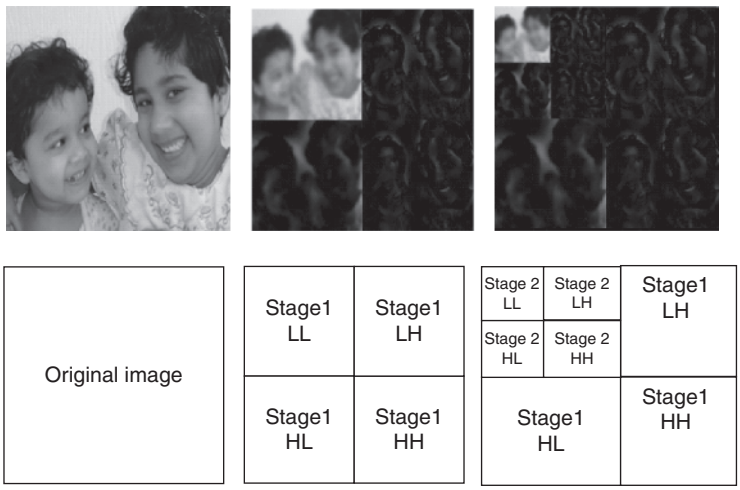


Figure 7-16 Original input and the output of discrete wavelet processing at the first two levels. The top row shows the imaged outputs. The bottom row shows what each block at a level contains.

Each component (or the tiles in each component) is decomposed using the DWT into a series of decomposition levels. Each level contains a number of subbands. These subbands contain frequency coefficients that describe the horizontal and vertical frequency characteristics of the original input. These coefficients are quantized to reduce precision, which causes a loss. The quantized coefficients are next reorganized into packets where each packet contains the quantized coefficients at a particular resolution. This results in the first packets containing low-quality approximations of the image, with each successive packet containing frequency coefficient information that improves the quality of the decoded output. The packets are organized into blocks and entropy coding is performed on each block independently.

5.3 JPEG 2000 Versus JPEG

The DWT used in JPEG 2000 is more state of the art and provides a better compression ratio when compared with the DCT used in JPEG. Besides that, the organization of the compressed bit stream in JPEG 2000, which included bit plane coding, packetization, and organization of packets into blocks, provides far more powerful features that had not been present in other standards prior to it. Examples of these features include the following:

- *Encode once—platform-dependent decoding*—In JPEG 2000, the encoder decides the maximum resolution and image quality to be produced—from the highly compressed to completely lossless. Because of the hierarchical organization of each band's coefficients, any image quality can be decompressed from the resulting bit stream. Additionally, the organization of the bit stream also makes

it possible to perform random access by decompressing a desired part of the image or even a specific image component. Unlike JPEG, it is not necessary to decode and decompress the entire bit stream to display section(s) of an image or the image itself at various resolutions. This is powerful for practical scenarios, as shown in Figure 7-17. The figure depicts how you can encode an image once with JPEG 2000 and decode it at various frequency resolutions and spatial resolutions depending on the bandwidth availability or the display platform.

- *Working with compressed images*—Normally, imaging operations such as simple geometrical transformations (cropping, flipping, rotating, and so on) and filtering (using the frequency domain) are performed on the pixel domain representation of the image. If the image is compressed, it is decompressed, and recompressed after the required operation. However, with JPEG 2000, such operations can be applied to the compressed representation of the image.

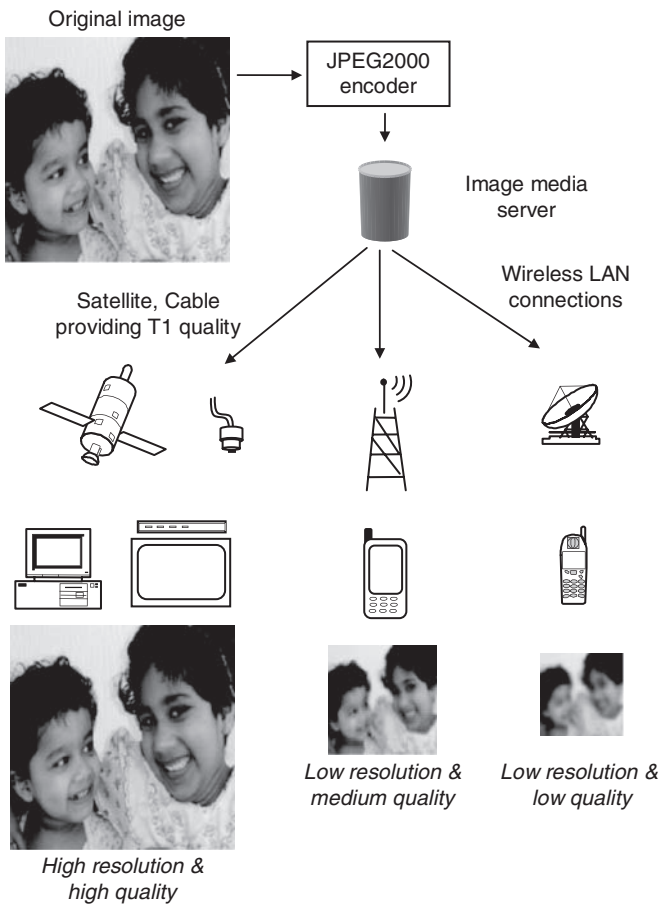


Figure 7-17 A practical application of JPEG2000. The image is encoded once using the DWT, but various end terminals can receive different resolutions from the same bit stream.

- *Region-of-interest encoding*—Region-of-interest (ROI) coding pertains to coding a specific region with higher quality compared with the rest of the image. An example of this is shown in Figure 7-18. This process can be predefined, or can change dynamically. Predefined ROIs are normally set at encoding time, whereas dynamic ROIs are used in applications where specific regions of the image are chosen for decoding at finer resolutions compared with other parts of the image. JPEG 2000 elegantly allows for ROI control by specifying a parameter known as an ROI mask. The ROI mask informs the encoder (if static selection) or the decoder (if dynamic selection) about the range of wavelet coefficients that contribute to a region's reconstruction. These coefficients can be decoded first, before all the background is decoded.

Figure 7-18 Region of interest (ROI). Using JPEG 2000 ROI coding, specific regions can be encoded or decoded with higher resolutions compared with the rest of the image.

6 FRACTAL IMAGE CODING

The DCT and DWT transform-based techniques described previously work by transforming the image to the frequency domain and minimizing the distortion (via appropriate quantization) given a bit rate. Fractal image compression techniques work by attempting to look for possible self-similarities within the image. If aspects of the image can be self-predicted, the entire image can be generated using a few image seed sections with appropriate transformations to create other areas of the image. The fundamental operation here is to find the seed areas, and to find transformations to predict other areas. In theory, this can lead to very good compression ratios, but practical implementations have shown the process of finding seeds and transformations to be computationally intractable. A basic understanding of fractals is required to comprehend fractal-based compression.

6.1 Fractals

To understand fractals informally, you may think of them as being self-similar geometrical objects, where parts of the object look very similar to the entire object. This similarity is not accidental, but is because of the process that creates it. In case of an image, a fractal is defined by a seed image(s) and a recursive transformation function(s), which at each level takes the original image and transforms it to a new image. A standard example given is an image generated by the Sierspinsky triangle. Here, the seed can be any image and the transformation is a scaling followed by copying for the initial image in a specific way, as shown in Figure 7-19.

There are two important aspects that can be seen from Figure 7-19:

- Whatever initial seed image is used, the final resulting fractal structure is very similar.
- At every iteration, more detail is added.

The first point is very important because it suggests that after a certain number of iterations, the final image always converges to a fixed image no matter what the beginning image was. This structure is purely determined by the transformation that is recursively applied at each level or iteration. In the examples shown in Figure 7-19, this transformation corresponds to a simple scaling of the image at the previous iteration to 50% and placing three copies of the scaled function in the form of a triangle. The transformation, therefore, completely determines the fractal, and the final resulting image is called the *attractor* of the process.

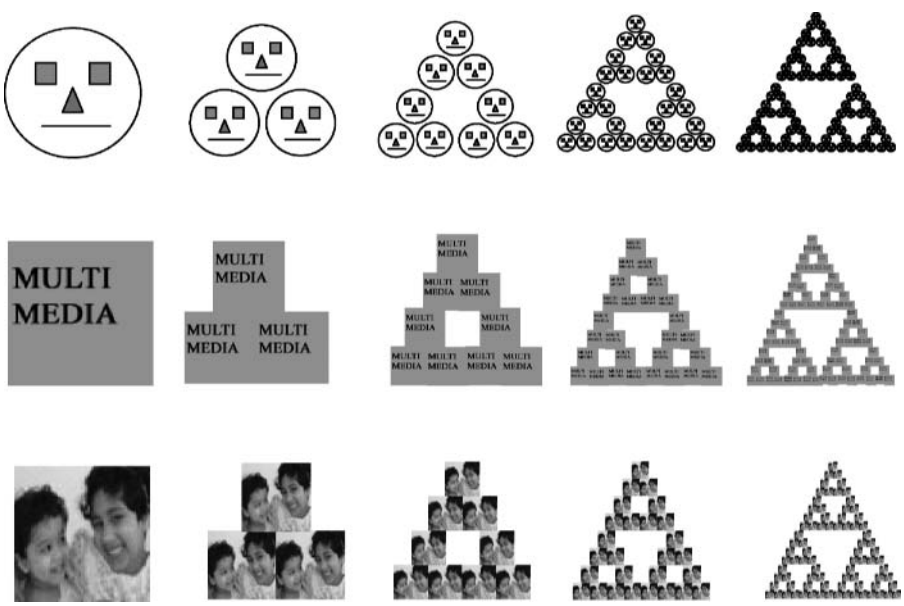


Figure 7-19 Fractal defined by the Sierspinski transformations. Each of the three rows shows five iterations of the fractal transformation. No matter what image you start with, the resulting fractal looks the same because of the same transformation applied to it.

The general-purpose transformations that are employed here are called *affine transformations*, which can scale, rotate, translate, and skew an input image. Affine transformations are mathematically represented by the following formula. An affine transform is completely defined by the values (a_i, b_i, c_i, d_i, e_i, f_i).

$$w_i\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix}$$

6.2 Fractal Block Coding

The main idea behind fractal image compression is to find such a transform whose end *attractor* will be the image that we want to compress. Knowing the transformation allows the complete reconstruction of the image. Finding the transformation (or values a_i, b_i, c_i, d_i, e_i, f_i) given the end *attractor* image is equivalent to performing the inverse process described earlier. This inverse process is not only difficult but also might at times be impossible, especially for natural images. However, a trick employed by all fractal compression techniques (attributed to Michael Barnsley) is not to look for one transformation, but a set of transformations that can collectively be combined to generate different fractals, each for different sections of the image.

For this purpose, the image is divided into regular segmented sets of blocks, each having a dimension $n \times n$. Each block here can be termed as the range block because it is part of the attractor. For each range block, we need to compute a domain block and a transformation, which, when applied to the domain block, recursively gives rise to the range block. The range block typically comes from the image itself. This is where the computational difficulty lies—for each range block, find the domain block (chosen from among image blocks in the image) and a transformation that reduces the distortion to the required level. Compression is achieved by the fact that when this process is over, the whole image can be produced from a few domain blocks and a few transformations, which requires fewer bits to store compared with image pixels. The process is described in Figure 7-20.

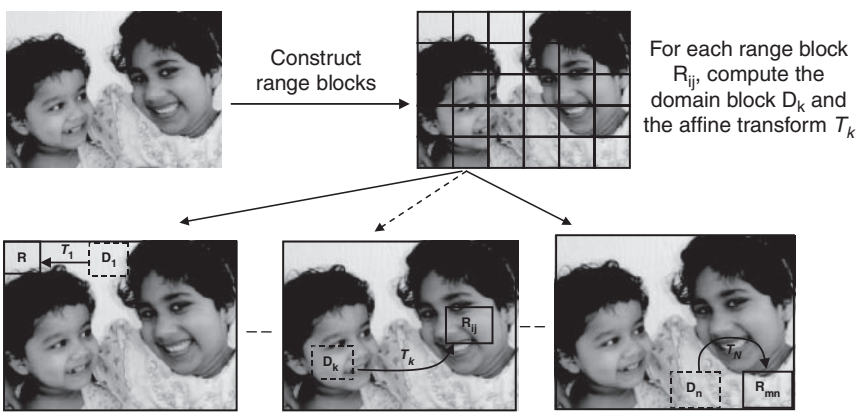


Figure 7-20 Fractal-based encoding. The block diagram shows how the domain blocks D_k can be determined for each range block R_{ij} by computing the transformation T_k. Ultimately, a few domain blocks D_k and the associated T_k transformations can be used to reconstruct the entire image.

6.3 The Future of Fractal Image Compression

Fractal image compression has produced results that are comparable to those obtained by DCT or DWT. In certain specific image examples, it is also known to perform better. Although the research directions seem promising, fractal image compression is still considered to be in research development. Many different research groups and companies are trying to develop new algorithms that result in faster encoding time and smaller files. But much of the impending adoption of fractal compression into standards has been stifled by the intractable nature of finding a set of transformations to compress the image. Hence, although tests have shown these techniques to sometimes outperform transform-based encoding techniques on standardized images in terms of quality, a fractal compression standard does not exist. The Fractal Image Format (FIF) is not standardized, is not embedded into any browsers, and is still an active area of research that could eventually evolve into another standard.

7 TRANSMISSION ISSUES IN COMPRESSED IMAGES

The goal of all image-coding algorithms is to provide an efficient way to compress and communicate image information. Although the state-of-the-art techniques that use DCTs and wavelets do provide good compression, transmission of digital images is still challenging with the growing number of images, their sizes, real-time interaction with compressed images, and the variety of bandwidths on which transmission needs to be supported. High-bandwidth networks do provide seamless delivery of image content; however, low-bandwidth networks often have large latencies before the end user can even view the image or interact with it. The following examples better illustrate these issues.

When researching or browsing digital libraries for a topic, a flood of images needs to be delivered to the user in some form, normally JPEG-encoded images organized on an HTML page. On lower-bandwidth connections, the sequential delivery hampers the experience because the end terminal decodes data block-by-block sequentially for all the images. The user can make no decision on the usability of the images until all the data blocks arrive and are decoded to get a complete picture. A better experience could be obtained by delivering all the data quickly in a coarse form first, where the user can immediately get a full view of the image(s). This view is then refined in time as additional data arrives. Decisions whether to view the whole data set or discard it for a new search can be made earlier, increasing the research efficiency.

Satellite imagery is commonly used for a variety of industrial and consumer-level imaging operations, such as weather reference, navigation, cartography, and GIS. The images (or image sections) are normally stored in the baseline JPEG format where all the blocks are sequentially encoded. This sequentially organized image data needs to be completely downloaded before it can be decoded and viewed by a user, who in this case, is probably more interested in interacting with the data, zooming in, and navigating around to view image information. The whole experience is made less interactive by the need to wait for all data until the image can be viewed.

The central issue to all these problems is that the end terminal needs to have all the data pertaining to the compressed bit stream prior to effectively decoding and

displaying the image. Rather than delivering the data stream in an absolute fashion, reorganizing the bit stream and delivering it in a progressive manner can be more effective. Instead of delivering the compressed data at a full resolution in a sequential manner, it will be more effective to transmit a part of the bit stream that approximates the entire image first. The quality of this image can then progressively improve as data arrives at the end terminal. Progressive schemes are useful for an end user to very quickly view/interact with the image. The ISO standards have modes of compression that makes the compressed bit stream more amenable to progressive delivery. The following sections discuss some of these progressive techniques.

7.1 Progressive Transmission Using DCTs in JPEG

We discussed the main baseline JPEG mode that sequentially delivers the compressed coefficients for all the blocks. Each block was encoded in a sequential scan (DC + zigzag AC coefficients). JPEG also has progressive modes where the each image block is encoded in multiple scans rather than a single scan. The first scan encodes a rough approximation of all the blocks, which upon decoding creates a crude but recognizable version of the image. This can be refined by subsequent scans that add detail to the image, until the picture quality reaches the level generated by quantization using the JPEG tables. The process of progressive transmission is illustrated in Figure 7-21. JPEG provides for two different methods by which progressive transmission can be achieved:

- *Spectral selection*—This works by transmitting only the DC coefficients of all the blocks in the first scan, then the first AC coefficient of all the blocks in the second scan, followed by all the second AC coefficients, and so on. Each frequency coefficient of all the blocks is being sent in individual scans. The first scan to arrive at the decoder are the DC values, which can be decoded and displayed to form a crude image containing low frequency. This is followed by decoding the additional coefficients as the scans arrive successively at the decoder.
- *Successive bit approximation*—Rather than transmitting each of the spectral coefficients individually, all the coefficients can be sent in one scan, but in a

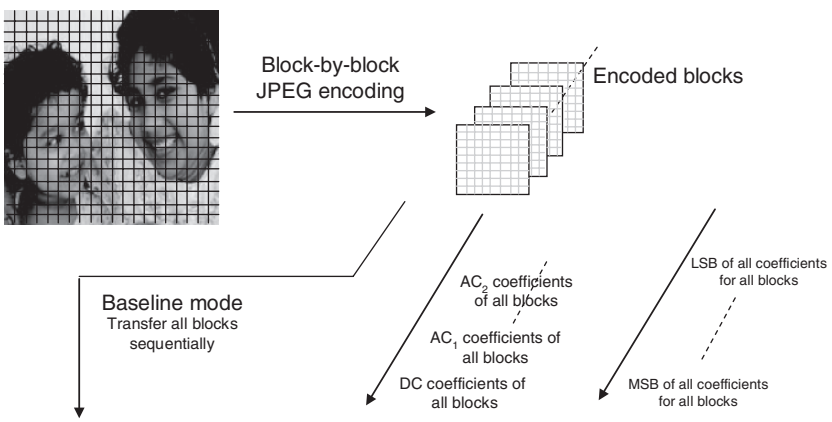


Figure 7-21 Progressive transmission in JPEG.

bit-by-bit manner. That is, the first scan contains the most significant bit of all the coefficients of all the blocks, the second scan contains the next most significant bit, and so on, until the last scan contains the least significant bit of all the coefficients from all the blocks.

An example showing progressive transmission, along with baseline transmission is shown in Figure 7-22. Apart from these two modes, JPEG provides a hierarchical

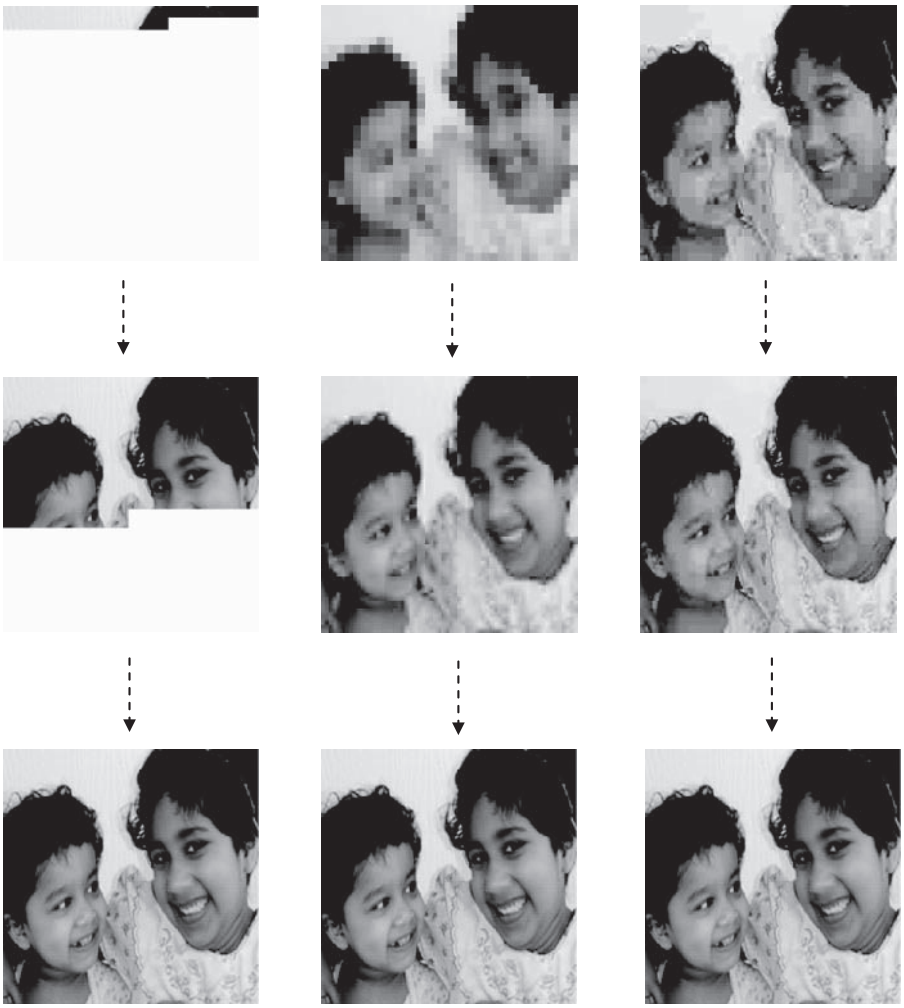


Figure 7-22 Progressive transmission in JPEG. Decoded results as data arrives at the decoder. The first column shows the decoded blocks as they arrive sequentially in the baseline mode. The second column shows the decoded data as the frequency coefficients of all the blocks sequentially arrive in the spectral selection mode. The last column shows the decoded data after frequency coefficients arrive in successive bit approximation. The second and third progressive procedures show that the whole image is available from the beginning.

mode that is also used for transmitting data progressively at different spatial resolutions. The working of this hierarchical mode is explained via an exercise in question 11 at the end of this chapter.

7.2 Progressive Transmission Using Wavelets in JPEG 2000

Wavelet compression, as used in JPEG 2000, naturally lends itself to progressive transmission. Given an input image, each stage transforms the input into four subimages that contain different bands in two dimensions. The upper-right corner in Figure 7-16 shows a two-level decomposition of a wavelet transform with each level having four subbands. For progressive transmission purposes, the data can be sent to the end client's decoder level-by-level from the highest to the lowest, and at each level transmitting the lower-frequency bands prior to the higher-frequency bands. This is shown in Figure 7-23. The decoder decodes each band as the data is received and incrementally produces better results.

8 THE DISCRETE COSINE TRANSFORM

This section explains the Discrete Cosine transform (DCT), which is extensively used in compression techniques, The DCT is a frequency transform that takes a signal, an image in our case, from the spatial or pixel domain to the frequency domain. Because the transform is completely invertible, the corresponding Inverse Discrete Cosine transform brings the frequency domain back to the spatial domain. The DCT-IDCT process is in theory completely lossless. The Discrete Cosine transform for an $n \times n$ image or image block is given by the following (where $n = 8$):

$$F(u,v) = \left(\frac{1}{4}C(u)C(v)\right)\left[\sum_{x=0}^{x=7}\sum_{y=0}^{y=7} f(x,y) \times \cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right]$$

The Inverse Discrete Cosine transform for the same is given by the following:

$$f(x,y) = \frac{1}{4}\left[\sum_{u=0}^{u=7}\sum_{v=0}^{v=7} C(u)C(v) \times F(u,v) \times \cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right]$$

$$where: C(u),C(v) = 1/\sqrt{2}\ for\ u, v = 0$$

$$C(u),C(v) = 1\ otherwise$$

Essentially, the preceding formula computes the frequency coefficients $F(u,v)$ for the input image $f(x,y)$. Each frequency coefficient $F(u,v)$ represents the amount of contribution that the 2D frequency (u,v) represents in the image. The JPEG standard works by using the DCT in a block-based mode where the original image is broken down into 8×8 sized blocks. Computationally, this is the same as taking each 8×8 block and treating it as an individual $f(x,y)$ image. Let us consider one such 8×8 image block.

All possible variations of pixel values in an 8×8 image block amount to image frequencies. However, the image frequencies can vary eight different ways

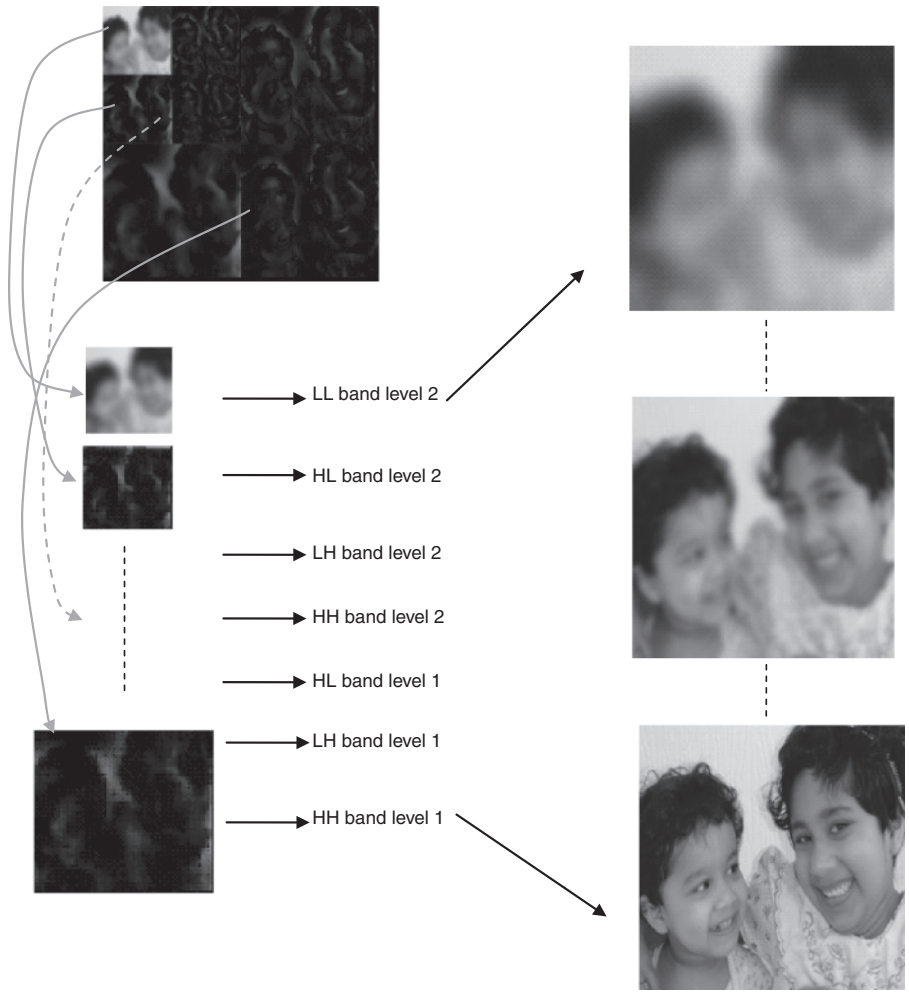


Figure 7-23 Progressive transmission in JPEG 2000 on a level 2 DWT-coded wavelet. The low- to high-frequency bands for each level are transmitted progressively for each level. The data for each band is decoded as it arrives at the decoder to produce decoded images progressively, as shown in the right column.

in the u direction and eight different ways in the v direction, altogether producing 64 variations. These 64 frequencies form the basis functions of the 8×8 image block and are shown in Figure 7-24, each indexed by (u,v). Each image block here—let us call it $B_{uv}(x,y)$—represents how the original $f(x,y)$ would look had there been only one frequency (u,v) in it. The top row (0,0), (0,1), (0,2) . . .(0,7) shows the eight possible variations for the frequency in the horizontal v direction. $B_{00}(x,y)$, $B_{01}(x,y)$ . . . $B_{07}(x,y)$ shows an $f(x,y)$ image block with no vertical frequency but the corresponding horizontal frequency. For each horizontal

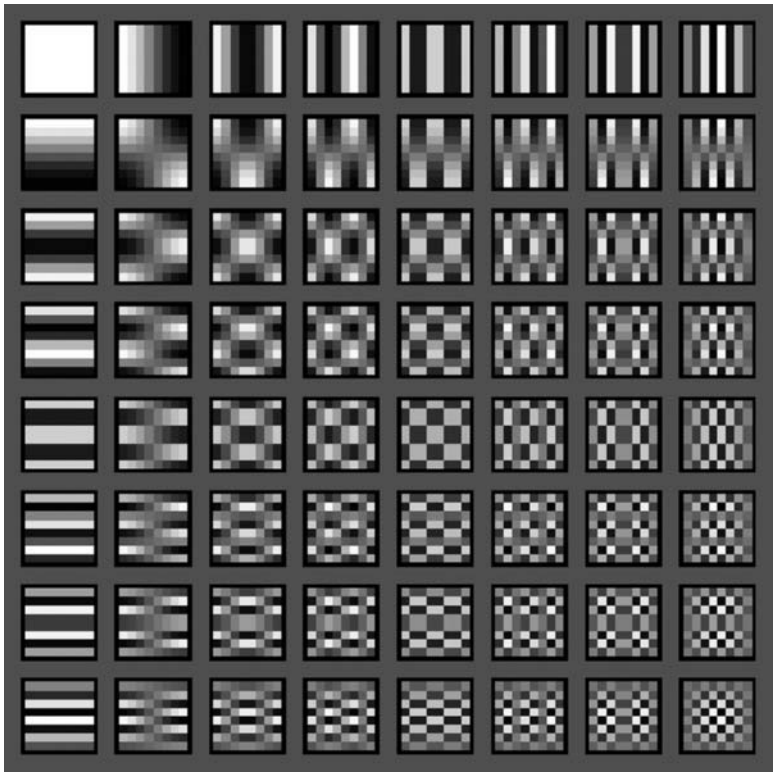


Figure 7-24 DCT basis functions for 2D case.

frequency, the column gives all possible variations in the vertical v direction, keeping the u direction frequency constant.

The DCT transform shown in the previous formula takes an 8×8 image block and produces 64 transform coefficient values $F(u,v)$. Each of the 64 $F(u,v)$ values correspond to a (u,v) frequency of Figure 7-24, representing how much of that frequency is present. In other words, we attempt to represent an image as a linear weighted combination of the basis functions as follows:

$$\begin{aligned} f(x,y) = {} & F(0,0) \times B_{00}(x,y) + F(0,1) \times B_{01}(x,y) + \ldots + F(0,7) \times B_{07}(x,y) + \\ & F(1,0) \times B_{10}(x,y) + F(1,1) \times B_{11}(x,y) + \ldots + F(1,7) \times B_{17}(x,y) + \\ & F(7,0) \times B_{70}(x,y) + F(7,1) \times B_{71}(x,y) + \ldots + F(7,7) \times B_{77}(x,y) \\ = {} & \sum_{v=0}^{v=7} \sum_{u=0}^{u=7} F(u,v) \times B_{uv}(x,y) \end{aligned}$$

So, given an input 8×8 image $f(x,y)$, the DCT attempts to compute all 64 basis frequency coefficients $F(u,v)$, such that the weighted combinations of these 64 basis images $B_{uv}(x,y)$ produces $f(x,y)$.

9 EXERCISES

1. [**R01**] Say you scan a 5″ × 7″ photograph in at 96 ppi (points per inch or pixels per inch) in RGB color mode (24-bit color). For the following questions, compute your answers in 8-bit bytes:
   - How big is the file?
   - If you quantize the full color mode to use 256 indexed colors, how big is the file?
   - If you change the original color mode to 8-bit gray scale, how big is the file?
   - If you quantize the color mode to black and white, how big is the file?
2. [**R04**] Run length encoding (RLE) works by scanning a run of symbols and recording the frequency of a symbol in a run. This is normally a one-dimensional scan. However, images are two-dimensional. Normally images make use of RLE by scanning the pixel data row-by-row. Although this does exploit redundancy in a single dimension, it does not efficiently exploit the coherence of pixel data in a 2D neighborhood. How can you extend RLE to work better on images in 2D?
3. [**R05**] Most image compression techniques are based on transform domain conversions.
   - Explain the principle behind the transform domain conversion.
   - Will transform-based conversions *always* lead to compression? Explain.
   - For what type of images will the transform domain-based compression yield better results?
   - What is the optimum transform for image compression?
   - Can a fixed transform be designed to provide better optimal performance for a set of images?
4. [**R04**] When quantizing the frequency coefficients in any image compression scheme, for example, JPEG, the quantization table used is rarely uniform.
   - Give a rationale for this non uniformity using human perception.
   - How would changing the numbers in the quantization table affect the picture after decompression (for example, if all of them were doubled)?
   - Think of how you would "customize" quantization tables in the JPEG standard. (*Hint*: Think of how you would adaptively change the quantization table depending on the data set.)
5. [**R08**] Assuming that JPEG compression is widespread on all platforms, you start an image processing company that does fast, simple image manipulations on compressed JPEG images (thus not decoding the bit stream).
   - Suppose you have an operation that takes the bit stream that corresponds to the 8 × 8 blocks in an image and replaces a few of them with the bit stream that corresponds to another 8 × 8 block from another image. This will not work—why?

   Here, you are attempting to block-by-block composite two images in the compressed domain, for example, a logo that covers a couple of blocks needs to be composited onto the image.

- What will you need to do to make the previous compositing operation work correctly?
- Normal image transformation operations such as image rotation, scaling, or projective transformations will not work in JPEG, unless you decompress the bit stream first, then perform the operation, and compress it back again. Explain why it will not work, and how JPEG 2000 attempts to solve this.
- Think of watermarking an image. Can you perform this directly on the JPEG bit stream? If so, explain how you would do so. If not, give a reason for why you can't. (You might need to know more about watermarking for this—please see Chapter 13.)

6. [**R06**] In JPEG image compression, the image is divided into 8 × 8 blocks and a transform coding (DCT) is performed on each block.
   - Reason out whether the 8 × 8 blocks coded are independent of each other.

   The bit code generated for a compressed block includes information about both the DC and AC coefficients. Explain how your uncompressed image is affected if the following occurs:
   - In a transmission, you somehow do not receive the compressed bit code for the blocks in the last row (and receive the rest).
   - In a transmission, you somehow do not receive the compressed bit code for the blocks in the first row (and receive the rest).
7. [**R05**] In JPEG image compression, the image is divided into 8 × 8 blocks and a transform coding (DCT) is performed on each block.
   - Explain what the advantages are in taking a block-based DCT instead of the whole-image DCT.
   - In a typical quantization table for JPEG, the quantization steps increase as we move to the right and to the bottom of the table. Why?
8. [**R04**] GIF and JPEG are both used extensively for image compression. One of the advantages of GIF over JPEG is that GIF compression supports transparency or the alpha channel.
   - Why does the traditional baseline mode of JPEG not support the alpha channel?
   - If JPEG were to support alpha channels, what changes would you propose be made to the standard for this?
   - Apart from alpha channel support, do you see any other advantage of GIF over the JPEG standard?
9. [**R04**] Given the following image block in Figure 7-25, answer the following questions.
   - What kind of an image block do you think this represents? Is it a natural photograph, a graphical object, etc.? Give reasons for your answer.
   - Compute the AC and DC coefficients. What do these coefficients normally represent?

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
161	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

Figure 7-25

- Among which coefficients is most of the energy concentrated?
- How do the quantized DCT coefficients look? Use the JPEG provided quantization table displayed in the text (Figure 7-7).

10. **[R03]** Given two images A and B as shown in Figure 7-26 that have to undergo the lossless JPEG compression algorithm, answer the following questions:

Image A				Image B			
127	125	128	129	127	125	128	129
128	126	130	131	127	125	128	129
129	127	131	133	132	118	117	114
130	129	132	135	132	138	113	113

Figure 7-26

- Which one would give better performance, assuming that the predictor X = B is used?
- Will any other predictor give a better performance?

11. **[R06]** We now know that in JPEG compression, the image components are each broken into blocks and each block is transformed to the frequency domain giving a DC and many AC coefficients that are encoded differently.
    - Why are the DC coefficients encoded separately from the AC coefficients? Can you think of an application scenario where this separation can prove to be helpful?
    - The AC coefficients are encoded using a zigzag ordering. Why?
    - Suppose we are encoding a video frame in a video sequence using JPEG (This is done in the intra mode of video compression—more details can be found in the next chapter). But if the video were interlaced instead of progressive, would you want to change the zigzag ordering of AC coefficients prior to entropy encoding? Explain what changes you would make.

12. [**R06**] In the text, we discussed how progressive image transmission can be attained by transmitting different spectral data at each transmission pass. Another way to do progressive image transmission is by representing the image data progressively as shown in Figure 7-27. Here, the image is constructed by storing several different resolution versions of the image. The different levels are stored so that the smallest resolution image (level 3) is stored first, followed by the other images in an increasing order of resolution (size). Progressive transmission can be achieved by transmitting the lowest representation, then the higher representation, and so on.

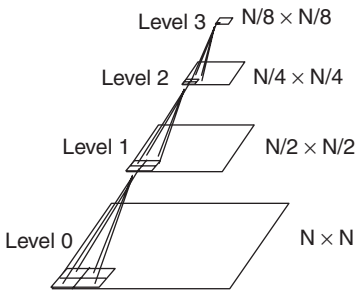


Figure 7-27

- If all the levels of the image are represented as shown in figure 7-27 and stored directly as suggested, how much increase would the overall file size have in comparison with the situation if we had stored only the original largest image (level 0) alone?
- Can you invent any easy way to get rid of this overhead? (*Hint*: Observe that all pixels at level 3 have to be part of level 2, level 1, and level 0.)
- With progressive transmission, you can transmit each level independently and let the decoder do the assembling as the levels arrive. For better compression, you decided to compress the data at each level using the JPEG pipeline. So each level is now a JPEG image. Will this work well? Give reasons.
- How can you improve to get better results? (*Hint*: Think of closed loop encoding.)

13. [**R04**] Let us say you were allowed to do some preprocessing on an input image that has been captured or stored in the RGB format prior to feeding it down into the JPEG chain to compress it.
    - What kind(s) of preprocessing would you do and why?
    - Can you or could you come up with a singular preprocessing for all kinds of images? If you could, what would it be? If not, what will you not be able to address?

14. [**R08**] The DCT-based technique used in JPEG employs a block-by-block DCT-based algorithm. The blocks have been decided by the standard to be of size 8×8.
    - Argue about the size of 8×8. What are the advantages of this size—could they have been bigger or smaller? Give reasons for your answer.

- Also why is it that they have the same resolution in the x and y directions? Give examples of images where employing an $n \times m$ would prove more beneficial—for $n < m$ and $m < n$.
- Assuming that the block-based DCT coding technique is still employed; can the compression ratio be increased? Explain your algorithm.

15. [**R04**] Fractal image compression is a lossy compression technique.
    - Where and how in the fractal computation process does the loss occur?
    - Where does the loss occur in the JPEG DCT computation process?
    - Fractal compression is also block based, where self-similarity of blocks is searched for. How do the two processes (fractal compression and JPEG DCT based compression) differ qualitatively? Give reasons.
16. [**R05**] An 8×8 image is shown in the Figure 7-28. Assuming that you want to perform lossless image compression, identify the smallest bit rate that you can compress the image using each of the following:
    - Discrete Cosine transform
    - Vector quantization
    - Prediction-based

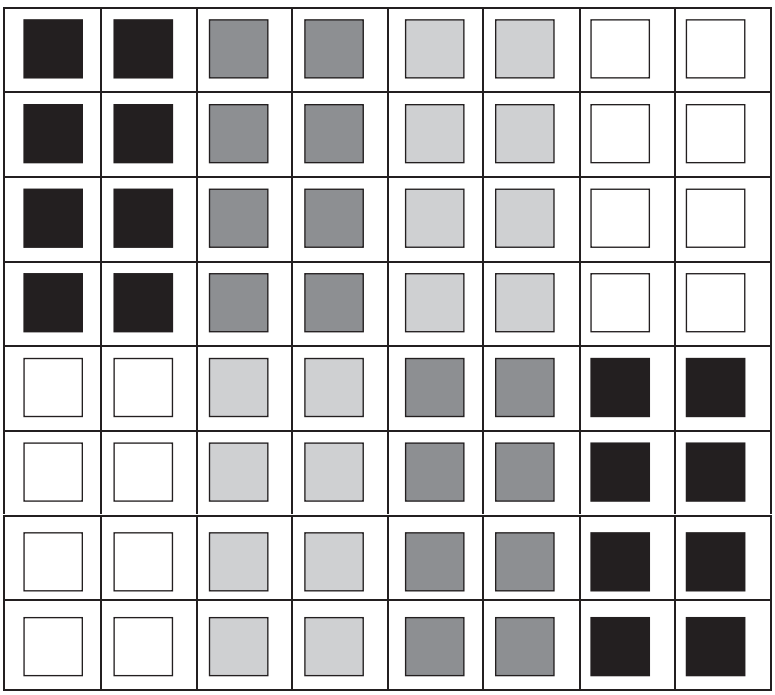


Figure 7-28

You can choose a block size of your own choice (8×8, 4×4, 2×2) and generate any kind of compression scheme you want. Also, you cannot agree a priori about your block size with the decode; you have to communicate all this

information to the decoder by encoding the chosen block size and the compression scheme. In the case of vector quantization, the number of vectors in the codebook, the content of the codebook, and the indexes of the quantized blocks also have to be encoded. In the case of prediction-based, an index of which prediction scheme (among the eight discussed in the text) has to be communicated along with the compressed data.

Repeat the same if your compression scheme can be lossy.

Do any of your answers change (lossy or lossless cases) if you assume that the decoder knows that the image is a four-color image of size 8×8 pixels?

17. **[R03]** GIF Versus PNG Formats
In the early stages of the World Wide Web, the GIF format was predominantly employed for images and their compressed distribution. However, this became problematic through the exercising of intellectual patents in 1995-96. Consequently, the open source committee created a new standard called the PNG format.

- Search the Web to find out about the PNG movement and how the standard differs from the GIF format.
- Find out the qualitative improvements PNG supports over the GIF format.

The PNG format has become the support of choice in the MPEG-4 world (along with JPEG); find out how MPEG-4 employs the PNG format. (You might want to do this after researching MPEG-4.)

PROGRAMMING ASSIGNMENTS

18. In this programming assignment, you will implement a JPEG style compression algorithm. This exercise will make you understand the DCT transform and how quantization leads to compression. Sample images, starting code for image display, and results have been supplied online, in the Student Downloads section of www.cengage.com. Implement the DCT algorithm, and employ it to perform "pseudo JPEG" compression. Here is the description of what you have to do.

The test images are supplied in YUV format. Break each component into 8×8 blocks. For each block, perform a DCT. Quantize the DCT coefficients using a uniform quantization table with the entry $= 2\hat{}n$. Now perform the reverse and decode. Denormalize the DCT coefficients and perform an IDCT to get back your image domain. Compare your results with the ones supplied for $n = 1, 2, 4, 6$ and 8.

You should keep the following programming in mind:

- The DCT function maps the real image space to the frequency domain. Image pixels are almost always represented by *n-bit* positive values for each channel. Remember, prior to computing the DCT, subtract $2^{(n-1)}$ from each image sample to make them range from $-2^{(n-1)}$ to $2^{(n-1)} - 1$.
- Consequently, remember to add the offset back after performing the IDCT.

19. Independent Compression of Y and U, V Channels

Here, you are going to study the effects of DCT quantization on the luminance and chrominance channels individually and study distortions caused by quantization. You will use the program you developed in exercise 17. Modify the program and perform the whole process by just quantizing the U or V channels and then by quantizing just the Y channel. In which case do you see more distortion or loss after the compression/decompression process? Comment on your results.

CHAPTER 8

Media Compression: Video

Similarly to digital images, digital video now plays a central role in a variety of consumer and commercial markets involving both low and high bandwidths. Digital video has been used in the entertainment industry for over a decade now to make movies and distribute them via DVDs. Soon, movies will be distributed digitally by movie studios to theaters, which will project them using digital video projectors, rather than film. The increasing video resolution and formats used in the entertainment industry have kept up with home entertainment demands, but this also results in increased amounts of data produced and distributed. Consumer digital video capture devices are becoming better, both in terms of the recorded quality as well as affordability, which has resulted in their widespread use to capture and archive personal video. Consumer devices typically capture CIF formats with 352×288 size frames at 30 frames per second. Recently, consumer video recording devices that record HD formats are also being made available. For CIF video, assuming 24 bits per pixel, this results in 10 megapixels per second. Onboard memories allow only a few seconds of storage without any effective compression. These consumer devices instead capture and store video in one of the MPEG formats, thus reducing the data to $1/50^{th}$ and even $1/100^{th}$ of its original size. Compression is, therefore, a critical component to make these devices useful.

Additionally, real-time distribution of video content poses a host of problems that need understanding of end-to-end issues from the encoding stations to the client decoder, to produce an effective viewer experience. Live video is distributed over different networks such as digital television, Internet, and so on. The requirements here are to distribute both the existing standard-definition formats and also emerging high-definition formats, which generate huge amounts of data. The data needs to be compressed, distributed, and decoded in real time. There are formats in place, such as MPEG-2 video, to meet such consumer-level applications. Another recent video format,

the H.264, is also gaining popularity because of the bandwidth savings that it can deliver over its precursor formats. Additional standards that support more powerful algorithms will also be required in the future to meet the increasing resolution requirements in the distribution of digital video. This chapter is devoted to the state of the art in video compression, discussing issues related to both non-real-time as well as real-time compression. Although the core video compression algorithm introduced in Section 1 is the same for both non-real-time and real-time techniques, the way it is used to meet the end requirements is different. This chapter begins by exploring a few video bit rate statistics in different markets that should show not only the need for compression, but also show quantitatively how much compression is expected in different applications. Next, in Section 1, we discuss the fundamental video-compression algorithm based on motion compensation techniques, and analyze its flavors and complexity in Sections 2 and 3. In Section 4, we talk about the popularly used compression standards and how they adapt the generic motion compensation technique in a variety of commercial applications. These standards include the ones proposed by the International Organization for Standardization (MPEG-1, MPEG-2, MPEG-4) as well as the International Telecommunication Union (H.261, H.263, H.264). When video is distributed over networks or stored, the bit rate generated by compression is important. Issues that relate to constant bit rate and variable bit rates are discussed in Section 5. Finally, in Section 6, we talk about real-time constraints and show snapshots of what a commercial encoder looks like. To understand this chapter, you need a thorough understanding of video representation, color, and frequency domain issues, which were presented in part 1 of this book.

The need to compress video can easily be inferred from the table shown in Figure 8-1. The table shows the amount of data that needs to be delivered to the end client decoder per second, so that the client can render video at the required frame rate.

From Figure 8-1, it should be clear that uncompressed video requires significant memory for storage and also a lot of time to transmit over commonly used networks and, consequently, cannot be streamed. Moreover, recording instruments such as digital video camcorders would not be able to write data to disk at the required rate without making use of expensive memory architectures.

1 GENERAL THEORY OF VIDEO COMPRESSION

In image compression, a static frame is compressed. The compression techniques mostly exploit both spatial and spectral redundancy. This redundancy exists in an image because an image is a coherent structure, and not a random collection of pixels. Local regions in an image have very similar pixels exhibiting limited variations. Now, because a video is a sequence of image frames, each of which must have spatial redundancy, you might say that we could compress video by applying an image compression algorithm to each frame, for instance compressing each image frame as a JPEG image. Of course, this would result in compression—and in multimedia, there is an informal standard used to describe this compression. It is known as motion JPEG (M-JPEG). The M-JPEG standard compresses each frame in a video sequence as a JPEG image. The decoder in these cases receives compressed frames, which it decodes frame by frame and displays. M-JPEG is indeed used by unsophisticated videoconferencing applications where a few

Multimedia video data	NTSC video	QCIF video	CIF video	HDTV (progressive) video	HDTV (interlaced) video
Frame size	720 × 486	176 × 144	352 × 288	1280 × 720	1920 × 1080
Bits/pixel	16 bpp	12 bpp	12 bpp	12 bpp	12 bpp
Frame rate	29.97	30	30	59.94	29.97
Uncompressed frame size (bytes)	700 KB	38 KB	152 KB	1.38 MB	3.11 MB
Uncompressed data produced per second (bits per second)	167.79 Mbps	9.12 Mbps	36.5 Mbps	662.89 Mbps	745.75 Mbps
Disk space to store 1 min of video in bytes	12.6 GB	68.4 MB	273 MB	4.96 GB	5.59 GB
Transmission time for 1 min of video (56 K Modem)	49.94 hours	2.71 hours	10.8 hours	Not worth it	Not worth it
Transmission time for 1 second of video (780 Kb DSL)	7.31 hours	22.31 minutes	0.775 hours	14.16 hours	15.93 hours
Transmission time for 1 second of video (1 Gb fibre optic)	10.07 seconds	0.55 seconds	2.19 seconds	39.72 seconds	44.7 seconds

Figure 8-1 Examples showing storage space, transmission bandwidth, and transmission time required for uncompressed video

video frames are compressed per second. For example, most instant messenger communication that involves video compresses it at small frame rates of 0.5–2 frames per second. It is also used in mobile applications such as inexpensive consumer digital cameras as well as cameras used with laptops, cell phones, and other computing devices.

However, significantly higher compression rates can be achieved by exploiting another kind of redundancy in video—*temporal redundancy*. Just as pixel values are locally similar within a frame, they are also correlated across frames. For example, when you see a video, most pixels that constitute the backdrop or background do not vary much from frame to frame. Areas may move either when objects in the video move, or when the camera moves, but this movement is continuous and, hence, predictable. This is illustrated in Figure 8-2, which shows a few video frames of a girl walking. The camera also moves to keep the subject close to the center of the image. Although each frame is different on a pixel-by-pixel basis, there is a great deal of correlation from frame to frame. The girl in the foreground moves, so the pixels on her face/dress have

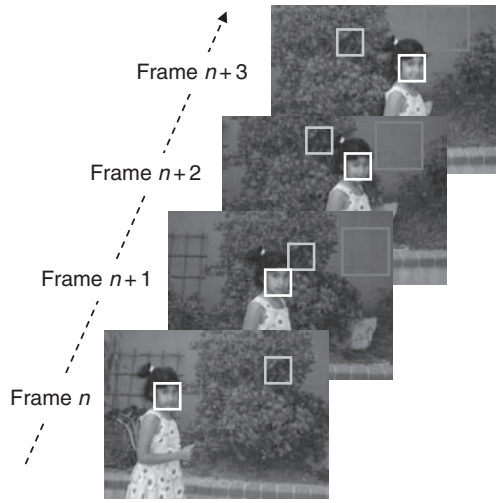


Figure 8-2 Temporal redundancy in video. Four frames of a video sequence are shown. The girl in the foreground is moving, whereas the background objects are static; however, the camera is also moving. Although each frame is different, there are areas of the background and the foreground that remain the same or are very similar.

different locations in each frame, but the color values of these pixels do not change much. The same reasoning can be applied to the pixels that make up the background. The background pixels in this case move even less than the foreground pixels.

For video, it makes more sense to code only the *changed* information from frame to frame rather than coding the whole frame—akin to the predictive DPCM techniques. As a simple extension of the one-dimensional DPCM technique explained in Chapter 5, we could find the frame-by-frame differences and code only the differences. The temporal correlation should result in the difference frame having lower information content than the frame itself. It would, therefore, be more efficient to encode the video by encoding the difference frames. An example of a frame-by-frame difference is shown in Figure 8-3. The top row shows a low-motion content video, where the background is almost unchanged and the girl in the foreground has undergone slight motion. The difference frame (frame $n + 1$ − frame n) has much less information content than frame $n + 1$. The decoder in this case reconstructs frame $n + 1$ by decoding the frame difference and adding that to the previously reconstructed frame n. The bottom row in the figure shows the similar computation in the case of higher motion, where the camera and the foreground are moving. The frame difference here has much higher entropy compared with the frame difference in the first case. This can get worse with higher motion from frame to frame.

Whenever there is motion due to movement of objects, temporal redundancy can be better exploited by predicting the motion of pixels and regions from frame to frame, rather than predicting the frame as a whole. The process of pixel- and region-based motion prediction to make better use of temporal redundancy is explained in the next few subsections.

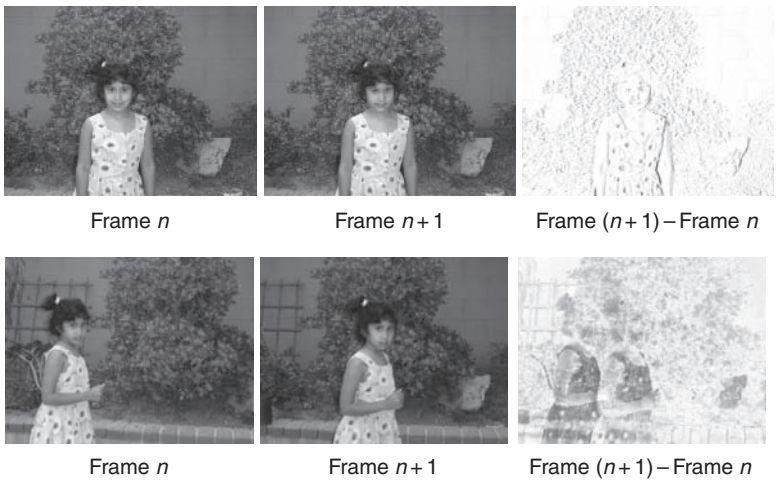


Figure 8-3 The top row shows two successive frames in a low-motion video. The frame difference of the Y channel is shown on the right. The bottom row shows two successive frames where the object/camera motion is higher. The difference image in this case contains a lot more information than the previous case. See the color insert in this textbook for a full-color version of this image.

1.1 Temporal Redundancy

Temporal redundancy can be modeled mathematically by studying how and why a pixel changes its value from frame to frame. Let us consider a pixel at position (x, y) in a frame n, as shown in Figure 8-4. The pixel has color values specified by $C_n(x, y)$. In the next frame $n + 1$ the pixel at the same position, known as the *colocated pixel*, has color values given by $C_{n+1}(x, y)$. We want to predict the value of $C_{n+1}(x, y)$ based on the known pixels of frame n. There are many possibilities to understand the relationship between $C_{n+1}(x, y)$ and $C_n(x, y)$.

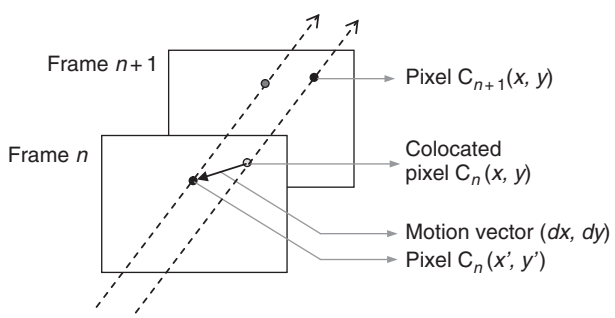


Figure 8-4 Pixel motion prediction. The pixel shown in the frame n has moved to a new location in frame n + 1. Consequently, $C_{n+1}(x, y)$ in frame n + 1 is not the same as $C_n(x, y)$ but is offset by the motion vector (dx,dy).

First, the pixel color might not change at all because none of the objects moved, the camera did not move, and the lighting conditions remained constant. We then have

$$C_{n+1}(x, y) = C_n(x, y)$$

This typically happens for pixels in the background when recording with a static camera. Although pixels are supposed to have the exact same value, there is always a slight change because of quantization and noise errors in the capturing device.

The pixel color might also change because the object(s) in the scene moved, the camera moved relative to the scene, or there were lighting changes in the scene that ultimately affected the pixel intensity values. In such cases,

$$C_{n+1}(x, y) \neq C_n(x, y)$$

Video sequences are normally captured at upward of 15 fps, where object-camera relative movements get manifested slowly from frame to frame. The pixel $C_{n+1}(x, y)$ might not be equal to $C_n(x, y)$ but equal to $C_n(x', y')$ for some (x', y') in the local neighborhood. This value can be represented as follows:

$$C_{n+1}(x, y) = C_n(x', y') = C_n(x - dx, y - dy)$$

where dx, dy is the vector that shows how the pixel moved. The (dx, dy) vector is known as the motion vector. Because of quantization noise and errors in prediction, we can expect the preceding equation to be modified as follows:

$$C_{n+1}(x, y) = C_n(x - dx, y - dy) + e(x, y)$$

where $e(x, y)$ is the small error difference. Thus, knowing $C_n(x', y')$ along with dx, dy and $e(x, y)$, we should be in a position to predict $C_{n+1}(x, y)$. This argument could be further developed by saying that all pixels of frame $n + 1$ can be predicted from the previous frame n, if the motion vectors (dx, dy) and the error differences $e(x, y)$ are known for each pixel. It would, thus, imply that we do not need to compress and send all the information of frame $n + 1$, but only that which is needed to predict it, namely the motion vectors and the errors. Practically, however, the coherency from frame to frame is better exploited by observing that *groups* of contiguous pixels, rather than individual pixels, move together with the same motion vector. Therefore, it makes more sense to predict frame $n + 1$ in the form of regions or blocks rather than individual pixels.

1.2 Block-Based Frame Prediction

The current image frame, which needs to be compressed, is known as the *target frame*. The target frame is predicted on a block-by-block basis from a previous[1] frame, which is known as the *reference frame*. The target frame is divided in blocks known as macroblocks and each macroblock is predicted from the most similar-looking region in the reference frame. Macroblock-based motion prediction is used for two main reasons. First, temporal coherency is better exploited by regions/blocks moving from frame to frame.

1 For the purpose of discussion here, we assume that the reference frame is one from the past; however, in a later section, we show the usage of future frames in predicting the current frame.

Second, for compression reasons, it is also more efficient to divide a frame into blocks and generate a motion vector for each block rather than a motion vector for each pixel.

The problem can be better understood by looking at Figure 8-5 where frame $n + 1$ is to be predicted based on frame n on a block-by-block basis. The prediction entails finding a motion vector (dx, dy) for each block. When the motion can be correctly predicted with accurate motion vectors, a well-matched area of frame n can be used to appropriately predict the macroblock. In such cases, the error difference between the actual and predicted block is very small. But this is not always the case, for a variety of reasons—the region of frame $n + 1$ is new and cannot be predicted correctly from frame n. For example, objects move in to or out of a scene at frame $n + 1$. In Figure 8-5, macroblocks

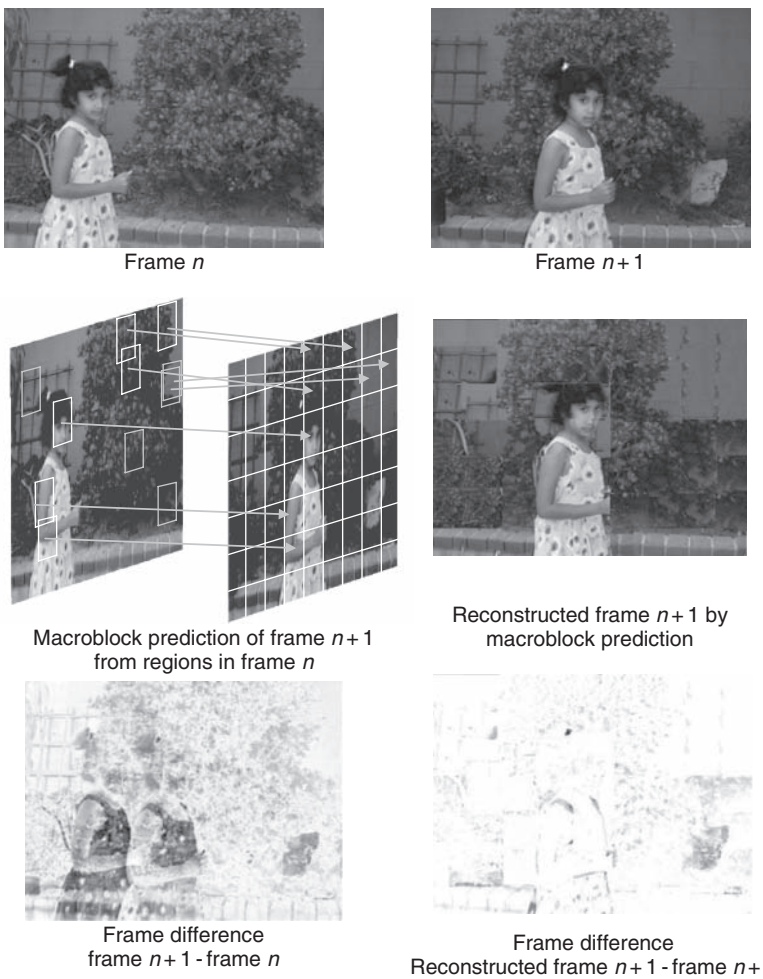


Figure 8-5 Macroblock-based frame prediction. Frame n + 1 is divided into macroblocks and each macroblock is predicted from an area in frame n. The reconstructed frame based on this prediction is formed by copying the frame n areas into the appropriate frame n + 1 macroblocks. The bottom line shows that the error difference reduces as a result of this prediction. See the color insert in this textbook for a full-color version of this image.

pertaining to the girl in the foreground and some of the central background macroblocks can be well predicted. Consequently, the error differences of these macroblocks have lesser entropy. But macroblock areas which are new in frame $n + 1$ and not visible in frame n cannot be well predicted from areas in frame n and, consequently, have higher entropy. Examples of such macroblocks can be seen in the lower right area of frame $n + 1$. But a comparison shown at the bottom of Figure 8-5 shows that the overall difference with motion prediction is much less than no motion prediction.

The process of predicting target frames from reference frames using macroblocks is known as motion compensation. The process of compressing a frame in this mode involves the following:

- Finding the best motion vector for each macroblock
- Creating a predicted or motion-compensated frame image
- Deriving the prediction error image
- Compressing error image using the JPEG pipeline (lossy) and the motion vectors using entropy coding (lossless)

Finding the best motion vector for a macroblock needs to be accurate for better prediction and compression. The process and the metrics used are explained in the next section. Once the motion vectors are available for each macroblock, a predicted frame can be computed using the motion vectors and the reference frame. The predicted frame is created by copying into each of its macroblocks the pixel regions from the reference frame using the computed motion vectors. The predicted frame is bound to be close to the current frame, but not exactly the same for reasons explained in section 1.1. The error image obtained by subtracting the pixels of the predicted frame from the current frame is necessary for the decoder to reconstruct the target frame. Before we delve into motion vector computation, it is important to understand what happens to the motion vectors and the error image at the encoder and how they get utilized at the decoder. The encoder and decoder processes are schematically illustrated in Figure 8-6. The encoder compresses the motion vectors using a lossless algorithm (for example, Huffman coding) and the error image using a lossy algorithm (as in the JPEG pipeline) and sends it to the decoder. Assuming that the decoder already received and reconstructed frame n, it now attempts to decode and reconstruct frame $n + 1$ using the encoded motion vectors and the error image for frame $n + 1$ sent to it by the encoder. The decoder first reconstructs frame $n + 1$ by filling each of its macroblocks with regions from frame n using the decoded motion vectors. To this reconstructed frame, the error image is added to get the final frame $n + 1$.

Thus, knowing the motion compensation process at the encoder and the reconstruction at the decoder, you should see that the motion vectors and the error image are intricately related. If the predictions based on motion vectors are bad, the error image has higher entropy. If the predictions based on motion vectors are good, the error image has lower information. The former case has more information to compress compared with the lower-entropy latter case. Therefore, the encoder can do a better job at compression if the error image has less entropy, which is obtained by accurate motion vector computations. Computing motion vectors correctly and accurately is, therefore, very critical to obtain better compression rates.

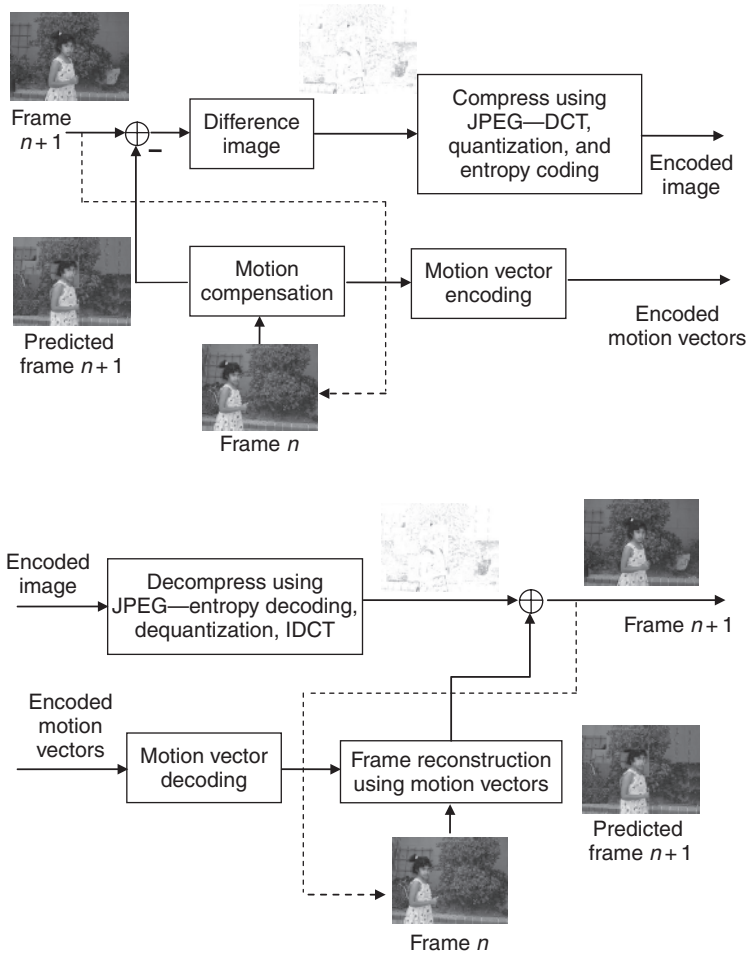


Figure 8-6 Video encoder (above) and decoder (below) block diagrams. The encoder encodes only motion vectors and the difference image. The decoder uses the motion vectors to reconstruct frame n + 1 and adds the error image to get the final frame n + 1.

1.3 Computing Motion Vectors

In this section, we describe how motion vectors are computed by searching in a specified area around the colocated position in the reference frame. The search area is specified by a parameter known as the *search parameter k*, as shown in Figure 8-7. The value of k typically ranges from 0 to 31, and is set depending on how fast objects or regions are moving from frame to frame. For fast motion, better predictions are obtained with larger k values because fast-moving objects tend to move farther from frame to frame. The search region defined by k is a rectangular area in the reference frame specified around the colocated position of

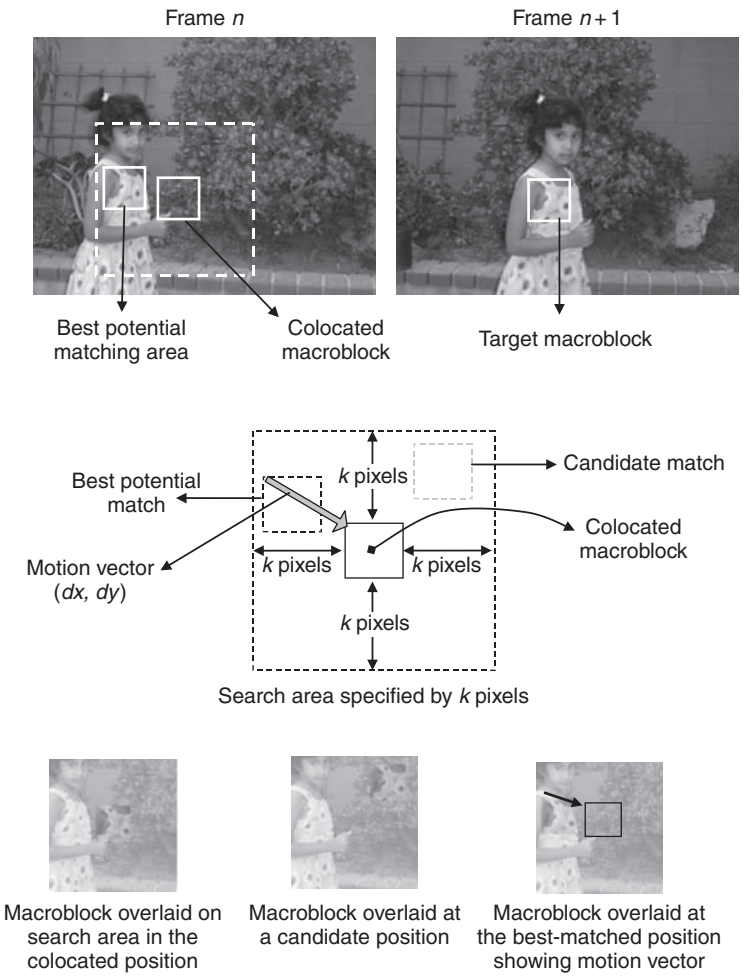


Figure 8-7 Motion vector search. The encoder needs to search the best match for macroblock in frame n+ from frame n. The search is performed within a search area specified by parameter k around the colocated position. The best-matching region obtained in the search area by a pixel-by-pixel difference gives the motion vector. See the color insert in this textbook for a full-color version of this image.

the target macroblock. The goal of the motion vector search is to find the best-matching area in the reference frame to the target macroblock out of all candidate macroblocks in the search region.

For convenience, let us use the upper-left corner as the origin of the target macroblock with width m and height n. Then, $C_{n+1}(x, y)$, where x has values in $[0, m]$ and y has values in $[0, n]$, defines the pixels of the macroblock. The colocated macroblock pixels in the reference frame (note that we do not divide the reference frame in the blocks) can be defined as $C_n(x, y)$ where x and y have the

same ranges. A candidate reference macroblock position to be evaluated for a match is defined by a motion vector $mv = (i, j)$ where i, j both vary in the search area and can take on values from $-k$ to $+k$. At any such candidate position, specified by $mv(i, j)$, we can compute the difference between the candidate area and the target block by their *mean absolute difference* (MAD), defined as follows:

$$MAD\,(i, j) = \frac{\sum_{p=1}^{m}\sum_{q=1}^{n}|C_{n+1}[p, q] - C_n[p + i, q + j]|}{mn}$$

The goal of the search task then is to find a motion vector (i, j) such that $MAD\,(i, j)$ is minimized. In the preceding formula, we used the mean absolute difference as a metric to find the best motion vector. Most commercial encoders use the sum of absolute differences (SAD) because of its lower computational cost. SAD is similar to MAD but without the division by mn.

Although MAD and SAD are commonly used, other metrics have been proposed in the motion compensation literature. Some of these are known to give better search results, but they do so at the cost of additional computations. Other metrics are as follows:
mean square difference (MSD)

$$\mathrm{MSD(i, j)} = \frac{\sum_{p=1}^{m}\sum_{q=1}^{n}(C_{n+1}[p, q] - C_n[p + i, q + j])^2}{mn}$$

pel difference classification

$$\mathrm{PEL(i, j)} = \sum_{p=1}^{m}\sum_{q=1}^{n}[ord\,(|C_{n+1}[p, q] - C_n[p + i, q + j]| \le t)]$$

where t is some predefined threshold that decides a match and

$ord(x) = 1$ if x is true.

projective ordering

$$\mathrm{PO(i, j)} = \sum_{p=1}^{m}\left|\sum_{q=1}^{n}C_{n+1}[p, q] - \sum_{q=1}^{n}C_n[p + i, q + j]\right| + \sum_{q=1}^{n}\left|\sum_{p=1}^{m}C_{n+1}[p, q] - \sum_{p=1}^{q}C_n[p + i, q + j]\right|$$

1.4 Size of Macroblocks

A natural question in block-based motion compensation involves the desired size of each macro block. The main goal of all predictions is to minimize the entropy in the error image. Also, the error image is encoded in the intramode using the JPEG pipeline and the motion vectors of all the blocks for that frame are encoded in a

lossless manner. For compression reasons, it is essential to transmit as few bits as possible for the predicted frames.

Smaller macroblocks increase the number of blocks in the target frame, which, in turn, implies a larger number of motion vectors to predict the target frame. This requires more bits to compress motion vectors, but smaller macroblocks tend to give better error images for small motions, requiring fewer bits. Larger macroblocks, on the other hand, yield fewer motion vectors to compress, but also tend to increase the prediction error. This is because larger areas could possibly cover more than one moving region within a large macro block. This is better illustrated in Figure 8-8, where the large macroblock matches give a larger error block when compared with

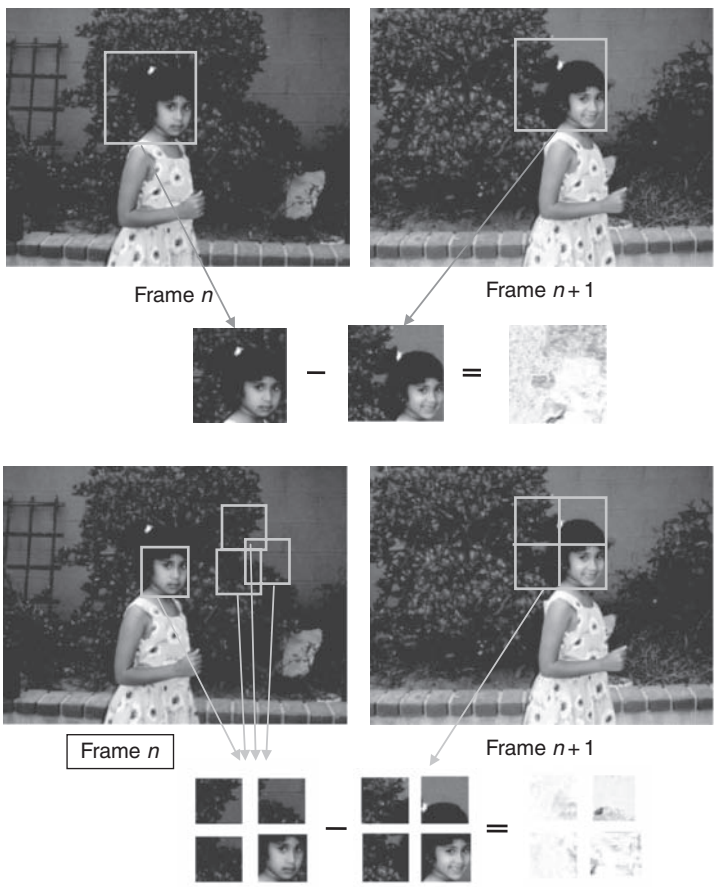


Figure 8-8 Macroblock sizes are important during motion compensation. The top set of images show the resulting higher entropy difference between the actual and predicted areas for a large macroblock. The bottom set of images illustrates how this entropy difference is lower for the same area using smaller sized macroblocks. The overall entropy in the individual differences is minimized because the sizes in the second case better approximate motion areas. See the color insert in this textbook for a full-color version of this image.

the same area approximated by smaller macroblocks. Most ISO and ITU standards use an empirically set macroblock size of 16 × 16, which seems to be a good compromise between minimizing the entropy of the error image and reducing the number of motion vectors for most ranges of motion. However, the later standards, such as H.264, use variable-sized macroblocks when required to allow flexibility and yield lower error predictions.

1.5 Open Loop versus Closed Loop Motion Compensation

The previous sections explained how the encoder predicts frame $n + 1$ from a reference frame n, which was given as input to the encoder. The motion vectors and error image computed as a result of motion compensation depend on frame n. The decoding and reconstruction of frame $n + 1$ at the decoder would, thus, be correct if the decoder also had the reference frame n. Unfortunately, the decoder only has a decoded and quantized version of reference frame n. The reconstruction of frame $n + 1$ using the decoded reference frame with motion vectors computed from the original reference frame is bound to increase the distortion, which will also accumulate with successive predictions. The encoder in this case performs motion compensation in an open loop, which results in undesirable distortions. It is more effective to perform motion compensation in a closed loop, as shown in Figure 8-9, where the encoder attempts to predict frame $n + 1$ not from the input reference frame n, but from a quantized and decoded version of frame n. The encoder in this case also simulates a decoder, which it uses to decode and reconstruct the reference frames to use in motion compensation. This prevents the error from accumulating.

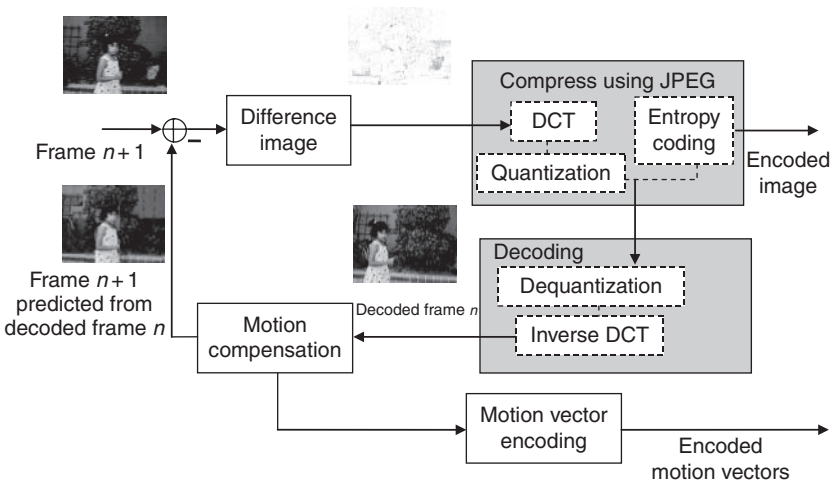


Figure 8-9 Closed loop motion compensation. The encoder simulates a decoder and decodes frame n. The decoded version of frame n is used to predict frame n + 1. The resulting reconstruction at the decoder does not accumulate errors.

```
referenceFrame = NULL;
While input frames are available
{
        currentFrame = getNextFrame();
        if (intramode)
        {
                codedFrame = codeFrame (currentFrame); //Using JPEG pipeline
                dumpToStream (codedFrame);
                reconstructedFrame = decodeFrame (codedFrame);
                referenceFrame  = reconstructedFrame;
        }
        else if (intermode)
        {
                for each macroblock in image
                {
                        // motion vector computed for each macroblock
                        mVs[i] = computeMotionVector(macroblock, referenceFrame);
                }

                // Create a motion compensated frame by copying blocks from reference
                // frame using computed motion vectors
                predictedFrame = compensateFrame(mVs, referenceFrame);

                // Compute the error Image
                errorFrame = currentFrame - predictedFrame;

                codedFrame = codeFrame(errorFrame); //Using JPEG  pipeline
                codedMvs = codeMotionVectors (mVs);  // entropy code motion vectors
                dumpToStream (codedMvs);
                dumpToStream (codedFrame);
        }
}
```

Figure 8-10 Pseudocode at the encoder

The pseudocode executions at the encoder and decoder in a closed loop are shown in Figures 8-10 and 8-11. Note that the encoding process is more complex than the decoding process.

2 TYPES OF PREDICTIONS

So far, we have seen that the compression of video frames occurs either in intramode, which follows the standard JPEG like pipeline for compressing an image, or in intermode, where the frame is predicted based on a reference frame. Mostly, the reference frame used to predict the current target frame is chosen to be the immediately

```
referenceFrame = NULL;
While encoded data is available
{
        currentCodedFrame = getNextCodedFrame();
        currentCodedMvs = getNextCodedMotionVectors();

        if (intramode)
        {
                frame = decodeFrame (currentCodedFrame); // Using JPEG pipeline
                referenceFrame = frame;
                sendToDisplay (frame);
        }
        else if (intermode)
        {
                errorFrame = decodeFrame (currentCodedFrame);
                        // Using JPEG pipeline
                mVs = decodeMotionVectors (currentCodedMvs);
                predictedFrame = compensateFrame (mv, referenceFrame);
                frame = predictedFrame + errorFrame;
                sendToDisplay (frame)
        }

}
```

Figure 8-11 Pseudocode at the decoder

preceding frame. In some instances, it makes sense to use a future frame as well, or a combination of a future and a past frame to predict the current frame. In the following sub sections we discuss the different types of frames generated during encoding—I, P, B as well as multiframe predictions. The older standards use only I and P frames, whereas recent standards use all of them. The choice of where to insert I frames, P frames, and B frames is application specific. To aid such applications, video structure is frequently imposed, which is also discussed next.

2.1 I Frames

These are intraframe coded where only spatial redundancy is used to compress that frame. An I frame is, therefore, encoded as a single image with no reference to any past or future frames. Consequently, this frame requires more bits for compression, compared with other predicted frames. Frames are coded using the standard JPEG pipeline, where the frame is divided into 8×8 blocks with each block transformed using the DCT followed by DPCM encoding of the DC coefficient and zigzag run length encoding of the AC coefficients. Please refer to Chapter 6 for the detailed explanation of the JPEG encoding process.

I frames are interspersed periodically in the encoding process, and provide access points to the decoder to decode the succeeding predicted frames. I frames require

more bits to compress than predicted frames. Hence, it may be natural to assume that predicted frames are always preferred during encoding. However, the cost of computing motion vectors in predicted frames might far exceed the bits available to compress the current frame, especially in real-time scenarios. In such cases inserting I frames is preferred, if the bandwidth allows for the extra bits.

2.2 P Frames

P frames are predictive coded, exploiting temporal redundancy by comparing them with the immediately preceding frame, which is known as a reference frame. This preceding reference frame might have been coded as an I frame or even a P frame. Coding P frames is similar to the process explained in Section 1, where the frame is predicted from the previous reference frame on a macroblock-by-macroblock basis by searching for the best motion vectors. The motion vectors are used to create a predicted frame. The error frame created by the difference between the reconstructed frame and the current frame. This error frame is bound of have lower entropy. Coding a P frame involves coding the motion vectors losslessly and coding the error frame using the JPEG pipeline. However, because the error frame has lower entropy, the coding uses higher-valued quantization tables. Typically, the quantization table used for an I frame is quadrupled and used for a P frame. P frames induce a sequential forward dependency during coding, as shown in Figure 8-12. They are expensive to compute, but are necessary for compression. An important problem the encoder faces is when to stop predicting using P frames, and instead insert an I frame. An I frame needs to be inserted where P frames cannot give much compression. This happens during scene transitions or scene changes, where the error images are high. It is pointless to go through a P frame computation only to find a high error enforcing an I frame. Compensating for unnecessary P frames at scene changes can be avoided if a scene change can be efficiently detected before starting P frame computation. Exercise 10 at the end of this chapter deals with this issue.

2.3 B Frames

B frames, also known as bidirectionally coded frames, are intercoded and also exploit temporal redundancy. They differ from the preceding P frames by using

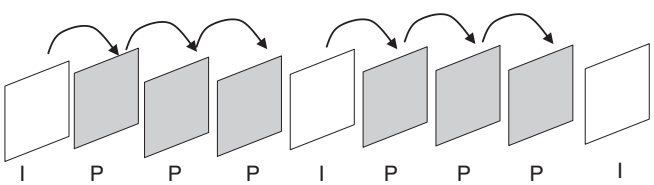


Figure 8-12 Interframe dependency in P frames. P frames are always forward predicted from a previous I frame or a P frame.

two reference frames instead of one. To predict a B frame, the previous or past frame and the next or future frame are used. Sometimes, areas of the current frame can be better predicted by the next future frame. This might happen because objects or the camera moves, exposing areas not seen in the past frames. Using both frames, therefore, increases the correctness in prediction during motion compensation. Note that the past and future reference frames can themselves be coded as an I or a P frame. For example, three frames are shown in Figure 8-13. The camera moves from left to right attempting to follow the foreground object. The middle target frame $n + 1$ is predicted as a B frame using reference frames n and $n + 2$. You can see that there are macroblocks in this frame to the left, which are better predicted by the past reference frame n. Macroblocks to the right, however, are better predicted using the future reference frame $n + 2$. The predicted B frame generated is quite similar to the actual target frame, yielding a lower-entropy difference image when compared with the ones obtained during a P frame computation, leading ultimately to better coding efficiency. However, there is a cost associated with their usage in terms of complexity at the encoder as well as reordering frames and delays during transmission.

The coding of B frames is more complex compared with I or P frames with the encoder having to make more decisions. To compute a matching macroblock, the encoder needs to search for the best motion vector in the past reference frame and also for the best motion vector in the future reference frame. Two motion vectors are computed for each macroblock. Ultimately, the macroblock gets coded in one of three modes:

- Backward predicted using only the past frame
- Forward predicted using only the future frame
- Interpolated, using both by averaging the two predicted blocks

The case corresponding to the best macroblock match and yielding the least entropy in the difference is chosen. B frames, therefore, might need to encode up to two motion vectors per macroblock. Searching in two frames, although useful for compression, needs more computation time.

B frames also necessitate reordering frames during transmission, which causes delays. The order in which frames arrive at the encoder is known as the display order. This is also the order in which the frames need to be displayed at the decoder after decoding. B frames induce a forward and backward dependency. The decoder has to have both the past and the future reference frames ready before it can decode the current B frame. Consequently, the encoder has to encode and send to the decoder both the future and past reference frames before coding and transmitting the current B frame. This enforces the encoder to make the coding and transmission order different from the display order. Because of the change in the order, all potential B frames need to be buffered while the encoder codes the future reference frame, imposing the encoder to deal with buffering and also causing a delay during transmission. Figure 8-14 shows an example of this.

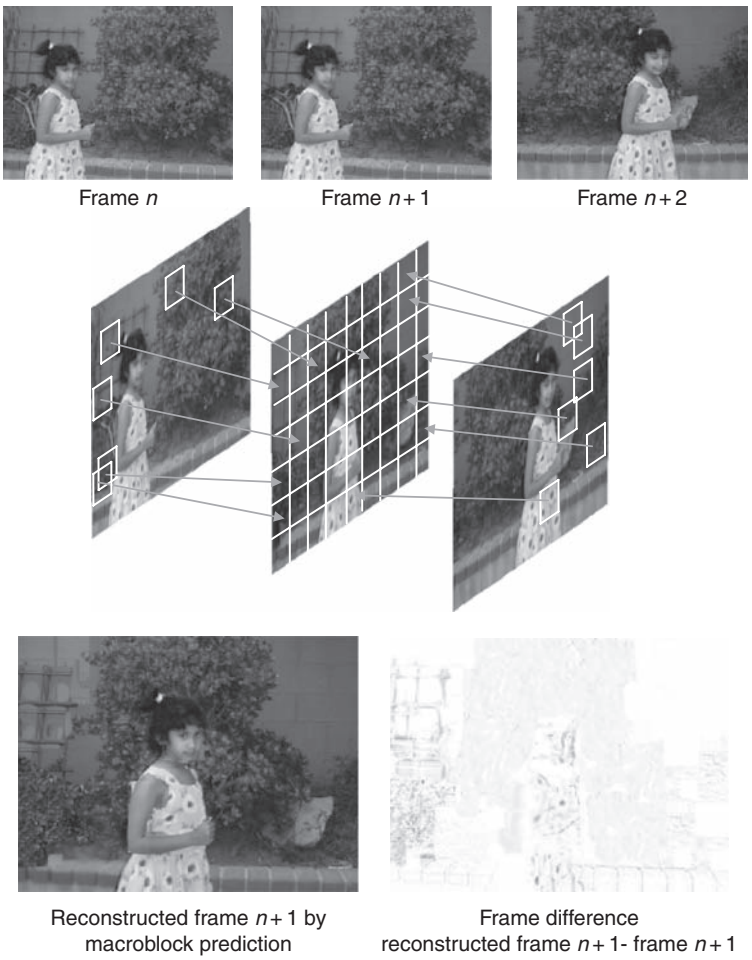


Figure 8-13 B frame computation. Three frames, n, n + 1, and n + 2, are shown where n + 1 is to be predicted. Macroblocks to the left of the frame n + 1 are better predicted by frame n, whereas those to the right are better predicted by frame n + 2. The reconstructed frame using macroblocks from the two frames is shown at the lower left with the difference image shown at the lower right. Compared with the P frame difference image in Figure 8-5, the B frame difference has less entropy. See the color insert in this textbook for a full-color version of this image.

2.4 Multiframe Prediction

Unlike P frames, which use just one past reference frame, or B frames, which use one past and one future reference frame, better predictions can be obtained by searching for macroblock matches using multiple frames in the past or future. For example, a B frame now need not be bidirectionally predicted, but can be bipredicted or multipredicted using two or more frames from the past or future as

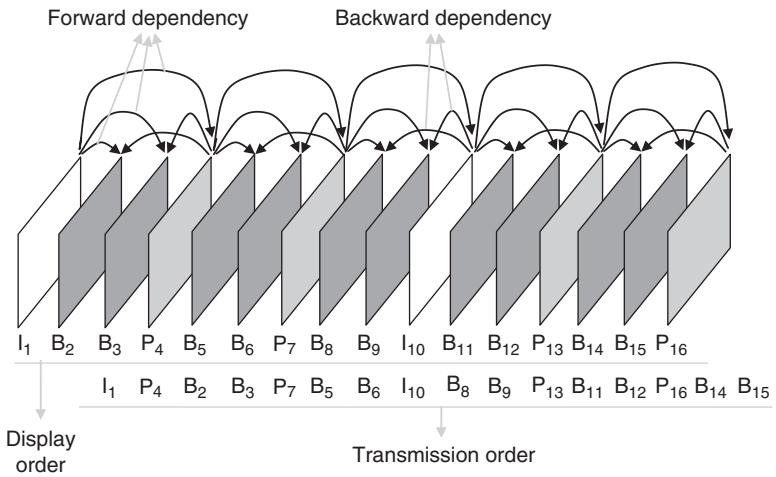


Figure 8-14 B frame predictions. A sequence of frames is shown encoded as I, P, and B frames. B frames have forward and backward dependencies. This imposes a change in the transmission/coding order shown on the bottom line.

illustrated in Figure 8-15. This increases the search time and space substantially, but can give better predictions and, hence, lower bit rates. With the availability of faster computers, multiframe predictions are now gaining popularity in recent standards.

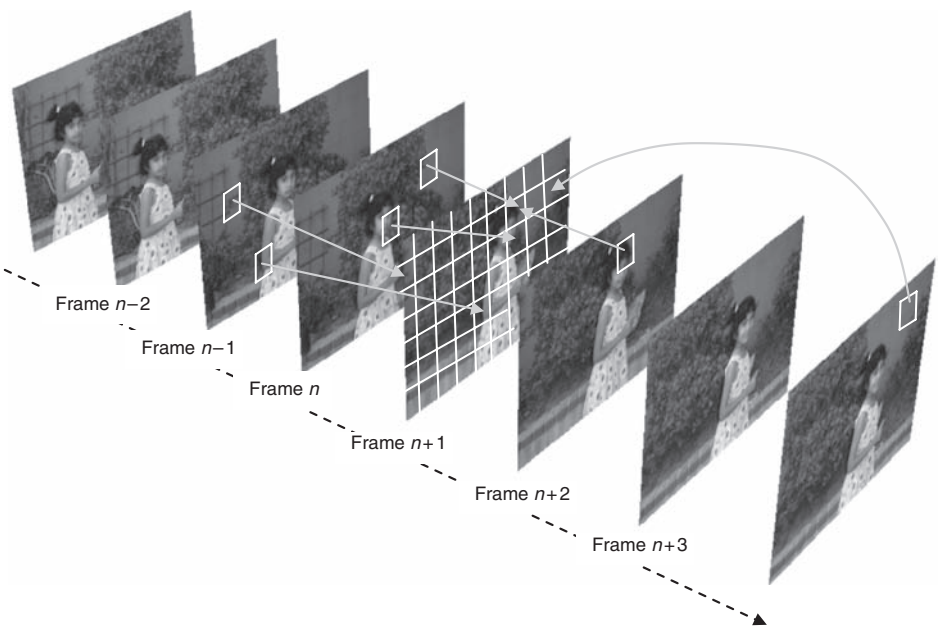


Figure 8-15 Multiframe prediction. A number of reference frames from the past and the future are used to predict frame n + 1. Regions from other multiple frames can do a better job at prediction.

2.5 Video Structure—Group of Pictures

With the usage of the different coding frame types, a variety of video-coding standards, such as MPEG, have adopted a hierarchical description to coded sequences known as a GOP or *group of pictures*. A GOP is a series of one or more frames to assist in random access of the frame sequence. The first coded frame in the group is an I frame, which is followed by an arrangement of P and B frames. The I frames and P frames are known as *anchor* frames and are used to predict B frames. An example is illustrated in Figure 8-16.

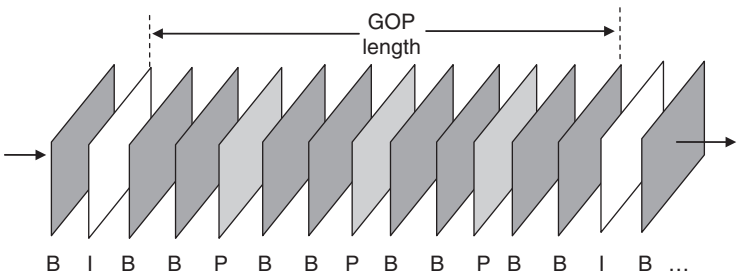


Figure 8-16 Group of Pictures (GOP). An example of a GOP used in MPEG-1 is shown. Here, the GOP length is 12 with three P frames between I frames and two B frames between P frames.

The GOP length is defined as the distance between I frames. Standard codecs represent a GOP with two parameters N and M. N is the size of the GOP that gives the number of frames in the GOP. The distance between the anchor frames (I and/or P) is specified by M. For the example shown in Figure 8-16, $N = 12$, $M = 3$. Alternatively, GOPs are also specified using two parameters that are input to the encoder—"*PbetweenI*", which gives the number of P frames between I frames, and "*BbetweenP*", which gives the number of consecutive B frames between P frames. The group of pictures can be of any length, but must have at least one I frame in any GOP. GOPs are very useful for applications that require random access to the bit stream—for example, fast forward or fast rewind, where the application can randomly access the I frame, decode it, and display it, not worrying about B and P frames. Applications that need support for such features normally have small GOP sizes.

Additionally, each frame might also be divided hierarchically into groups of macroblocks called slices or GOBs. The earlier standards, such as H.261, use the term GOBs, whereas the later MPEG standards use the term *slice*. The reason for defining a frame-level hierarchy is to have the ability to distribute bits more flexibly within a frame. Some slices might be given more bits because of higher entropy, compared with other slices. This allows for variable-length code resetting to prevent error propagation and, consequently, better quality encoding. Figure 8-22 in section 4.3 on MPEG-1 standards shows an example of a frame divided into slices.

The encoded video bit stream can, thus, be looked at hierarchically, as shown in Figure 8-17 with the video frames divided into GOPs. Each GOP is further divided into slices or GOBs. A GOB consists of macroblocks. Each macroblock consists of a differentially coded DC coefficient and runs of variable-length coded AC coefficients.

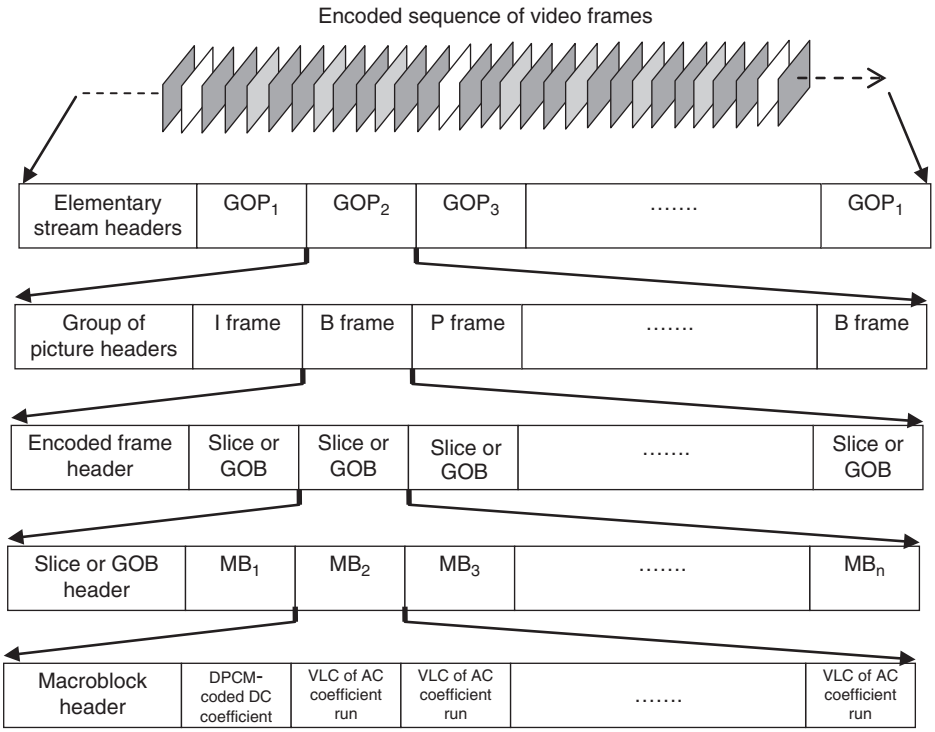


Figure 8-17 Video structure. A typical bit stream structure is hierarchically encoded with GOPs, frames, slices or GOBs, macroblocks, and runs of coefficients.

3 COMPLEXITY OF MOTION COMPENSATION

Given a target macroblock, the motion vector computation proceeds by trying to find the best-matching block in the reference image in a specified search area. There are various candidate motion vectors to choose from. The motion vector for the target macro blocks is chosen to be a motion vector for which the difference between the target block and reference block area is minimal. Given a macroblock of size $n \times n$, the complexity of evaluating the error metric at each position is $O(n^2)$, whether you choose MAD, MSD, and so on. However, the number of positions at which a MAD or MSD is evaluated to decide the motion vector for a macroblock depends on the search techniques used. In the following sections, we analyze some of these techniques. Computing motion vectors is one of the central problems in any video compression pipeline. On one hand, computing good motion vectors is necessary for reducing the entropy of the error image that needs to be compressed. On the other hand, each motion vector computation could take a significant amount of time and an encoder might not always have all the time it needs to compute the best motion vector. On the average, 60% to 80% of the total encoding time is spent just on motion vector search. In practical video-encoding applications, a search parameter k

(which decides the search area) along with the type of search to be performed is specified as input to the encoder, depending on the application scenario, the compression bit rate desired, real-time constraints, and so on.

3.1 Sequential or Brute Force Search

This is also referred to as a full search or brute force search. In a brute force search, the error metric is evaluated at *all* candidate positions. Given a macroblock of size $n \times n$ and search area defined by k, there are $(2k + 1)^2$ candidate motion vector positions at which a MAD needs to be evaluated. A reference macroblock centered at each of the positions is compared with the target macroblock from the target frame. This results in $(2k + 1)^2$ error metric computations, where a pixel-by-pixel difference is evaluated. Figure 8-18 shows an example with a search area specified with $k = 1$. There are nine positions as shown. Because computing one error metric has n^2 computations, the total complexity of this search is $(2k + 1)^2 \times n^2$. The vector that corresponds to the least MAD is taken to be the motion vector for the macroblock in the target frame.

The complexity of this method surely shows the cost associated. As an example, for a standard definition video of size 720×480 at 30 frames per second, a macroblock size of 16×16, and a search size of 31, the numbers can be analyzed as follows:

Number of macroblocks per frame $= 720 \times 480 / (16 \times 16) = 1350$
Number of MSD evaluations for each motion vector $= (2 \times 31 + 1)^2 = 3969$
Total number of MSD evaluation per second $= 161 \times 10^6$
Each MSD evaluation is $O(n^2)$, where n is 16. Assuming each MSD evaluation takes 1 millisecond, this would correspond to 161 seconds.

This certainly does not make it useful for real-time video encoding. However, it does have the advantage that you will choose the best motion vectors resulting in least-entropy error images and, consequently, the best quality for the desired bit rate.

A variety of methods to speed up the motion vector computation are discussed next. All these methods are based on the same principle of selectively checking only a small number of candidate motion vector positions rather than an exhaustive search. This is done assuming that the distortion error monotonically decreases toward the best candidate.

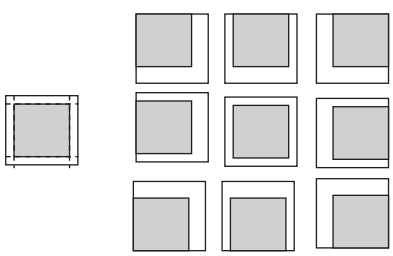


Figure 8-18 Search space example with k = 1. There are nine candidate positions computed as (2k + 1)². In general, the search or the value ranges from 0 to 31 and depends on the motion in the video.

3.2 Logarithmic Search

Logarithmic search works on the assumption that the error distortion metric (MAD or MSD) decreases monotonically in the direction of motion. In this case, the metric is evaluated in an iterative fashion at nine positions during each iteration. In the beginning, given a search area parameter k, the evaluation proceeds by computing the metric at the center of the search area and at eight locations with vectors $(k/2, k/2)$, $(0, k/2)$, $(-k/2, k/2)$, $(-k/2, 0)$, $(-k/2, -k/2)$, $(0, -k/2)$, $(k/2, -k/2)$, and $(k/2, 0)$. These are used as seed locations to come up with an initial match at the minimum MAD position. At the next iteration, the center of the new search region is moved to it, and the search step size is reduced to half. At the second iteration level, the step size is $k/4$ and the iterative evaluation proceeds until the step size is 1, at which point the best match yields the best motion vector. Figure 8-19 shows a sample search with a search parameter $k = 16$.

The logarithmic search described here is similar to binary search and takes $log_2 k$ iterations. In the example shown in Figure 8-19, the search terminates in four iterations. This means $4 \times 9 = 36$ MAD evaluations. In general, though, we do not need to compute nine, but only eight evaluations from the second iteration onward because one of the computed MAD locations is used as the center for the next search. This search technique proceeds fast, and is also termed as fast motion estimation (FME). However, if the monotonic assumption does not hold in the pixel content, it can produce suboptimal results.

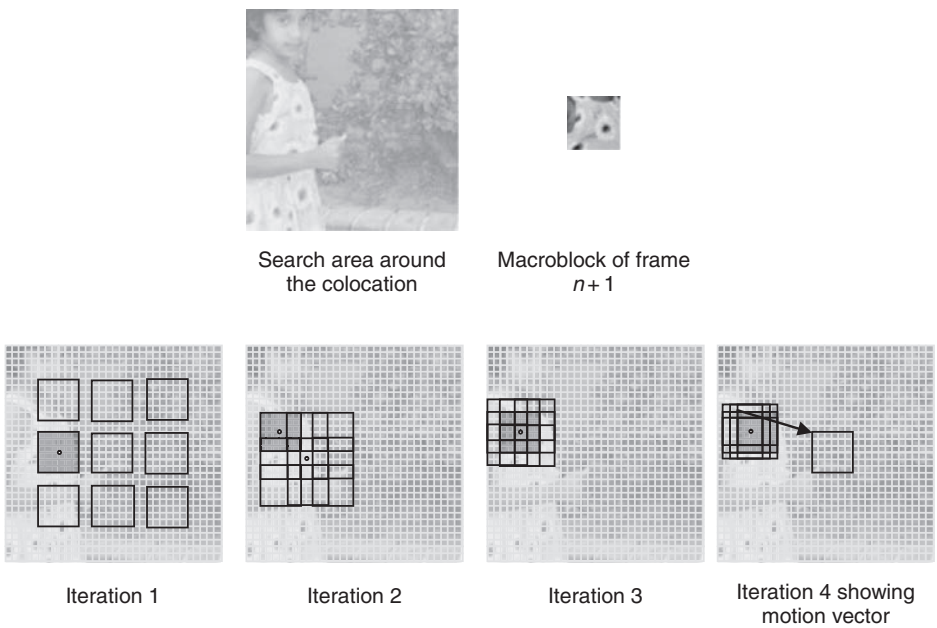


Figure 8-19 Logarithmic search. The top row shows the macroblock whose best match needs to be searched in the search area with k = 16 of frame n. The bottom row shows four iterations of the logarithmic search. At each iteration, the best-matched MAD block is shown shaded. Each iteration converges closer to the best match. See the color insert in this textbook for a full-color version of this image.

3.3 Hierarchical Search

Sometimes, the assumption of monotonic variation of image intensity that is used in logarithmic searches does not hold, especially for larger image displacements. This often causes incorrect motion estimations that do not yield lower entropy errors. Hierarchical searches attempt to be just as fast and do better than logarithmic searches. This is achieved by searching in a multiresolution manner, where a full search is carried out at the highest level of the hierarchy, followed by refinements trickling through the lower levels. The target frame, as well as the reference frame, is said to be at level 1. Successive levels are created by subsampling and 2D filtering the previous level. This creates a pyramid structure for both frames, as shown in Figure 8-20. At each level, the image size reduces by a factor of 2 in both dimensions. This reduction is also true for the macroblock sizes that need to be predicted in the target pyramid as well as the search area parameter k in the reference pyramid.

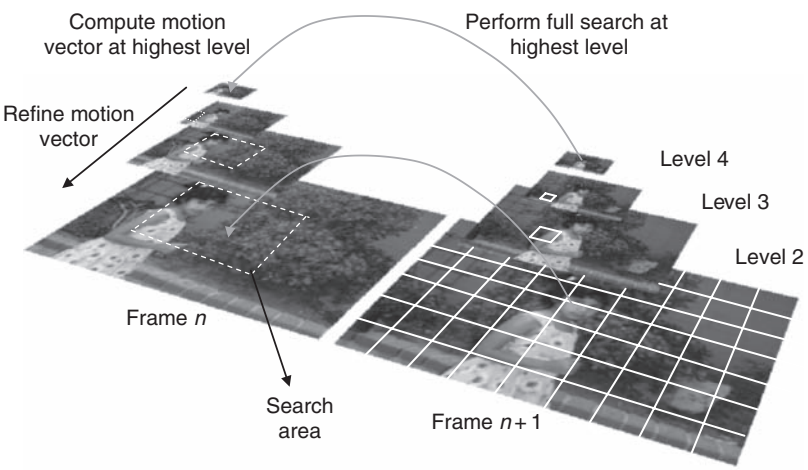


Figure 8-20 Hierarchical motion vector search. A hierarchy of subsampled images is obtained for the target and reference images. A motion vector search for a sample macroblock proceeds by performing a full motion search at the highest level to obtain a motion vector. The motion vector is further refined by using the lower levels in the reference pyramid. See the color insert in this textbook for a full-color version of this image.

Shown at the bottom of Figure 8-20 are level 1 images of the reference frame n and the target frame $n + 1$. The target frame to be predicted is shown broken into macroblock areas. Both the reference and target images are shown subsampled to level 4. An example macroblock is shown at each level in the target hierarchy along with its corresponding search area in each level of the reference hierarchy. Note that the macroblock reduces in size and so does the search area in the reference hierarchy at each level. If the macroblock is of size $n \times n$ and the search area parameter is k to begin with, at level 2 of the pyramid, the reference macroblock will be $n/2 \times n/2$ while the corresponding search area parameter will be $k/2$. A full search is carried out at the

highest level, which proceeds quickly because of smaller macroblock size and smaller search area. This computed motion vector gets refined by comparing it with other missed candidates in the lower levels of the reference hierarchy. You can see from the example in Figure 8-21 that there are eight new candidate positions in the local neighborhood of the motion vector at the new lower level. A MAD is computed at these new locations to see whether the motion vector can be refined to a new position. The refinement process is repeated till the lowest level.

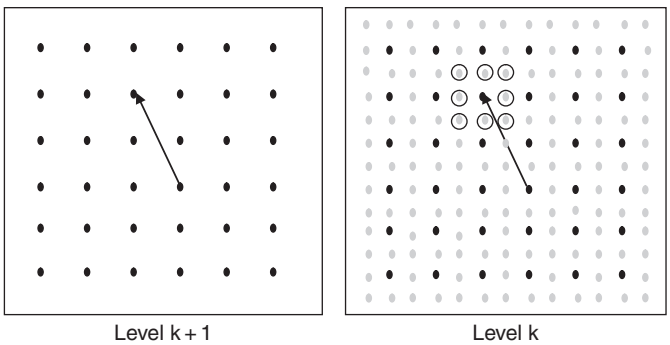


Figure 8-21 Hierarchical search evaluations. The left figure shows a motion vector computed at a higher-level k + 1 in the hierarchy. When transitioning to the lower level k, there are eight new additional locations, where the error might be lower. The motion vector can be refined to one of these eight positions.

An illustration of the complexity can be considered when comparing the hierarchical search with the standard full search. In digital HDTV sports content, the search parameter k is normally kept to 32 (or more) because of higher resolution in the content. A full search would need $(2 \times 32 + 1)^2 = 4225$ MAD evaluations. A four level hierarchical search would result in a $k = 4$ at level 4. A full search at this level entails $(2 \times 4 + 1)^2 = 81$ MAD evaluations. At each successive lower level, there will be another eight refinement positions, bringing an additional $(8 \times 3) = 24$ evaluations since there are three levels at which this is done. The total number of evaluations for a hierarchical search is the sum of the searches done at the highest level and refinements at the intermediary levels bringing the total to $(81 + 24) = 105$.

4 VIDEO-CODING STANDARDS

A variety of video-coding standards have been established. These standards are broadly categorized under two groups—the ITU (International Telecommunication Union), which has developed the H.26x video standard family, and the ISO (International Organization for Standardization), which has developed the MPEG standards. The MPEG community was established in 1988 with a few members but has now grown into a much wider group of more than 400 members representing both the industry and academia. All these

standards define only a compressed bit stream so that any standards-based decoder can decode the bit stream. However, the compression algorithms followed in the encoders is completely dependent on the proprietary technology implemented by the encoder manufacturers. The standards have been developed since 1990. Each standard was initiated with an objective to support an application with the then mature research and development technology. The earlier standards, thus, have very limited scope, whereas the later standards are attempting to have a more ubiquitous presence.

4.1 H.261

H.261 was designed to support video teleconferencing over ISDN lines. The standard was established by the ITU in 1990, when the motion compensation technology was mature enough to effectively deal with compressing video with talking heads having little or no motion. It was mainly designed for the QCIF resolution (176 × 144) but can also support CIF-sized video (352 × 288). It has no support for interlaced video. H.261 is also sometimes referred to as *px*64 compression because it was intended for usage on the ISDN networks, which support data rates at multiples of 64 Kbps depending on the quality of desired transmission and available bandwidth support.

Compression occurs by compressing frames in the intramode (I frames) and in the intermode (P frames). There is no support for B frames. I frames are compressed in a JPEG like manner using no references.

4.2 H.263

The H.263 standard was designed by ITU to extend the H.261-based videoconferencing and support a wider range of picture formats, including 4CIF (704 × 576) and 16CIF (1408 × 1152). The video-coding H.263 standard along with the G.723 speech-coding standard formed the H.324 media standard for communication over telephone networks. The video-coding part was to be supported on maximum available bit rates of 33.6 Kbps and normal available bit rate of 28 Kbps.

Like the H.261, it uses I and P frames, which are both encoded in the standard manner. However, it also used B frames. B frames provide a lot more compression by using a past and a future reference frame to better predict macroblocks. However, B frame usage suffers from delays in encoding and reordering of transmitted frames, as explained in Section 2.3.

4.3 MPEG-1

MPEG-1 was the first series of audio-video coding standards established by MPEG (Moving Picture Experts Group) and approved by the ISO in November 1991. It consists of many parts, including a systems, video bit stream, and audio bit stream specification and conformance. MPEG-1 video coding was designed to aid storage of noninterlaced digital video at 1.5 Mbps. MPEG-1 video encoding was used in the VCD format, which was a first attempt to get digital video quality comparable to analog VHS video. MPEG-1 is designed to encode SIF-sized frame resolutions at 1150 Kbps. The picture quality overall does compare to VHS video, but visual artifacts are noticeable when there is high

motion in frames. The VCD format never gained a foothold in Northern America and Europe because of low-cost reproduction leading to piracy issues, but they have become popular throughout Asia. Some of the salient points of MPEG-1 are as follows:

- Video compression uses intra- and intermodes, where the intermodes use both P and B frames.
- One distinguishing feature of the MPEG family when compared with H.261 and H.263 is the usage of slices while encoding predictive frames. A frame is divided into slices of macroblocks where each slice might contain a variable number of macroblocks, as shown in Figure 8-22. Each slice is encoded independently with different quantizer scales. This provides additional freedom and flexibility during rate control.

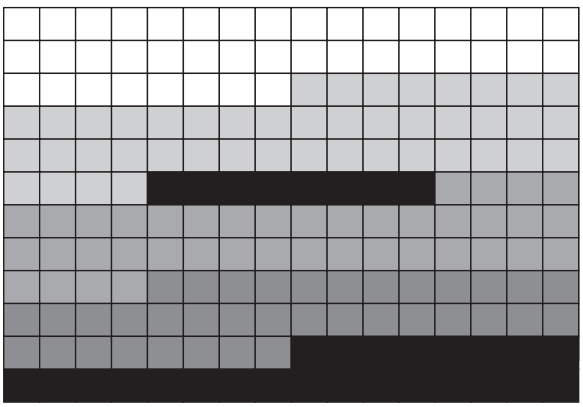


Figure 8-22 Slices in an MPEG-1 frame. Each slice is coded independently and can be quantized differently depending on the entropy of the slice.

- MPEG-1 uses subpixel (half pixel) motion compensation with motion vectors generated using bilinear interpolation methods. This increases the motion compensation precision, further reducing entropy in the difference images.
- Random access of frames in MPEG-1 is allowed for applications such as trick play (fast forward and rewind). The frames are organized as GOPs and the decoder can seek to any frame position using the GOP hierarchy.

4.4 MPEG-2

MPEG-2 defines the next set of audiovisual formats from the MPEG community. Unlike MPEG-1, MPEG-2 was designed for higher bandwidths and higher quality supporting bit rates from 4 to 9 Mbps, which was initially adopted for broadcasting digital television. Over time, the standard and the bit rate support has been expanded to include high- resolution video in HDTV. MPEG-2 was ratified in 1994. Like MPEG-1, it provides support for compressed audio and video bitstreams, but in addition it also provides support for systems level bitstreams. The systems level defines a *program stream* format and a *transport stream* format. Both of these are container formats that take packetized

elementary streams (PES), which are formatted compressed video and audio bitstreams, and multiplex them into a single stream for different purposes. The program stream has been defined for storage on reliable media such as disks, while the transport stream is designed for transmission where loss of data and delays are possible and synchronization needs to be maintained. A considerable amount of infrastructure and investment has gone into MPEG-2 to broadcast digital TV over satellite and cable networks. Additionally, the video part has also been adopted into the popular DVD (digital versatile disc or digital video disc) formats. Some of the salient features of MPEG-2 video coding are as follows:

- MPEG-2 encodes video using I, P, and B frames similar to MPEG-1 with half-pixel approximation for motion vectors.
- MPEG-2 has support for interlaced video. Interlaced video, as explained in Chapter 2, consists of alternating fields. Consequently, during intermode compression, fields need to be predicted from fields. For this, MPEG-2 defines different modes of prediction for predicting frames from frames, fields from fields, and fields from frames.
- In addition to defining video and audio, MPEG-2 was also designed as a transmission standard providing support for a variety of packet formats with error-correction capability across noisy channels.
- One new aspect introduced in MPEG-2 video was scalable encoding, which generates coded bit streams in such a way that decoders of varying complexities can decode different resolutions and, thus, different quality from the same bit stream. This allows for encoders to encode once and support different bandwidths, having end clients with decoders of different complexity. The scalability is achieved as shown in Figure 8-23, where the input video is preprocessed in a base layer and an enhancement layer. The two layers are coded independently. Thus, a decoder having standard capabilities can choose to ignore the enhancement layer, or, if it does have higher capabilities, it can decode the enhancement layer to get better video quality.

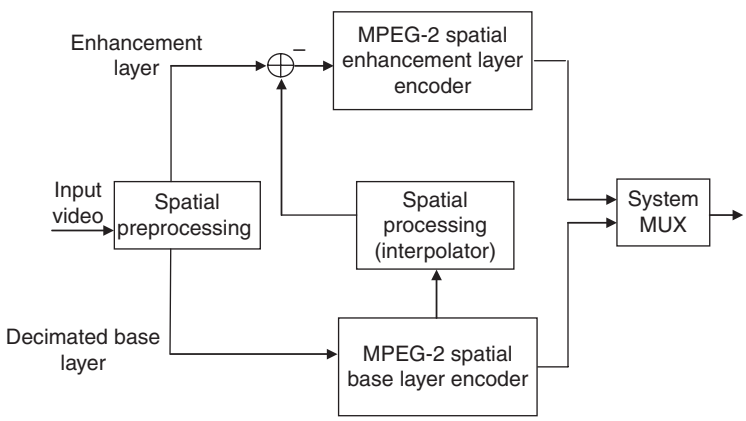


Figure 8-23 Scalable MPEG-2 video encoding. The input video is broken into two layers, a base layer containing low frequency and an enhancement layer containing high frequency. Both layers are coded independently and then multiplexed into the coded bit stream.

4.5 MPEG-4—VOP and Object Base Coding, SP and ASP

MPEG-4 is one of the latest encoding standards from the MPEG community and has a much broader scope than just encoding video or audio. It is a true multimedia format that has video, audio, 2D/3D graphics, and systems levels parts. Unlike the previous formats that dealt with only one video and only one audio stream audio-video streams, MPEG-4 deals with the organization of content into media objects that are encoded as compressed object streams. Much of the details of this standard as a whole are discussed in Chapter 14 of part 3 in this book. Here, we briefly talk about the new details that pertain to the video part.

The earlier versions of the video part and the associated profiles were finalized and ratified in the first half of 1999; however, the later versions such as MPEG-4 version 10 were completed later in 2003. Version 10 of the visual part is also known as MPEG-4 AVC (Advanced Visual Coding) and is similar to H.264. This standard is discussed in the next section. MPEG-4 arose from a need to have a scalable standard that supports a wide bandwidth range from low bandwidth to high bandwidth. Examples include the support of streaming video at less than 64 Kbps suitable for Internet applications, wireless handheld devices, as well as upward of 4 Mbps for higher-bandwidth broadcasts. Some of the advancements supported by MPEG-4 video when compared with the previous standards include the following:

- It supports both progressive and interlaced video encoding.
- The video compression obtained by MPEG-4 ASP (Advanced Simple Profile) is better than MPEG-2 by 25% for the same video quality. This is because MPEG-4 does quarter-pixel accuracy during motion prediction, along with global motion compensation.
- The standard is object based and supports multiple video streams that can be organized (along with other audio, graphics, image streams) into a hierarchical presentation using scene layout descriptions.
- MPEG-4 supports coding video into video object planes. A video can consist of different video object planes (VOPs), and the image frames in each VOP can have arbitrary shapes, not necessarily rectangular. This is useful for sending two video feeds to the client, who can then composite them for a MPEG-4 player. For example, one video could be of a reporter in a studio chroma-keyed in front of a blue screen, while the other could be that of an outdoor setting. The composition of both occurs at the client side. This increases the flexibility in creating and distributing content.
- MPEG-4 compression provides temporal scalability, which utilizes advances in object recognition. By separating the video into different layers, the foreground object, such as an actor or speaker, can be encoded with lower compression while background objects, such as trees and scenery, can be encoded with higher compression.
- MPEG-4 also provides a synchronized text tract for courseware development and a synchronized metadata track for indexing and access at the frame level.

MPEG-4 can be viewed as a new paradigm in the way media is encoded and distributed when compared with its predecessor standards. Chapter 14 devoted to the MPEG-4 standard where the media object framework is explained in more detail.

4.6 H.264 or MPEG-4—AVC

The H.264 standard aimed to create video quality at substantially lower (less than half) bit rates than the previous standards, such as MPEG-2, H.263, and so on, without increasing the design complexity by much. This latest standard has been ratified in 2004, but proposals were called for as early as 1998 by the Video Coding Experts Group (VCEG). The project was then called H.26L, but in 2000 VCEG and MPEG-4 video committee formed a Joint Video Team (JVT) and finalized this standard, which is now known as H.264 and also as MPEG-4 Advanced Video Codec (AVC). H.264 was designed for technical solutions addressing a variety of applications such as broadcast entertainment over cable and satellite, high-quality interactive storage on magnetic devices, video-on-demand over DSL/cable modems, and MMS over Ethernet/LAN/wireless. Additionally, it has also been adopted into the Blu-ray and HD-DVD (now defunct) formats, which are high-definition followers of the standard DVD format.

Although the main algorithm is very similar to the generic motion compensation algorithm, there are a few salient variations, which allow for the reduction in the bandwidth. These are as follows:

- During intraframe coding or coding of error images in intermode, although most previous standards only use 8×8 blocks, H.264 provides for smaller block usage up to 4×4.
- Additionally, the intracoding is coded more effectively by organizing the blocks into slices. Each slice uses directional spatial prediction. Rather than DCT coding each block, the blocks in each slice are predicted from the previous block and only differences are DCT coded. This is done to further exploit the spatial redundancy when present.
- H.264 allows variable block–sized motion compensation. Whereas most previous standards used 16×16 fixed sized macroblocks to predict a target frame, H.264 allows the macroblock to be broken down hierarchically into macroblocks of sizes 8×16, 16×8, and 8×8 blocks. The 8×8 block can be further broken down into 8×4, 4×8, and 4×4 blocks. This is illustrated in Figure 8-24. As Section 1.4 explains, macroblock size is very important to motion prediction. The ability to adjust this size depending on how the content has moved allows for better frame prediction, especially at the boundaries of the objects in motion, and, consequently, for lesser entropy in the error image. However, the search algorithm does get more complex.
- The standard performs quarter pixel motion compensation (QPEL). This subpixel approximation of motion vectors tends to better predict motion blocks yielding even fewer errors and, hence, lower bit rates.

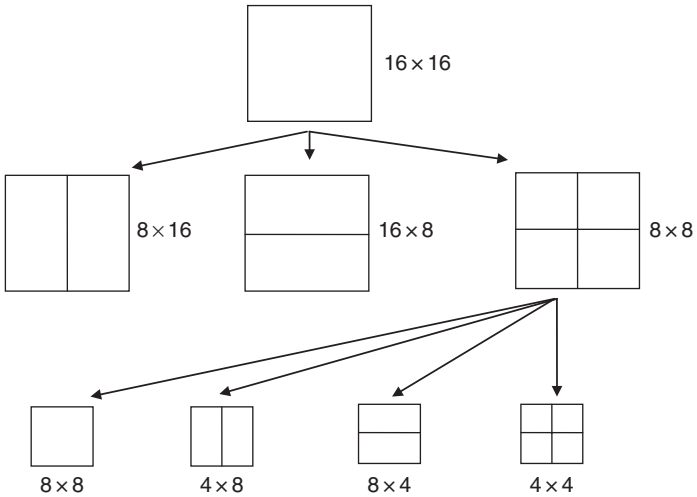


Figure 8-24 Hierarchical breakdown of macroblocks used for motion compensation in H.264. The top macroblock is a 16 × 16 used in MPEG-2. For H.264, this is approximated by 8 × 16, 16 × 8, 8 × 8, 4 × 8, 8 × 4, and 4 × 4 blocks.

- H.264 provides for multiple reference picture motion compensation. In other words, during frame prediction you are not limited to using the immediate previous reference I or P frame for motion vector searches. You can use any frame(s) from the past or future that your buffer can hold. This increases the likelihood of better matches while predicting macroblocks. This essentially relaxes the naming of frames as P or B. Instead, frames are singularly predicted, bipredicted, and so on using both past and future frames.
- The entropy coding done on motion vectors as well as the quantized error images uses context-adaptive binary arithmetic coding (CABAC). The previous standards used Huffman or arithmetic coding where the codes are predesigned based on statistical probabilities of symbols occurring in a large sample set. In H.264, CABAC assigns codes based on probabilities of symbols in the context of the current data stream and, hence, generates better entropy-coding bit rates.
- The H.264 standard also provides an in-loop deblocking filter at the encoder and the decoder. It helps prevent the blocking artifacts common to DCT-based image compression techniques and results in better visual appearance as well as better compression. The filter operates on the edges of each transform block while compressing I frames or error images in predicted frames. The edges of the small block (8 × 8 or 4 × 4) are assigned a boundary strength, which gives a measure of distortion at the block boundaries. Stronger levels of filter are applied to the pixels around the edges depending on the edge's strength.

5 VBR ENCODING, CBR ENCODING, AND RATE CONTROL

Digital video is used now in a variety of applications requiring storage and distribution. The standards discussed previously have been put into place to make some of these technologies viable. We have also seen that the compression of digital video is a combination of predictive and statistical coding processes whose main objective is to reduce the amount of bits used in the coded bit stream. However, depending on the content and application requirements, the number of bits generated keeps on varying over different intervals of time. For example, when there is little or no motion in a few consecutive frames, the P and B frame predictions are more accurate, yielding lesser entropy in the difference images. In addition, motion vectors are close to zero when there is no motion, resulting in better motion vector compression. However, when there is high motion over a few frames, the number of bits required to code those sets of frames increases. High motion frames yield nonzero motion vectors, which require more bits during entropy coding. Additionally, the P and B frame predictions tend to have more error and, hence, higher entropy, requiring more bits to compress. In general, any video content is full of slow- and fast-motion segments scattered throughout, requiring varying number of bits over time during compression, as illustrated in Figure 8-25. Thus, video encoding normally would generate a variable bit rate (VBR) for storage or transmission reasons.

VBR encoding gives better quality video with lower distortion because bits are provided as required. In most storage applications that make use of VBR, there is no stringent control enforced on the number of bits that can be used. For example, when encoding video for a DVD or for storage on your hard disk, there may be an upper limit on the total bits that are used because of the capacity of the DVD, but the bits may be distributed nonlinearly with respect to time as required by high- and low-motion frames.

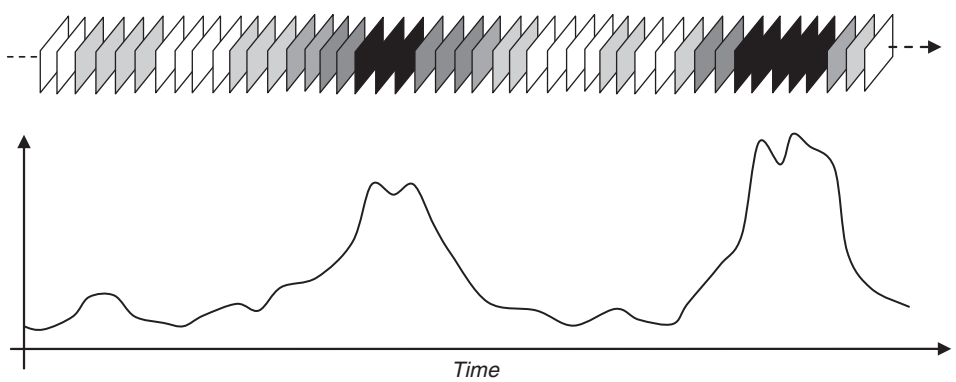


Figure 8-25 Bit budget graph. The top frames are color coded to show motion activity. The darker the frame, the more motion. The lower graph shows the bit requirements over time. In high-motion areas, more bits are required during compression compared with low-motion frames, inherently creating VBR. A CBR output would tend to more regularly distribute bits, making the graph more flat.

The whole equation changes when encoding video for streaming reasons. The video can be encoded statically and streamed at a later time, or it can be encoded in real time. The distribution of bits needs to be even more controlled for real-time streaming in live video presentations. This distribution of bits is known as rate control and poses a number of interesting issues.

There is a network bandwidth requirement that the encoder needs to adhere to. For example, if CIF-size digital video is encoded at 1 Mbps and streamed to a computer with 256 Kb connection, the client will not be able to receive data and keep up with the display frame rate. The encoder has to encode the video for a specified bandwidth. Normally, bandwidth requirements are defined by the video quality desired for predefined video profiles. Video profiles specify video frame sizes and frame rates. For example, standard-definition digital television (SDTV) is compressed up to 2 Mbps using MPEG-2, whereas high-definition digital television (HDTV) is compressed at 15–17 Mbps using MPEG-2.

When video is streamed to a client, it is imperative to have an uninterrupted and smooth viewing experience at the client's end. The client in such cases has to decode data and display the data at the required frame rate—24, 25, or 30 fps for film, PAL, or NTSC formats. This means that the data coming to the decoder should on average contain the required number of encoded frames per second. One simple way to solve this problem is to fix the frame type sequence (GOP) at the encoder and allocate a total number of bits within a GOP assuming a known target bit rate. We can fix the number of bits required for an I frame, for a P frame, and for a B frame knowing that I frames statistically take 3 times the bits as P frames, which, in turn, take between 2 and 5 times that of B frames. During encoding, each frame is quantized so as to meet its target rate. This simple solution does meet the CBR constraints but fails to produce good-quality encoding because it does not effectively make use of the fact that video content's complexity can vary with time.

When the video complexity changes with time, it might be desirable to devote more bits to higher-entropy parts of the sequence than the lower-entropy parts, allocating a different number of bits to each frame. Another alternative is to allow the number of bits per I, P, and B frames to *adaptively* vary from each GOP. As in the preceding case, the total number of bits for a GOP is decided assuming a known target bit rate. During encoding, the GOP frames are cached at the encoder and analyzed for motion activity and texture. If there is less motion and high texture during a certain time instance, a greater proportion of the bits are assigned to I frames. Similarly, if the motion is high, the bits assigned to P frames should increase. Most adaptive encoders encode B frames with low quality to allow the anchor I and P frames to be coded at the best quality possible. The adaptive allocation of bits in general is a difficult problem, and commercial encoders use their own proprietary architectures and heuristics to arrive at better quality coding.

In addition, if CBR has to be maintained in a changing bit allocation scenario, the decoder has to have a buffer that stores bits not needed to decode the immediate picture. The size of this buffer directly governs the freedom by which the encoder can vary the distribution of bits. If the buffer is large, the encoder can use larger variation, increasing the coding quality. However, this comes at the computational cost of increasing the decoding delay. Thus, when performing rate control, the encoder and the decoder are closely tied.

6 A COMMERCIAL ENCODER

An encoding session involves the use of software and/or hardware elements that take video frames as input, along with parameters that control the encoding process. Many times, to get the desired results, the video frames are preprocessed using a variety of low-pass filters. Video captured using devices frequently have noise due to quantization and sampling. During encoding, the high-frequency noise effects, if not removed, manifest themselves in nonzero high-frequency coefficients, which, in turn, increase the entropy of the run length coded AC coefficients in blocks. Another example of the need to preprocess before encoding occurs while digitally remastering older movies into digital formats. The older movies are available on film, which is scanned into a digital image frame by frame. The scanning process introduces artifacts of dust and other particles on the scanning screens into the image causing high-frequency content. In addition, there is inherent film grain noise, which gets amplified during the conversion. All these reasons require filtering prior to encoding. This reduces bit rate and increases compression quality.

The encoding process can be controlled by a few parameters. Some of the parameters are shown in Figure 8-26. The top GUI shows basic parameters exposed to novice users where the user can control the target bit rate of encoding as well as qualitative parameters such as *quality, recovery*, and *complexity* and the lower GUI shows advanced parameters. The basic parameters ultimately control the advanced parameters, which require more technical understanding of the encoding process and are normally adjusted only by expert users. For example, the *quality* parameter controls the visual encoding quality. Changing this internally adjusts the high-level parameters such as the quantization factor and the search window. Similarly, the recovery parameter defines how easily the encoding allows error recovery. The error *recovery* is better with more I frames inserted. Consequently, *recovery* parameter controls advanced parameters such as the *P-Between-I* and *B-Between-P* parameters. The parameters are explained as follows:

- *Skip Frame Index*—Present in some encoders, this parameter allows the user to skip the suggested number of frames between two frames to be encoded. This reduces the effective frame rate and is used for low-bandwidth applications. For example, video captured at 30 fps can be delivered at 15 fps or lower to allow continuous reception at lower bandwidths. The other parameters allow control over the quality of the encoding.
- *P-Between-I*—The P-Between-I parameter gives the number of P frames between any two I frames. By increasing this value, the GOP size increases. For slow and medium motion video, this value might be large. However for fast-motion video, this value tends to be low because motion prediction is harder to get right.
- *B-Between-P*—The B-Between-P parameter controls the number of consecutive B frames between P frames. B frames give more compression, although they take longer to compute. The use of the number of B frames is a compromise between the bit rate needed and the time needs for the compression.

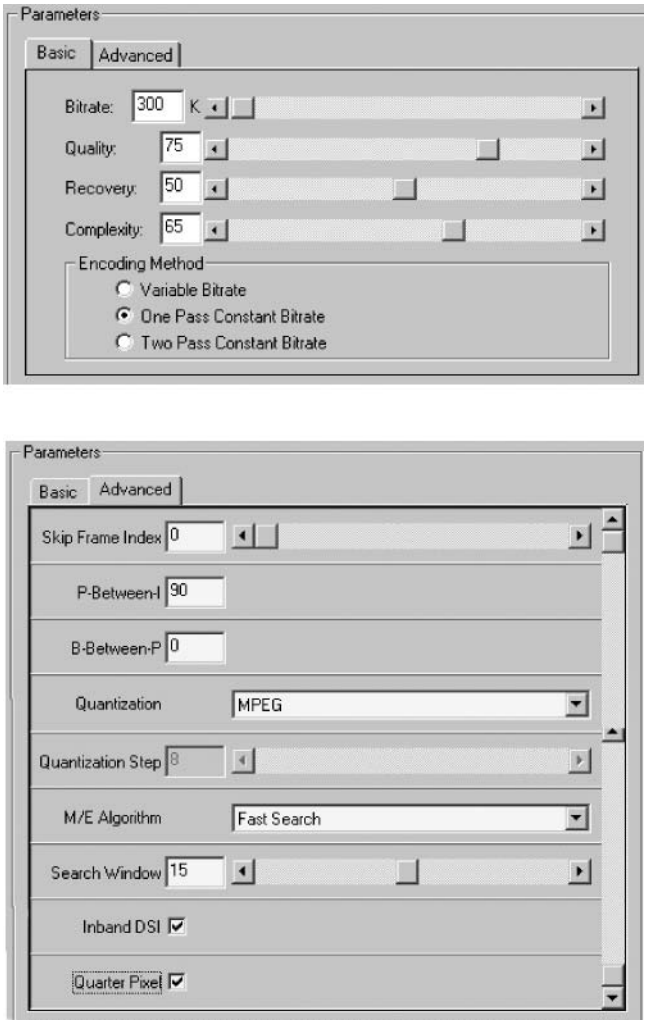


Figure 8-26 A sample GUI showing parameters that control the processing in an encoder. The top GUI shows simple, basic parameters exposed to encoding novices, whereas the bottom GUI shows advanced parameters exposed to encoding experts.

- *Quantization table*—This is the table that the DCT blocks of the I frame or the error image DCT blocks of P and B frames use to quantize the coefficient values during compression. The ISO formats and the ITU formats have arrived at different quantization tables. The example in Figure 8-26 shows the use of MPEG tables. Other parameters include H.263 tables, and so on.
- *Quantization Step*—This parameter gives the amount by which the base values in the quantization table are scaled up during compression. A higher number

here increases the quantization intervals, thus increasing the compression and also the distortion. This parameter is similar to the operation that the "quality factor" performs during JPEG compression.

- *Motion Estimation Algorithm*—This parameter allows the encoder to choose one of many search methods for computing motion vectors. Typical values include *full search*, *fast search* (logarithmic), *hierarchical search*, and so forth.
- *Search Window*—This is the value of the search parameter *k*, which tells the encoder the range in which to search for the matching macroblock. The range of this value is from 0 to 31, which matches the maximum number of bits allocated for a motion vector coordinate.
- *Inband DSI*—DSI stands for Decoder Specific Information. Every encoded video or any media stream (known as an elementary stream) typically has information in its header such as profiles, frame size, and so on that allows the decoder to configure itself prior to decoding. If a client initiates a download, the decoder can read this information from the header of the delivered data and configure itself for the decoding of the subsequent data bits that arrive. However, in a broadcast application like cable television, a client might choose to view the stream halfway during a presentation. To allow decoders to configure themselves, the DSI packets are inserted intermittently in the video stream.
- *Quarter Pixel mode*—This is a binary parameter used by MPEG-4 and H.264 encoders. The MPEG-2 encoders use half pixel mode. If the parameter is turned on, the encoder computes motion vectors with quarter pixel approximation using floating-point ranges. This obviously takes longer to encode, but does produce better predictions and, consequently, better compression.

An encoder in action is shown in Figure 8-27. The top figure shows parameter settings and input details on a sample high-motion YUV file. The bottom figure shows a preview of the encoding process. The input frame and the encoded/decoded output frame are shown. Previews are useful to visualize output data quality for the parameter settings used and to make the best choice given the bit rate and quality constraints under which encoding has to be performed. Once decided, the encoding may proceed as a batch process.

7 EXERCISES

1. **[R02]** Motion compensation is the main process used to exploit temporal redundancy in any video compression.
   - What is motion compensation and why is it used in video compression?
   - Explain the difference between open loop and closed loop motion compensation. Which one is preferred and why?
   - In video encoding, what takes more time—encoding or decoding?

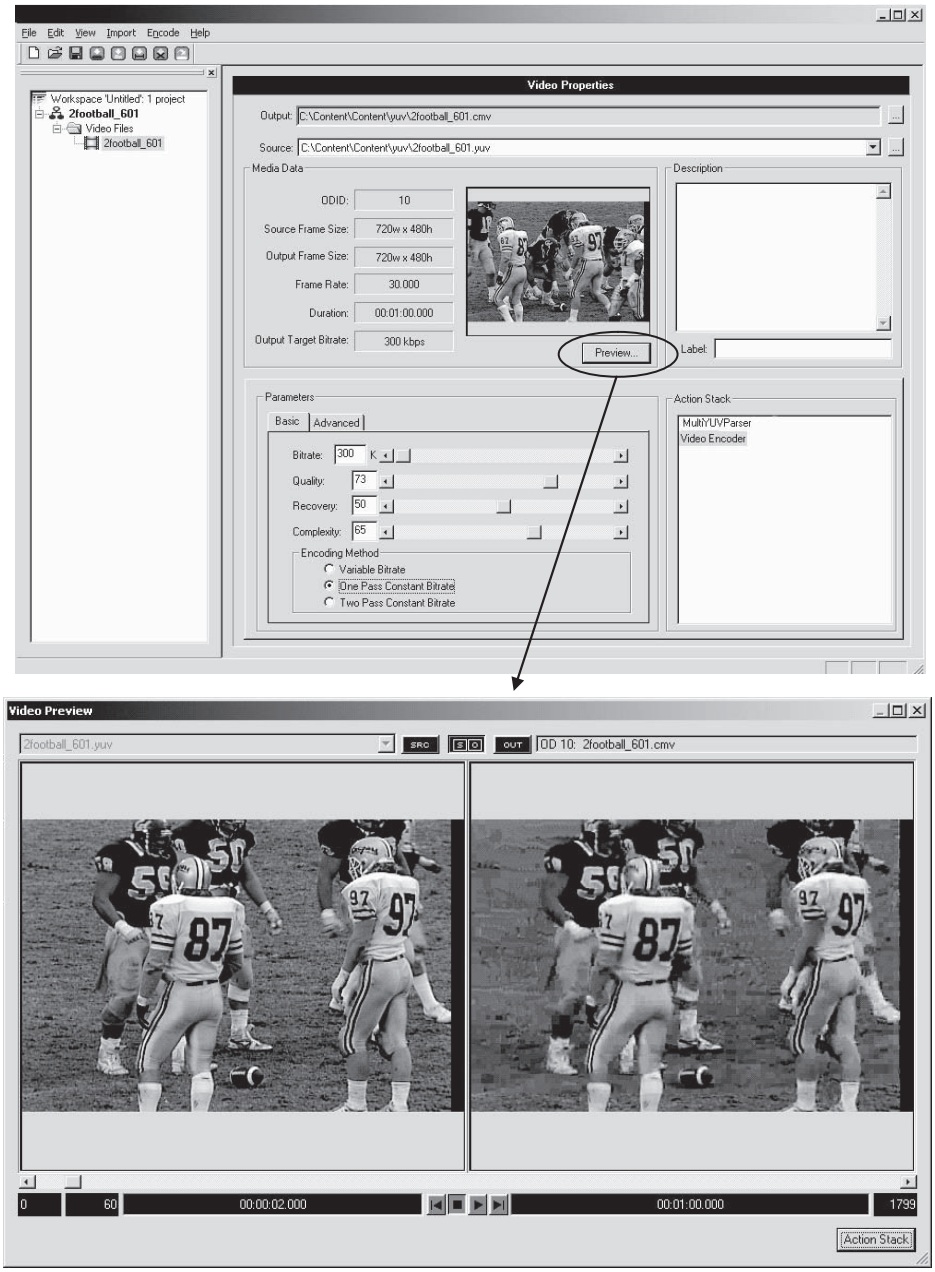


Figure 8-27 An encoding process. The top GUI shows parameters and controls. The bottom image shows an encoder frame in process with the original and the decoded output quality.

2. [**R03**] Motion compensation in video encoding attempts to achieve compression by predicting the successive frames (P frames).
   - Will this always achieve compression? If yes, explain why; if not, give a concrete counterexample.
   - Prior to motion vector computation, each frame is converted to the YCrCb format. Does motion vector computation occur on all three channels for a macroblock?
   - What if we maintained the frames in the RGB color space, which channel(s) would you use to compute your motion vector?
3. [**R04**] Variable bit rates and constant bit rates are used for creating compressed video.
   - When it comes to encoding video for distribution in the form of DVDs, which one produces better quality and why?
   - Why are variable bit rates generated during compression of video streams? Give at least two reasons.
   - For streaming purposes, you would ideally like to use CBR. You will definitely use I and P frames, but you have a choice of whether to use B frames. Explain giving qualitative reasons why/when you would choose B frames and why/when you would not.
4. [**R06**] The MPEG video stream consists of I, P, and B frames. Although the compression is great with P and B frames to meet the requirements of commercial products, one feature that is very impractical or difficult to implement is trick play—such as fast forward or fast rewind (1X, 2X, 4X, and so on).
   - Can you suggest what makes this difficult?
   - In the text, we mentioned the use of GOPs to achieve this. How do GOPs work and what must the decoder know to fast forward or fast rewind?
   - Suppose that your bit stream cannot be structured using GOPs, although you could still use I, P, and B frames. Suggest a way to achieve this functionality that is efficient and yet maintains good compression.
5. [**R04**] Video compression in all standards uses the block-based motion compensation algorithm. The size of macroblocks is normally fixed to be 16×16.
   - Why is this choice of 16×16 used in standards?
   - Under which circumstances would you want to decrease this size? Why?
   - Can you think of any applications where you would want to increase it?
   - Let us say you could design your own standard where you can dynamically increase or decrease this size, but it is fixed for each frame. For example, encoding frame n can be done with 16×16 macroblocks, whereas encoding frame m can be done with 4×4 macroblocks or 32×32 macroblocks. What kind of algorithm would you implement to control the choice of size on a frame-by-frame basis?
   - Can you think of a suitable application where employing the adaptive macroblock size technique proves useful?
   - H.263 allows you to divide a 16×16 macro block into four 8×8 blocks for motion compensation. When is this feature useful?

6. [**R03**] B frames tend to reduce the bit rate and attain superior compression—and as a result, they are used by many standards.
   - However, H.261 does not use B frames. Only P and I frames are used. H.261 refrained from using B frames or even making amendments to include its usage. Can you suggest what the reason for this is?
   - In MPEG-style encoding, encoders attempt to quantize or compress B frames more effectively than the other frame types. Mention two reasons why this should be the case.
7. [**R03**] Suppose we encode a sequence of 20 frames using MPEG. Each frame is encoded in I (intraframe), P (forward prediction), or B (bidirectional prediction) mode according to the following order:

   I P P B B P I B I P P B B B B P I P I P

   - What is the correct transmission and decoding order?
   - B frames induce a delay at the encoder. What is the minimum delay (in terms of frame period) due to bidirectional encoding in this case?
8. [**R03**] When using search techniques to compute motion vectors, the text has discussed a lot of methods, from full search, to logarithmic search.
   - If compressing video content to put onto a DVD for distribution, which motion vector search technique would you use and why?
   - Which motion vector search technique would you use when compressing video of a live game intended for real-time viewing on a distributed client system?
   - When using hierarchical search with three levels and with a search parameter of $k = 16$, what is the maximum value (or magnitude) of the motion vector at level 3?
   - If the motion vector value at level 3 in the preceding question was (2,-1), how many possible values can there be for the actual motion vector on level 1? What are these values?
9. [**R05**] A video has a resolution of 352×288 at 30 frames per second and you are compressing it using I and P frames. Each motion vector computation takes an average of 3 milliseconds. Between any two I frames there can be at most 20 other frames.
   - What is the maximum time (in seconds) that could be spent in motion vector computation for compressing 2 seconds of video?
   - Can this be used for real-time encoding and distribution setup?
   - Assume that the encoder is a software/hardware program whose algorithm working cannot be modified. What tricks can you use to make the encoder go faster?
10. [**R06**] Although motion compensation is used to compress video P and B frames, it is also an expensive process and the whole computation is useful when there is a good amount of redundancy from frame to frame. However, in certain instances like scene changes and cuts, performing motion compensation is wasteful and often results in a high-entropy error image.

In this case, it would be appropriate to insert an I frame. You need a way to quickly detect a scene change.

- One way to achieve this is to do a frame-by-frame difference with the previous frame and see the result sum of all difference pixels. Computationally, this is fast and if there is temporal redundancy, the sum value is low. However, if nothing is in common, the error difference is high. A high error would mean that you encode the current frame as an I frame, else use a P frame. When would this work and what problems do you see with this process?
- How can you *quickly* and *reliably* detect scene changes so that the encoder does not waste time by blindly motion compensating the current frame and encode an I frame instead?
- What is the complexity of your algorithm?
- Can you suggest how (if at all) your algorithm can be modified to detect fades that frequently happen at the beginning or end of a movie or even dissolves that happen as one scene transitions into another?

11. [**R04**] In this question, and the next, you will attempt to understand how a decoder's clock cycle works. An incoming group of 20 pictures is encoded by an MPEG family, as shown in the following line:

$$I_1\ P_2\ P_3\ P_4\ I_5\ P_6\ P_7\ P_8\ I_9\ P_{10}\ P_{11}\ P_{12}\ I_{13}\ P_{14}\ P_{15}\ P_{16}\ I_{17}\ P_{18}\ P_{19}\ P_{20}$$

The encoded sequence is sent to the decoder. The decoder works by having two logical engines running in parallel in different threads. One is called the decoding engine, which does the decoding and the other is called the presentation engine, which is responsible for taking the decoded frame and displaying it. The table in Figure 8-28 displays a state diagram at the decoder explaining which frames are received, which ones are being decoded, and which ones are being presented or rendered. There is always a playout buffer, which in this case has been assumed to be of size 10. The decoder thread has two columns; the first column gives the frame being decoded and the second column gives any reference frame that needs to be cached. The presentation thread shows the frame being displayed (after being decoded). Note that it takes 10 clock cycles to fill the playout buffer and, hence, decoding starts at time 11 and, consequently, the first frame is available for display at time 12.

- The beginning 12 cycles are shown; fill in the rest of the table. The state at time 23 is also shown for reference.
- At what time does the decoder thread start and when does it finish?
- At what time does the presentation thread start and when does it finish?

12. [**R08**] This is an extension of Exercise 11. Suppose the preceding GOP is encoded using B frames to form the encoded sequence:

$$I_1\ B_2\ B_3\ P_4\ B_5\ B_6\ B_7\ B_8\ I_9\ B_{10}\ B_{11}\ P_{12}\ B_{13}\ B_{14}\ I_{15}\ B_{16}\ B_{17}\ P_{18}\ B_{19}\ P_{20}$$

In the absence of B frames, the sequential processing was derived as part of the preceding question. However, in the presence of B frames, this processing

Time cycle	Received frames in buffer	Decoder thread		Presentation thread
1	I_1			
2	$P_2\ I_1$			
3	$P_3\ P_2\ I_1$			
4	$P_4\ P_3\ P_2\ I_1$			
5	$I_5\ P_4\ P_3\ P_2\ I_1$			
6	$P_6\ I_5\ P_4\ P_3\ P_2\ I_1$			
7	$P_7\ P_6\ I_5\ P_4\ P_3\ P_2\ I_1$			
8	$P_8\ P_7\ P_6\ I_5\ P_4\ P_3\ P_2\ I_1$			
9	$P_9\ P_8\ P_7\ P_6\ I_5\ P_4\ P_3\ P_2\ I_1$			
10	$P_{10}\ I_9\ P_8\ P_7\ P_6\ I_5\ P_4\ P_3\ P_2\ I_1$			
11	$P_{11}\ P_{10}\ I_9\ P_8\ P_7\ P_6\ I_5\ P_4\ P_3\ P_2$	I_1		
12	$P_{12}\ P_{11}\ P_{10}\ I_9\ P_8\ P_7\ P_6\ I_5\ P_4\ P_3$	P_2	I_1	I_1
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
23	— — — $P_{20}\ P_{19}\ P_{18}\ I_{17}\ P_{16}\ P_{15}\ P_{14}$	I_{13}	—	P_{12}
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

Figure 8-28

needs to change because the stream has to be reordered for transmission introducing a delay.

- What does the decoded stream look like and how much delay does the encoder need to undergo before sending data to the decoder?

The aim of this next set of questions is to reanalyze what happens in the decoder in the presence of these B frames by constructing a table similar to the one in Exercise 11. Assume that we have a playout buffer of size 10 and decoding begins at time 11. However, note that here the decoding of a frame may require up to two reference frames in memory.

- Complete the table in Figure 8-29 to understand which frames are decoded at each clock cycle, which frames are used in the decoder's reference frame stack of two frames, and at what times frames are rendered? The table is started for you with the state at time 23 shown for reference.
- At what time does the decoder thread start and when does it finish?
- At what time does the presentation thread start and when does it finish?

Time cycle	Received frames in buffer	Decoder thread	Presentation thread
1	I_1		
2	$P_4\ I_1$		
3	$B_2\ P_4\ I_1$		
4	$B_3\ B_2\ P_4\ I_1$		
5	$I_9\ B_3\ B_2\ P_4\ I_1$		
6	$B_5\ I_9\ B_3\ B_2\ P_4\ I_1$		
7	$B_6\ B_5\ I_9\ B_3\ B_2\ P_4\ I_1$		
8	$B_7\ B_6\ B_5\ I_9\ B_3\ B_2\ P_4\ I_1$		
9	$B_8\ B_7\ B_6\ B_5\ I_9\ B_3\ B_2\ P_4\ I_1$		
10	$P_{12}\ B_8\ B_7\ B_6\ B_5\ I_9\ B_3\ B_2\ P_4\ I_1$		
11	$B_{10}\ P_{12}\ B_8\ B_7\ B_6\ B_5\ I_9\ B_3\ B_2\ P_4$	I_1 — —	
12	$B_{11}\ B_{10}\ P_{12}\ B_8\ B_7\ B_6\ B_5\ I_9\ B_3\ B_2$	P_4 I_1—	
13	$I_{15}\ B_{11}\ B_{10}\ P_{12}\ B_8\ B_7\ B_6\ B_5\ I_9\ B_3$	B_2 $P_4\ I_1$	I_1
.	.	. .	.
.	.	. .	.
.	.	. .	.
23	— — — $B_{19}\ P_{20}\ B_{17}\ B_{16}\ P_{18}\ B_{14}\ B_{13}$	I_{15} $P_{12}\ I_9$	B_{11}
.	.	. .	.
.	.	. .	.
.	.	. .	.

Figure 8-29

13. [**R07**] Suppose we have a CIF (352 × 288 frame size) format digital video at 30 fps which plays for 10 minutes. Assume each pixel requires 12 bits in the YUV 4:2:0 representation. We need to deliver it over a DSL network supporting an average of 500 Kbps at CBR. To achieve CBR better, you organize your frames into groups (GOPs) of 45 frames each.
    - On average, how many bits are required for each GOP?
    - Each GOP is headed by an I frame and followed by P and B frames. There are at most two B frames between P and I frames. Assuming I frames compress to 20% and P frames to 1.2%, what should the minimal B frame compression ratio be?
14. [**R08**] Normally, when encoding I frames in video, or when encoding the error images in predicted frames, the standard JPEG pipeline is followed where the frame is divided into 8 × 8 blocks and each block is DCT transformed followed by zigzag ordering of the AC coefficients. This works best when the

video is progressive. However, in the case of interlaced video, each field is digitized separately.

- Can you combine two fields into one frame and then compress the frame as an I frame? What kinds of problems do you see happening here?
- If you maintained the frames as two fields and encoded each field with 8×8 block-based DCT transforms, what kind of problems could you potentially see with use of zigzag ordering?
- Under what circumstances (fast motion or slow motion) will using fields as themselves work and yield less entropy in the zigzag ordering?
- Can you think of a better way of ordering that could possibly give lesser entropy and, hence, better encoding?

15. [**R05**] Two parameters specified during encoding are the *bit rate* and *quantization level*. The bit rate specifies the target rate at which encoding has to occur, whereas the quantization level controls the amount of quantization allowed in frames.
    - Which one (mention both if so) should an encoder allow to be varied for CBR, and which one should be allowed for VBR?
    - Rate control during CBR is a difficult problem, especially for real time. However, what strategy can you adopt for passive encoding, where the encoder does have to encode or stream in real time?
16. [**R06**] Consumer-level video encoders capture videos up to 640×480 at 30 fps, compress them in real time, and store the video to onboard memories. These memory devices are not inexpensive and can very quickly get filled in a few minutes. Higher compression ratios are desirable here. When there is high motion in the scene, more bits are needed.
    - Apart from the normal motion-related issues in the content being captured, can you think of other problems that would unnecessarily increase the compressed size?
    - What preprocessing steps can you perform with the video after capture but prior to encoding to get rid of these issues?

PROGRAMMING ASSIGNMENTS

17. [**R07**] In this programming assignment, you will implement motion compensation and study the effect this has on errors generated in prediction. Sample video files with format information are provided in the Student Downloads section of www.cengage.com. Also provided is sample code to read and display an image. Modify this code to read and display video frames. Assume that Frame 1 will always be an I frame and the following frames will be P frames. This assumption should suffice for the short video sequences that you are working with, each having at most 100 frames.
    - For the first part, assume that you want to predict P frames as whole frames rather than blocks. Each frame is predicted as a whole using the

previous frame. Correspondingly, implement a subroutine that takes two frames as input and computes the differences between them. When you call this subroutine on two successive frames, it will return an error frame. There is no motion vector to compute here. Display all the error frames. Note that the information content of the error should be less than that of the frame itself.

- The next step is to implement block-based motion prediction with motion vectors for each block. Each block will be the standard MPEG size 16×16. Write a subroutine that will take two frames, one reference frame that will be used to search into and a target frame that you will predict. Break the target frame into macroblocks of 16×16. If the image frame width and height is not a multiple of 16, pad the frame with black pixels accordingly. For each block in the target frame, go to the corresponding colocation in the reference frame and find the best-matching area, as explained in the text of this chapter. Use the SAD evaluation with a search area defined by $k = 16$, so your motion vectors are at most size 16 pixels in both dimensions. Using the predicted block, compute the error block as the difference between the original block and the predicted reference block. When you do this for all the blocks, you will get an error frame. Display all the error frames. You should find that your error frames are substantially lower compared with the previous case—though they take much longer to compute.

18. **[R08]** In this programming assignment, you will see how block-based motion vector prediction can be used in tasks other than compression. An interesting application of using blocks is to remove objects or people in a video. For example, let us say you have a video sequence where the camera has not moved and the background is relatively static but a few foreground objects are moving. Your task is to approximate the moving object by blocks and replace those blocks by the background as though the object was never there. The general-purpose solution is much harder, but here are some ideas to get you to do an interesting assignment in this area. When implementing this, make sure that you can play around with the block size as a parameter to evaluate how well your object removal algorithm works with different sized macroblocks.

    - First, read in a set of video frames. Assume that the first frame has only the background with no objects in motion. Given a frame *n* in the sequence, break it into blocks. Find a motion vector for each block from the previous reference frame. For all the blocks that correspond to the background, you should have a zero (or close to zero) motion vector because your camera was static, and for blocks of an object in motion, you should have a nonzero motion vector. Keep track of all these motion vectors; you could even draw them in each block when you render the video.
    - Next, find the nonzero motion vector blocks. Replace the block by the corresponding content of the background. These pixels should be taken from a previous frame where there was no object in motion. Replacing all

such blocks should remove moving objects by replacing them with the background. Repeat the task using different sized macroblocks.

- You might find artifacts and discontinuities at the boundaries of the blocks that you are replacing. How will you minimize such artifacts?

This specific solution will work only when you assume that the camera is not moving, the objects that move exhibit rigid motion, and there are no parallax changes. In general video, the camera is bound to move, objects exhibit non-rigid motion, and, in addition, there will be lighting changes as the camera moves.

- But let us say you could reliably detect all macroblocks on each frame that correspond to the background. How can you make use of this to get better compression in addition to using macroblock-based local motion compensation?

This page intentionally left blank

CHAPTER 9

Media Compression: Audio

Audio was the first digitally distributed media type, in the form of compact discs (CDs), an innovation welcomed by the industry and consumers alike. Although CD audio was uncompressed, its fidelity and quality ensured the obliteration of analog audio. The success of digital audio CDs resulted in the distribution of other digital media types on compact discs, such as multimedia CD-ROMs, and DVDs. The digital audio industry is also now transitioning to use more sophisticated distributions, such as connected and wireless networks with inexpensive and light memory devices. Such distributions face a series of hindrances that include reduced network bandwidth and limited storage capacities. For example, streaming digital high-fidelity audio via the Internet at higher bandwidths is possible today, but the compression ratios still need improvements to stream over cell phone networks. Today, a variety of applications, including Apple's iPod and online music sites, store stereo-quality audio in the popular MP3 format. XM Radio provides subscription-based access to high-band digital audio broadcast. It uses a dedicated satellite distribution network needing special receivers. VoIP and vocoders have made communication across the world convenient and inexpensive. The next generation of such applications will need to stream audio to mobile devices, such as CD players in cars, cell phones, and in-house entertainment surround sound receivers. These current and new applications have created a demand for high-quality digital audio at different ranges of bit rates. The result is a variety of industry standards, such as the MPEG MP3, AAC, Dolby AC-3, and DTS.

This chapter deals with digital audio-compression techniques to obtain compact digital representations for narrow-band (speech) and wide-band (music) audio, including both research and standards into which these techniques have been adopted. Audio formats were discussed in Chapter 2. Section 1 discusses the

need to perform audio compression and gives bandwidth requirements for a variety of commonly used audio signals. Next, Section 2 explains how and why audio-compression techniques fundamentally differ from the visual domain compression techniques that were discussed in the preceding chapter. The different generic methods of audio-encoding techniques are then explained in Sections 3–6. Some of these techniques make use of psychoacoustics, which explains how the human auditory system senses sound, and they capitalize on its drawbacks. Finally, Section 7 enumerates the working and salient features of popularly used formats from the MPEG and ITU standards committees such as MP3, G.711, G.722, Advanced Audio Coding (AAC), code excited linear prediction (CELP), and Dolby AC-3.

1 THE NEED FOR AUDIO COMPRESSION

Figure 9-1 lists the storage and transmission requirements for various kinds of audio signals. The final two rows describe the amount of time needed to transmit one second of data. As you can see, the amount of data needed to be delivered to the end client per second does not suffice the real-time needs at commonly available modem rates and even at broadband rates for some signals. For example, uncompressed speech communication over a 56 Kb channel still takes more than a second of transmission time, not withstanding any network latency. Real-time

Audio data	Speech (mono)	CD (stereo)	FM radio (stereo)	5.1 surround sound
Sample size	8 bits	16 bits	16 bits	16 bits
Sample rate	8 KHz	44.1 KHz	22 KHz	44.1 KHz
One second of uncompressed size (B for bytes)	8 KB	88.2 KB (stereo)	44 KB (stereo)	530 KB (6 channels)
One second of transmission bandwidth (b for bits)	64 Kbps	1.4 Mbps (stereo)	704 Kbps (stereo)	4.23 Mbps (6 channels)
Transmission times for one second of data (56 Kb modem)	1.14 seconds	25 seconds	12.5 seconds	76 seconds
Transmission times for one second of data (780 Kb DSL)	0.08 seconds	1.8 seconds	0.9 seconds	5.4 seconds

Figure 9-1 Examples showing storage space, transmission bandwidth, and transmission time required for uncompressed audio formats

requirements for conversational telephony have shown that the overall delay acceptable is around 200 milliseconds.

Speech is low band and has a lower sampling rate requirement compared with high-fidelity sound such as in music, where the sampling rate is 44.1 KHz. The CD standard stores almost 70 minutes of uncompressed digital music on one CD ($= 70 \times 60 \times 1.4 = 5880$ Megabits or 735 Megabytes). Transmitting this uncompressed content for streaming purposes is virtually impossible on the current consumer bandwidths. Various compression techniques and standards have been designed to help make this possible.

2 AUDIO-COMPRESSION THEORY

It is natural to compare audio-compression techniques with the previously discussed visual media compression techniques for video and images. Digital audio is represented as a one-dimensional set of PCM samples, whereas images have spatial 2D representations, and video has 3D spatiotemporal representations. Because of the dimensional simplicity of audio signals, you might expect audio-compression techniques to be less complex when compared with the visual domain counterparts of block-based and motion-predicted compression techniques. This is not true, however, and the compression ratios obtained for visual signals tend to be much larger than the compression ratios obtained for audio signals. First, the visual domain signals carry larger amounts of information and, correspondingly, more signal redundancy compared with audio signals. But more important, the working of the human auditory system (HAS) in the ear is very different from the human visual system (HVS). HAS is more sensitive to noise and quality degradation when compared with HVS and, therefore, audio-compression techniques need to be careful with any loss and distortion that might be imposed during compression.

Digital audio is represented in the form of channels. The number of channels describes whether the audio signal is mono, stereo, or even surround sound. Each channel consists of a sequence of samples and can be described by a sampling rate and quantization bits. When this digital representation is compressed, a number of trade-offs need to be addressed. In general, the perceived quality after compression and decompression needs to be maximized, while at the same time the amount of information needed to represent the audio signal needs to be minimized. These two conflicting goals have to be addressed by *any* audio-compression scheme—taking into consideration the following properties:

- *Fidelity*—Fidelity describes how perceptually different the output of the codec is when compared with the original signal. This is the most important metric of an audio codec and describes the overall audio quality. The expected fidelity might vary with applications. For example, ''telephone quality'' might be considered acceptable in applications where the intelligibility of spoken words is important. But higher levels of fidelity are expected for distribution of

music. Higher fidelity normally would necessitate higher data rates and greater codec complexity.

- *Data rate or bit rate*—The data rate, which is linked to the throughput and storage, is an important factor for compression. Obviously, higher data rates have better audio quality but imply higher costs in transmission and storage.
- *Complexity*—The complexity of an audio codec translates into hardware and software costs in the encoder and decoder. The complexity trade-offs depend on the application. For example, in a non-real-time audio broadcast situation where the encoding is done once, the encoding complexity can be high—requiring more time but producing good fidelity at a lower bit rate. Low-cost decoders are preferable compared with the encoder. Conversely, in a real-time videoconferencing application, both the encoder and decoder need scaled-down complexities to address real-time issues.

In general, any audio codec's goal is to provide high fidelity, with lower data rates while sustaining as low a complexity as possible.

Almost all video-compression techniques and standards are based on specific adaptations of the generic motion compensation algorithms, as was explained in the preceding chapter. Audio-compression techniques are more varied and can be categorized into different groups depending on how the audio signals are understood. The earlier standards established for digital voice communications and CD music do not make use of any advanced audio theory, and only assume that the signal is a simple digital waveform. Using this assumption, the compression techniques aim to reduce only the statistical redundancy. Other techniques borrow more sophisticated theories from psychoacoustics and remove redundancy in digital audio depending on what the HAS can or cannot perceive. Psychoacoustics is a rich area of research and is continuing to make a critical impact on the quality of compressed digital audio. Another class of techniques focuses purely on low-band speech, and aims to parameterize human voice models. Finally, if the audio signal is viewed as composed of a list of events, as in an orchestra, the signal can be compressed by semantically describing these events and their interaction. This is the area of structured audio. All these different groups of compression techniques are explained in the next few sections.

Common to all techniques is the understanding of the way sound amplitudes are measured, the use of these measurements to validate the perception of frequencies, and the control of thresholds in the compression process. Sound amplitude is measured in decibels, which is a relative (not absolute) measuring system. The decibel scale is a simple way of comparing two sound intensity levels (*watt per square centimeter*), and is defined as 10 log (I/I_o), where I is the intensity of the measured sound signal, and I_o is an arbitrary reference intensity. For audio purposes, I_o is taken to be the intensity of sound that humans can barely hear, and has been experimentally measured to be 10^{-16} *watts/sq. cm*. Any I with intensity lower than I_o will, therefore, have a negative decibel value and vice versa. If an audio signal has an intensity of 10^{-9} watts/sq cm, its decibel level is 70 dB. Because of the relative scale, an increase in 1 dB implies that the sound intensity has gone up tenfold. Figure 9-2 shows some common decibel levels.

Type of sound	Decibel level
Threshold of hearing (just silent)	0 dB
Average whisper	20 dB
Conversation	40 dB
Traffic on a busy street	70–90 dB
Threshold of pain (going deaf)	120–130 dB
Jet aircraft at takeoff	140 dB

Figure 9-2 Audio decibel measurements for some commonly occurring sounds

As we shall see later, the sound dB levels are very generically defined here, and the perception of sound by human ears depends not only on the intensity level, but also on the frequency. The next sections discuss generic audio compression techniques according to whether we view the signal as a waveform (Section 3), as a signal that is ultimately perceived by the human ear and can, thus, be subjected to psychoacoustic principles (Section 4), as a signal produced by a sound source (Section 5), or as a series of specific events as in a musical composition (Section 6).

3 AUDIO AS A WAVEFORM

This class of algorithms makes no special assumption about the audio signal, and performs all the analysis in the time domain. Compression here can only be achieved by different quantization strategies, followed by the removal of statistical redundancies using entropy coding. Although the techniques described in the following sections have been used in earlier audio standards for digital voice communication, they fail to provide good compression ratios when compared with recent standards that make use of psychoacoustics. Consequently, because of their inefficient compression ratios, they are normally used in conjunction with other audio compression schemes. Waveform compression methods in this category are grouped as follows.

3.1 DPCM and Entropy Coding

Differential pulse code modulation (DPCM) is the most simplistic form of compression, and was introduced in Chapter 6. It exploits the fact that the range of differences in the amplitude between successive audio samples is less than the range of the actual sample amplitudes. As a result, coding the differences between the successive signal samples reduces the entropy (variation in amplitudes) when compared with the original signal samples. The smaller differences $d(n)$ can then be coded with fewer bits. However, reducing the number of bits introduces quantization errors. Compression can be achieved if these errors are tolerable for human perception. Refer to Figures 6-10, 6-11, and 6-12 in Chapter 6 for a more detailed analysis of DPCM techniques, which include open loop and closed loop.

DPCM techniques do achieve modest compression. For example, pulse code modulated speech is obtained by sampling the analog audio signal at 8 KHz with each sample quantized to 8 bits, resulting in a bit rate of 64 Kbps. If 7 bits are used to represent the differences between successive samples, the resulting bit rate goes down 56 Kbps with an acceptable quality.

An important aspect of DPCM is to understand that it is a prediction-based method, where the next sample is predicted as the current sample, and the difference between the predicted and actual sample is coded. The closer the predicted value is to the actual one, the smaller the range of variation in the difference errors. Predicting the next sample value based only on one previous sample may be simple but not as effective as involving a number of immediately preceding samples in the prediction process. In such cases, the preceding samples are weighted by coefficients that suggest how much a preceding sample influences the current sample value. These are called predictor coefficients. Figure 9-3 illustrates that the accuracy of the predicted value of a sample increases as more samples are used in the prediction process.

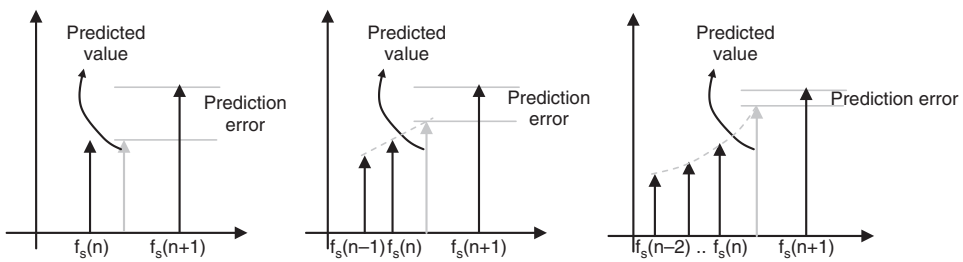


Figure 9-3 Prediction of sample values in DPCM techniques. The next sample to be coded is predicted based only on preceding sample (left), as a linear combination of two preceding samples (middle), and as a weighted combination of three previous samples (right). The prediction error shown as the difference between the horizontal lines in each case successively decreases as more samples are used.

3.2 Delta Modulation

Delta modulation is similar to DPCM, except that the differences $d(n)$ are coded with one bit—0 or 1. Given a current state, a 0 signifies that the next sample has increased by a constant delta amount, whereas a 1 indicates that the next sample has decreased by a constant delta amount. The delta amount is fixed and can be decided depending on the input signal's sampling rate. Every step introduces an error, which is low for slower-changing signals but can get larger if the signal starts to change rapidly.

The obvious advantage here is the controlled bit rate, but that comes at the cost of more error compared with other similar techniques. Delta modulation has evolved into a simple but efficient method of digitizing voice signals for communications and voice I/O data processing. Digital voice signals have lower frequency content (up to 8 KHz). Using delta modulation for voice signals consequently results in small errors acceptable for certain voice applications.

3.3 ADPCM

Adaptive differential pulse code modulation (ADPCM) is similar to DPCM, except for the fact that it adaptively modifies the number of bits used to quantize the differences. A block diagram illustrating the working of ADPCM is shown in Figure 9-4.

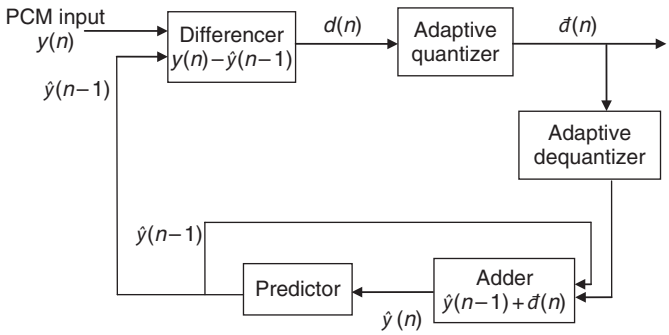


Figure 9-4 Adaptive DPCM. The block diagram is similar to DPCM with an adaptive quantizer inserted after the differencer. This block adaptively decides how many quantization bits to use to quantize d(n) to đ(n).

ADPCM bit streams have an additional complexity imposed by the need to insert control bits whenever the quantization bits change. This is essential to let the decoder know how to parse the incoming bits. Frequent insertion of control bits can easily increase the entropy and the bit rate. Therefore, most ADPCM encoders use two modes for adaptive quantization—one for low frequency and the other for high frequency. When the frequency content of the input signal is low, $d(n)$ is smaller and, therefore, fewer bits can be used to quantize the value, and vice versa.

3.4 Logarithmic Quantization Scales—A-law and μ law

In the discussion so far, the dynamic range of the digitized audio signal is uniformly quantized. However, some encoding methods such as A-law and μ-law (pronounced mu-law) follow a logarithm quantization scale, also known as companding. It is based on the statistical observation that intensities of many audio signals, such as speech, are more likely to be concentrated at lower intensity levels, rather than at higher intensity levels in the dynamic audio range. Because the distribution of PCM values in such signals is not uniform over the entire dynamic range, quantization errors should also be distributed nonuniformly. That is, values that are more probable should produce lower errors than values that are less probable. This can be achieved by adjusting quantization intervals nonuniformly. The technique is to vary the distance between quantization reconstruction levels so that the distance increases as the amplitude of sample increases. Quantizations corresponding to lower amplitudes have smaller error than quantizations corresponding to higher amplitudes. Figure 9-5 illustrates this, where the original signal on the left is shown digitized using eight uniform quantization intervals (center) and eight logarithmic quantization intervals

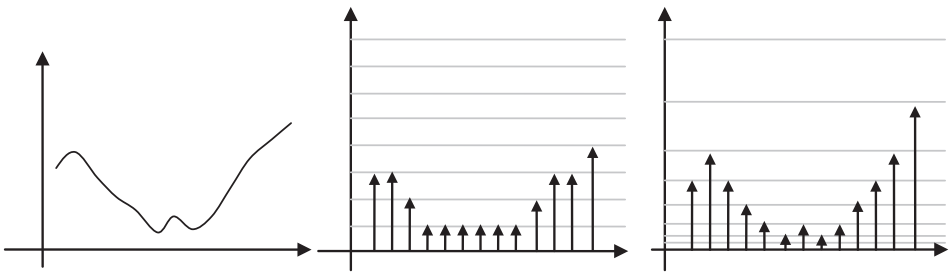


Figure 9-5 Nonlinear quantization scales. The left image shows the original analog signal. The corresponding digitized signals are shown in the center and right. The center signal is obtained by uniform quantization intervals, whereas the far-right signal is obtained by a logarithmically quantized interval scale. The digitized signal on the right better approximates the original signal.

(right). The digitized signal to the right preserves the original signal characteristics better than the digitized signal in the center.

Nonuniform quantization intervals are used in human speech compression standards. Audio speech signals tend to have a large dynamic range, up to 60 decibels, thus requiring a large number of quantization levels, typically 4096 levels, which require 13 bits. The human ear, however, is more sensitive to lower amplitude intensities than to larger amplitude values. Better compression results are obtained by minimizing quantization errors at lower amplitudes than at higher amplitudes. To do this, the sampled signal at 13 bits is companded (logarithmically mapped) to 256 nonuniform 8-bit quantization levels. Two particular logarithmic quantization techniques for PCM audio are defined by the ITU (International Telecommunication Union) and are in worldwide use. These are the American μ-law standard, which compands 14 bits to 8 bits, and the European A-law standard, which compands 13 bits to 8 bits.

4 AUDIO COMPRESSION USING PSYCHOACOUSTICS

Compression achieved based on statistical analysis in PCM coding techniques is not sufficient for the data rates required of modern networked applications such as streaming audio. In this section, we discuss a class of compression techniques based on sophisticated analysis of how the human auditory system works. The ear is a physical structure and, thus, has its own limitations and artifacts. The understanding of the perceptual limitations of the ear can be exploited for audio compression by appropriately quantizing and even discarding parts of the signal that are not perceived by the human ear. Psychoacoustics is the branch of psychology that deals with this study of the way humans perceive sound. In the following subsections, we explain the structural and perceptual working of the ear, analyze a few psychoacoustic results, and later, in Section 4.5, we show how these results are used in audio compression.

4.1 Anatomy of the Ear

Figure 9-6 shows the anatomical parts of the ear. Although simplified, it should suffice to fundamentally explain the physical attributes of hearing and, thereby, the limitations. Sound enters the ear and travels through the ear canal. The eardrum receives the sound pulses and conveys the pressure changes via middle ear bones and cochlea to auditory nerve endings. It is important to note that the flow of information via air and fluids is mostly unidirectional. Hence, most of the models in psychoacoustics, whether simple or sophisticated, pose their analysis as a single dimensional effect. The middle ear, consisting of the eardrum and very small bones, is connected to a small membrane attached to the fluid-filled cochlea. The middle ear's main function is to convert the acoustic air pressure vibrations in the ear canal to fluid pressure changes in the cochlea. The inner ear starts with the oval window at one end of the cochlea. The mechanical vibrations from the middle ear excite the fluid in the cochlea via this membrane. The cochlea is a long helically coiled and tapered tube filled with fluid. Internally, the cochlea is lined with hair cells that sense frequency changes propagated through its fluid. The human auditory system behaves very nonlinearly when it comes to perceiving specific frequencies due to the shape of the cochlea, the nonuniform distribution of the sensory cells, and the overall perceptual mechanism.

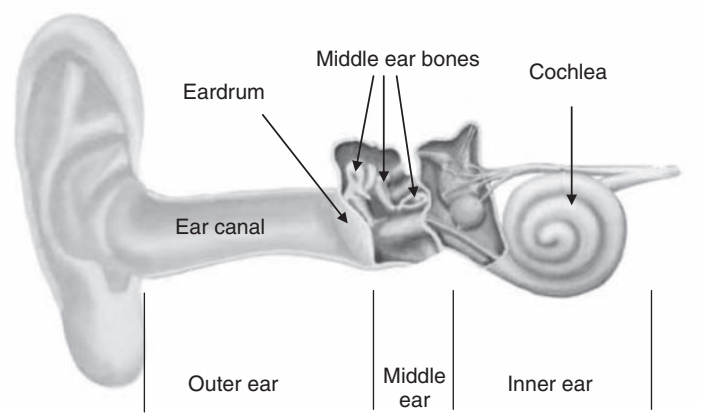


Figure 9-6 Structure of ear. The eardrum in the outer ear communicates sound pressure changes to the middle ear, which consists of three bones, and then to the cochlea in the inner ear. The spiral structure of the cochlea and the sensory nerve distribution within it are mainly responsible for our auditory perception mechanism as well as the limitations.

4.2 Frequency Domain Limits

The human auditory system is capable of perceiving frequencies between 20 Hz and 20 KHz. Higher frequencies (ultrasonic, above 20 KHz) are not perceived by the ear. The dynamic range of hearing, defined as the ratio of the maximum sound amplitude to the quietest sound humans can hear is around 120 decibels. Prior to compression,

appropriate filters can be utilized to ensure that only the "audible frequency content" is input to the encoder.

4.3 Time Domain Limits

A sound signal in the time domain can be decomposed into events that occur at specific times. Two events can be uniquely perceived by the ear depending on how far apart in time they are separated. Normally, events separated by more than 30 milliseconds can be resolved separately. The perception of simultaneous events (less than 30 milliseconds apart) is resolved in the frequency domain, as explained in the next subsection.

4.4 Masking or Hiding

Masking is one of the most important concepts in perceptual hearing. If two sound tones or frequencies are present in the signal, the presence of one might hide the perception of the other, the result being that only one tone is perceived. The audibility level of one frequency is affected by the presence of another neighborhood frequency. To understand how frequencies mask each other, let's consider the curve showing the threshold of human hearing in Figure 9-7. This curve gives the decibel levels for all frequencies at which the sound at that frequency is just audible. To create such a curve, a human subject is kept in a quiet room and a particular frequency tone is generated so that that subject can hear it. It is then reduced to zero and increased until it is just barely audible and the decibel level for that frequency is noted. Repeating this process for all the frequencies generates the curve shown in Figure 9-7.

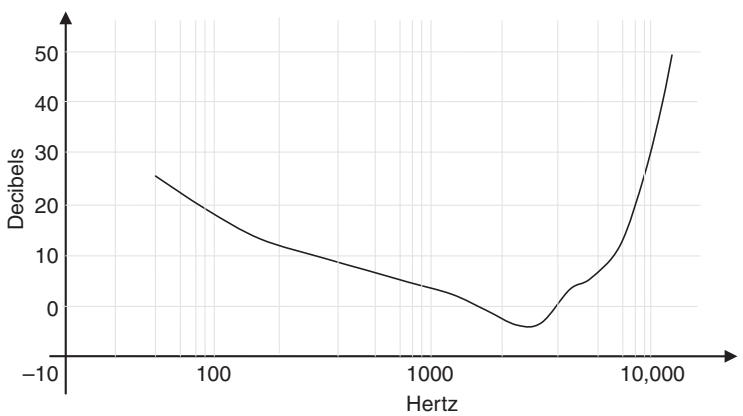


Figure 9-7 Threshold curve for human hearing. The curve shows a plot for the audible level at all frequencies. The ear's response is not the same for all frequencies and changes nonlinearly. For clarity reasons, the graph plot uses a piecewise logarithmic frequency scale on the x-axis.

If the incoming audio signal has a frequency whose energy level is below the threshold curve, it will not be heard. This is important in compression because if a frequency is ultimately not going to be perceived, there is no need to assign any bits to it. However, if the input signal has frequency whose amplitude level is above the threshold curve, it must be assigned bits during compression. In addition, the presence of a perceptible frequency changes the shape of the threshold curve. An example of this is shown in Figure 9-8, where the threshold curve was created just like in Figure 9-7, but all the audible levels were recorded in the presence of a 1 KHz frequency at 45 dB. From the curve, it can be seen that when there is a 1 KHz tone present, other frequency tones less than 45 dB in the neighborhood of 1 KHz are not perceived. A neighborhood tone of 1.1 KHz at 20 dB would normally have been heard well (in quiet), but now will not be heard as a distinct tone in the presence of the stronger 1 KHz tone. The 1 KHz tone is *masking* or *hiding* the 1.1 KHz neighborhood tone for the decibel levels discussed. Here the 1 KHz tone is known as the *masker* and any other tone in the neighborhood, which is masked, is known as the *maskee*. From an encoder's point of view, if a frequency is masked and, consequently, not heard, there is no need to encode it.

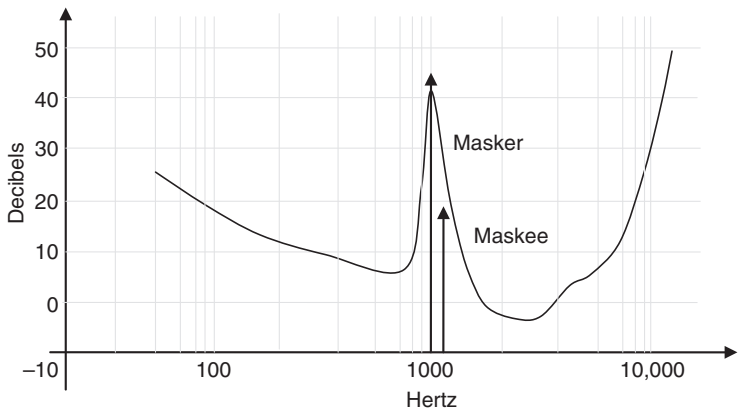


Figure 9-8 Threshold curve in the presence of a 1 KHz frequency at 30 dB. The curve has changed in relation to the threshold curve in the quiet of Figure 9-7. Also shown is a 1.1 KHz tone at 20 dB. Normally, this tone would have been heard, but now gets masked (under the curve) by the stronger 1 KHz tone.

It is important to understand that Figure 9-8 holds only for a single masking tone, specifically for the 1 KHz masking frequency. In the presence of another singular tone, the plot looks very different. In general, the higher the frequency of the masker, the broader the range of influence it has and more neighborhood frequencies it masks, as shown in Figure 9-9. The x-axis is uniform and shown in the 1–20 KHz range. Four increasing frequencies are shown at the same 45 decibel level. The changed threshold curve is shown for each frequency. It can be seen that as the frequency increases, the masking area also increases, which is a perceptual nonuniformity.

Figure 9-9 shows the masking influence of individual frequencies separately. The higher the frequency, the larger the spectral spread that it hides. This oddity is

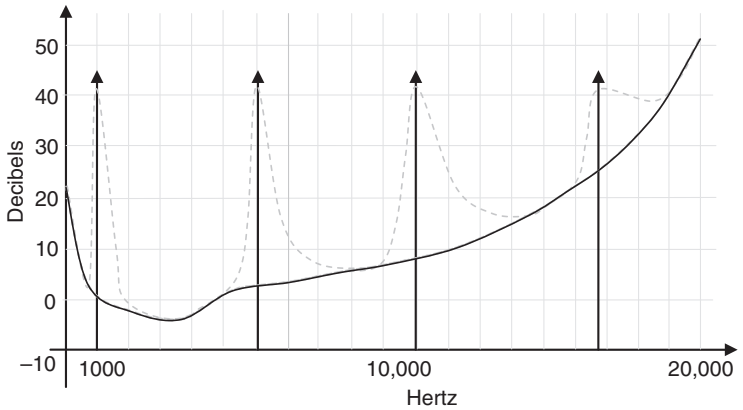


Figure 9-9 Threshold curves for different individual frequencies. Four frequencies are shown, along with the way each one modifies the threshold curve. At the same amplitude level, the higher the frequency, the greater the area it masks.

compounded by the fact that this masking effect is not uniform across the auditory spectrum. The human auditory system is naturally divided into different bands such that the ear cannot perceptually resolve two frequencies in the same band. Experiments indicate that this critical bandwidth remains constant, 100 Hz in width at frequencies below 500 Hz. But this width increases linearly for frequencies above 500 Hz, from 100 Hz to about 6 KHz. Although this is an empirical formulation, it shows that the ear operates like a set of band-pass filters, each allowing a limited range of frequencies through and blocking all others. The presence of multiple masking frequencies in different bands is shown in Figure 9-10. The threshold curve, therefore, changes in a complex and nonuniform manner for different frequencies. Good perceptual models are needed to accurately model the curve with energies distributed unevenly at every frequency.

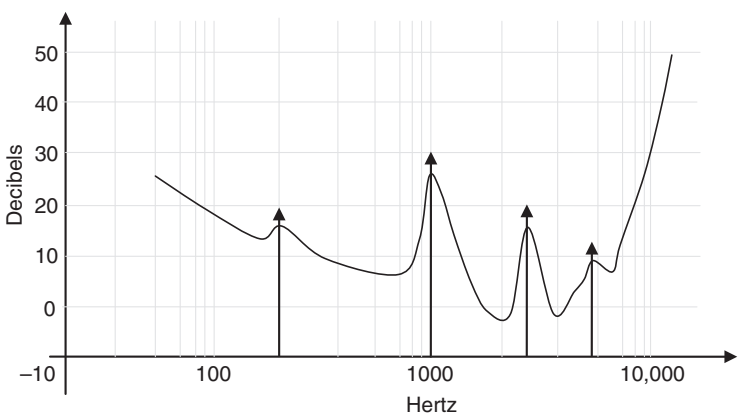


Figure 9-10 Masking effects with multiple maskers. In the presence of multiple maskers, the threshold curve gets raised, masking a lot more frequencies.

The previous few paragraphs mainly explained how frequencies mask each other. Masking effects are also seen in the time domain and are known as temporal masking. They can be understood by extending the experiment discussed previously. When a 1 KHz and 1.1 KHz tone are simultaneously present at 45 dB and 20 dB respectively, the 1 KHz tone masks the 1.1 KHz tone and, hence, the higher tone cannot be heard. If the 45 dB masking tone is removed and only the *maskee* 20 dB tone is present, the ear should perceive it because it is above the threshold curve in quiet. However, the 20 dB tone is not heard instantly but after only after a finite small amount of time. The amount of time is dependent on the intensity difference between the *masker* and the *maskee* and also on how close the *maskee* is to the *masker*. The closer the *maskee* is to the masking tone, the greater the likelihood that the tone cannot be heard after the masking tone is stopped. Also, the larger the masking tone's intensity, the longer it will take for the *maskee* to be heard after the masking tone's removal.

4.5 Perceptual Encoder

All perceptual audio encoders work in the same generic way, where the input signal is transformed to the frequency domain and the frequency coefficients are quantized. A block diagram of this process is shown in Figure 9-11. During encoding, the perceptual distortion that occurs due to quantization has to be minimized. This is very similar in principle to the way visual coding techniques work—allocate bits during quantization of frequency coefficients in such a way that the more perceptually dominant frequencies get more bits. However, there is one major difference. In image and video coding, low frequencies are generally considered dominant. Hence, a constant quantization table (see Chapter 6) suffices during the encoding process. The quantization step in image coding is, thus, reduced to a simple lookup to find out how many bits a frequency coefficient has

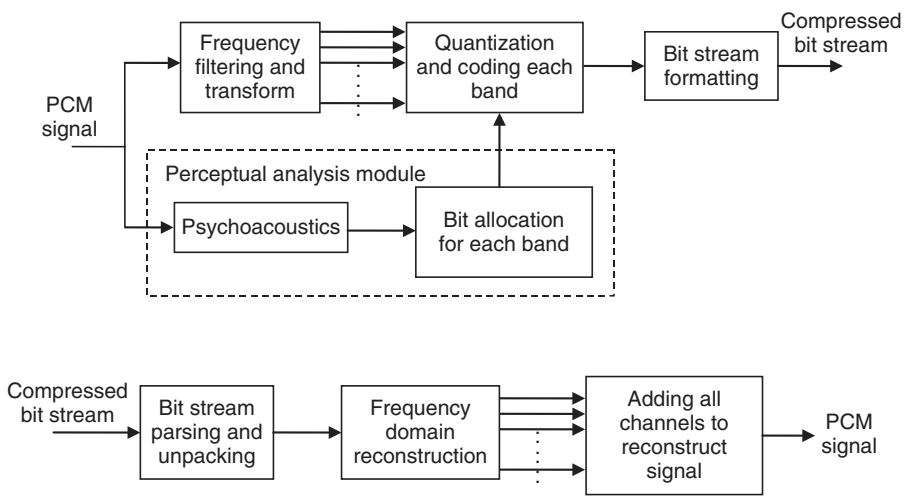


Figure 9-11 Generic perceptual encoder (above) and decoder (below). The encoder is more complex than the decoder, and constantly analyzes incoming frequencies relative to an auditory perceptual model. Frequencies that are masked are assigned no bits or fewer bits.

to be allocated. In audio, there is no universal frequency or frequency range that perceptually dominates. In fact, frequencies that dominate at one instant in time might be masked at another instant of time, depending on what tones are heard. The quantization table used to allocate bits to frequency coefficients, therefore, constantly changes dynamically. This is what makes an audio encoder process more complex because it has to constantly model the audio threshold curve, selecting the important as well as imperceptible frequencies on the fly. The overall process can be explained as follows:

- The input signal is divided into windows of samples. Each window contains samples that are normally multiples of two between 512 and 4096. All the windows can be thought of as independent signals.
- The data window then follows two parallel paths—first, it undergoes subband filtering, where the time domain samples are split into different frequency signals generating frequency coefficients for each band. These frequency coefficients need to be quantized.
- Simultaneously, the input signal goes through a perceptual analysis module, which attempts to model the threshold curve for that data set using the psychoacoustic principles discussed previously. The perceptual modeling process analyzes the importance of frequency bands, and a bit allocation strategy is conveyed to the encoder.
- The encoder then quantizes the frequency coefficients depending on the bit allocation strategy conveyed to it by the perceptual analysis process. Important and definitely perceptible bands are given more bits than the bands that get completely masked. The bands between those are assigned fewer bits. Figure 9-12 illustrates this bit allocation process.

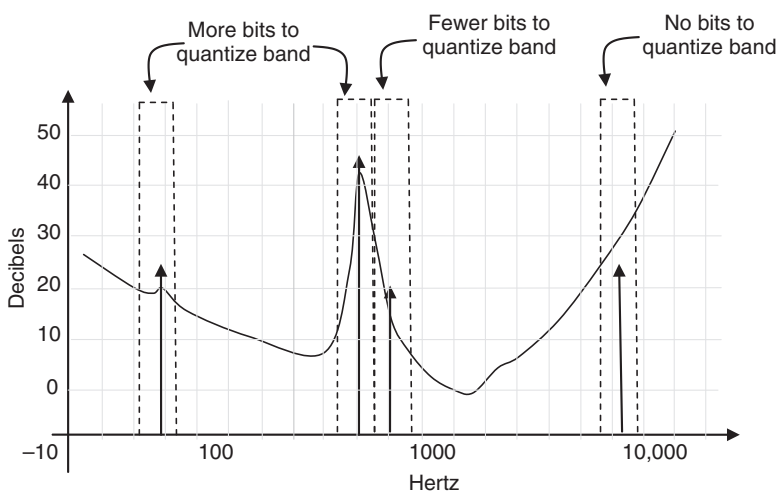


Figure 9-12 Bit distribution among different frequency bands. The frequencies above the psychoacoustically modeled threshold curve need more bits compared with bands at the threshold boundary or those below the boundary.

5 MODEL-BASED AUDIO COMPRESSION

This set of techniques follows a different approach by making use of the fact that the sound source is well known and, hence, can be parameterized or modeled. The encoders, also known as source coders, attempt to extract parameters of the model from the input signal that has to be coded. It is these parameters that are communicated to the decoder, rather than a set of quantized samples or frequency coefficients. The decoder also knows the model of the source and attempts to synthesize the sound using the received parameters. A simple example to illustrate this concept occurs when attempting to compress sounds produced by sirens. Sirens are physical instruments that synthesize sound. If a siren is recorded, and the sound has to be compressed, rather than compressing the time and/or frequency domain data, we could understand instead how the sound is produced, and describe the source by parameterizing the pitch, amplitude levels, periodicity, and other intonations in the sound. These parameters can then be used to synthesize the sound that produces the siren. Extracting and storing just these parameters achieves substantially more compression when compared with the traditional time and frequency domain methods.

This class of techniques is useful only for a known set of sounds whose sources are well understood. Human speech compression is another classic example that benefits from a sound model. Regardless of the language, human voice is produced by laws of physics, and can be modeled as air being pushed from the lungs through a vocal tract and out through the mouth. The vocal tract can be thought of as an open-ended tube open at the mouth with a buzzer at the other end. The vocal chords in the larynx act like a buzzer producing a buzz that can be parameterized by the frequency and amplitude. The throat and the mouth, which compose the vocal tract, act like a tube open at one end. This tube creates resonance effects, known as formants. Because the human speech creation can now be modeled as a parameterized buzzer, whose produced frequencies resonate, any speech signal that is produced can be analyzed to extract these formants. The contribution of the formants can then be removed from the speech signal by a process known as inverse filtering, creating a residue. The amplitude and frequency of the residue can then be estimated.

Linear predictive coding (LPC), which is considered one of the most powerful speech analysis techniques, is often used to imitate human speech production. It has two parts: an analyzer or encoder and a synthesizer or decoder. The encoder takes the speech signal, breaks it into blocks or audio frames, and analyzes each audio frame to determine if it is voiced (from the larynx) and unvoiced (silent). Voiced signals normally belong to vowels and are characterized by high amplitudes and lower frequencies. An example shown in Figure 9-13 is the "i" sound in *fish* or the "e" sound in *test*. Unvoiced signals are characterized by higher frequency and lower amplitude such as the "sh" sound in *fish* or the "s" sound in *test*.

The encoding process works by first finding the nature of excitation—whether the sound frame or segment is voiced or unvoiced. If voiced, the frequency period is computed, which can be used by the decoder to create a similar "buzz." The residue is computed by removing the synthesized "buzz" from the original sound. The residue can then be modeled by a few filter parameters. For every sound segment, the encoder codes and transmits the excitation state (voiced or unvoiced), frequency

parameters (if voiced), gain factors, and filter coefficients. Figure 9-14 shows the block diagram for an LPC-based encoder and decoder. The decoder works by synthesizing the buzz for voiced segments and using random noise for the unvoiced segments. These are then inverse LPC filtered to produce speech.

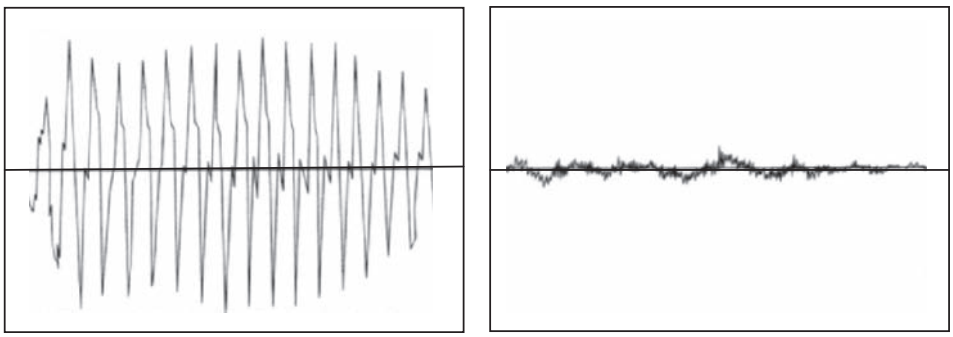


Figure 9-13 Sound signal of segment showing the "i" sound in fish (left) and the "sh" sound in fish (right). The left sound is "voiced" from the larynx showing high amplitude and low frequency. The right sound is "unvoiced" characterized by higher frequency and low amplitude.

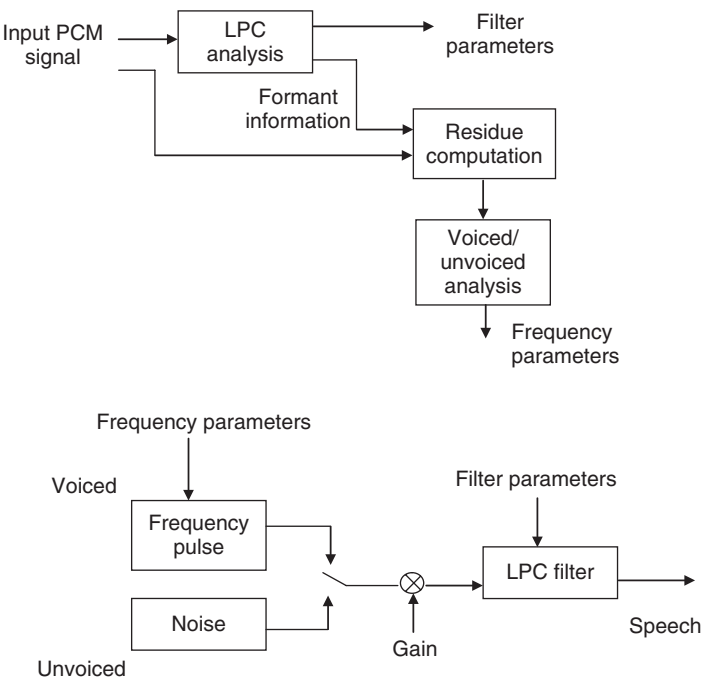


Figure 9-14 Model-based voice compression using Linear Predictive Coding (LPC). The encoder (above) computes the filter parameters that model the voice tract using LPC. The base residue is analyzed for voiced and unvoiced segments resulting in frequency parameters for voiced segments. The decoder (below) uses the frequency parameters to synthesize voiced and unvoiced speech segments, which are then inverse filtered to produce speech.

6 AUDIO COMPRESSION USING EVENT LISTS

This class of algorithms differs from the previous algorithms in a fundamental way. Rather than viewing the sound data as a sequence of signal samples that are analyzed either in the time or frequency domain, it *describes* the sound data using parameters that denote the music-making process as in a musical orchestra—the sequence of notes (frequencies) played, the timing between the notes, the mixing effects, the timbre, and so on. Timbre describes the envelopes that encapsulate differences of the same sound played on different instruments. As such, these well-structured representations do not have any sound samples, but rather are made up of semantic information that can be interpreted by an external model to synthesize sounds, as shown in Figure 9-15.

The encoder, thus, outputs a list of events, each of which is described by musical parameters, such as pitch or frequency of the note, decay time, mixing mode, and so on. The decoder synthesizes sound from this event list description by using an external instrument model. This model is often emulated as an external hardware synthesizer that takes a set of event list descriptions in an algorithmic synthesis language and synthesizes sound. The synthesis methodologies are explained in the following sections. Examples of standards that use these descriptions for encoding are MIDI (Musical Instrument Digital Interface), MPEG-4's SAOL (Structured Audio Orchestra Language), and SASL (Structured Audio Score Language). Event list representations are appropriate to describe sound tracks, piano concertos, and percussion instruments where each sound can be parsed as a separate event. However, continuous audio sounds such as those from a violin or flute are harder to describe.

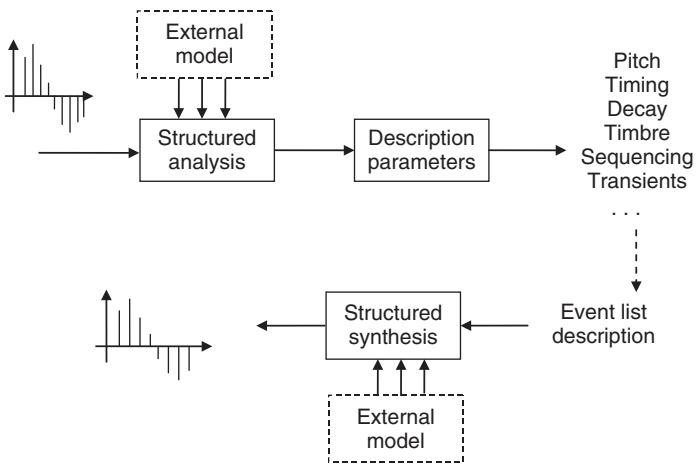


Figure 9-15 Event list encoder (above) and decoder (below). The encoder analyzes the sound to extract semantic information to create a list of events, each represented by musical parameters, such as pitch, time of decay, tempo, and so on. Any structured audio decoder (MIDI) takes this description and synthesizes sound based on instrument models (such as a piano, guitar, and so on).

6.1 Structured Representations and Synthesis Methodologies

Structured audio representations can be characterized by a number of features. This section enumerates some of the algorithmic representations of sound developed as music synthesis techniques, all of which use the underlying event list parameter model. Each method is suitably used for applications depending on the sound that must be generated and the parameter sophistication.

6.1.1 Additive Synthesis

Additive synthesis is a method of synthesizing sound using the basic sinusoidal fundamental frequencies. Here time-varying amplitudes and frequencies are superposed additively. Additive synthesis not only works well with periodic signals, but also with quasiperiodic musical sounds. The encoder analyzes acoustic data to extract parameters such as amplitude and frequency. The decoder resynthesizes sound by driving a bank of sinusoidal oscillators, which generate fundamental frequencies depending on these parameters. The sound of most musical instruments can be generated using additive synthesis; however, impulsive sounds cannot be realistically synthesized. Impulsive sounds require an infinite number of sinusoidal frequencies.

6.1.2 Subtractive Synthesis

The process of subtractive synthesis can be described by a rich harmonic source that is ultimately filtered. This is the same model that is used by speech encoders where the voiced speech created by periodic glottal excitations is filtered by the vocal and nasal tract. In music, the output of a periodic oscillator that uses fundamental sinusoids (or even saw tooth, square waves) is passed through filters and amplifiers. This method can synthesize many different sounds from a small number of model parameters.

6.1.3 Frequency Modulation

This is a very popular method of sound synthesis invented in the mid-1960s. Here, a sound oscillator generates a host frequency f_c, which later gets modulated by a modulating frequency f_m. If the amplitude of the modulator frequency f_m is zero, the output is the same as the host frequency f_c. A nonzero amplitude for f_m causes the output to deviate from f_c. For example, if $f_c = 256$ Hz, a modulation might cause the frequency to vary by a few Hz in either direction. Thus, by the use of one host or carrier frequency f_c and a modulator frequency f_m, many frequency sounds can be synthesized.

6.1.4 Physical Modeling

Physical modeling synthesis is a more sophisticated and recent methodology of synthesizing sound using equations and algorithms that emulate the actual physical structure of the source and the sound. The parameters used here describe the physical materials used in the instrument and its usage—for example, plucking a string of a guitar or beating a drum. In this case, to model the sound of a drum, an equation can describe how the drumstick transfers energy into the drum skin. Subsequently, the sound's movement and decay can be algorithmically described by the drum body dimensions, the drum skin properties (stiffness, mass, and so forth), how it resonates with the cylindrical drum body, and other boundary conditions. Similarly, you could model equations that can synthesize sounds for a violin, flute, and so on.

6.2 Advantage of Structured Audio

Structured audio applications have many advantages provided by their efficiency or conciseness in representation and the flexibility or availability of hooks that enable searching and modifying the audio signal. Some of the these advantages may be enumerated as follows:

- *Low-bandwidth transmission*—Much of the work around structured audio has been well received for its ultra-bit-rate compression. Bandwidth compression is an area in which structured audio techniques provide impressive gains over any other audio-encoding technique.
- *Music synthesis and creation*—The formal description that comes from structured audio representations makes it easy to help create musical harmonies in a similar way that a composer interacts with the performer. Structured audio systems can represent synthesis methods with expressive styles and yet allow the use of simple controls by composers.
- *Parametric sound generation*—Traditional waveform representations do not provide high-level control of sound compared with parametric representations. Applications like virtual environments, video games, and so on need sounds to be generated and/or parametrically controlled, which is where structured audio has a definite advantage.
- *Interactive music applications*—In such applications, a computer system listens to a human performance and adjusts is own performance in response. The adjustment here is dependent on the application or the end effect desired. For example, a structured audio system might be used as an automatic accompaniment to a performer.
- *Audio content-based retrieval*—The semantic descriptions provided by structured audio representations can be used for automatically analyzing and categorizing the digital content of musical harmonies.

Structured audio is definitely advantageous for a variety of applications that involve music where the signal can be interpreted as a sequence of sounds with properties that event list models can extract. In the next section, we discuss a variety of audio standards that use the different classes of audio-coding techniques discussed.

7 AUDIO CODING STANDARDS

The advances in audio-coding techniques have provided valuable compression necessary for many applications involving storage and transmission of digital audio. However, standards are necessary to make these applications a commercial viability. There are now many international and commercial audio-coding standards including the ISO/MPEG family, the ITU family, and stand-alone proprietary standards such as Dolby AC-3. These specifications were developed in collaboration with many research institutions and audio industry leaders, such as Phillips, Thomson Consumer Electronics, Fraunhofer Institute, and Dolby. Some of the more prevalently used standards are discussed in the following sections.

7.1 MPEG-1

The MPEG-1 audio-coding standard consists of three layers of audio-coding schemes with increasing complexity, resulting in successively better compression. These are better known as MPEG-1 Layer I, MPEG-1 Layer II, and MPEG-1 Layer III, popularly known as MP3. The layers are backward compatible, so that a compressed Layer 1 bit stream can be decoded by a Layer 2/Layer 3 decoder and a Layer 2 compressed bit stream can be decoded by a Layer 3 decoder. All the layers have a similar high-level architecture, which first applies a filter bank to the input to break it into different subbands. Simultaneously, it tries to apply a psychoacoustic model to the data as described in Section 4. The psychoacoustic analysis provides a bit allocation strategy that is used to quantize the different frequency bands. MPEG-1 operates in one of four possible modes: mono, stereo, dual channel, and joint stereo.

The block diagram for Layers I and II, which are very similar, is shown in Figure 9-16. The input goes through an analysis filter bank, which splits the incoming signal into 32 different equally spaced subband signals. Each band also has a decimated sampling rate, which is $1/32^{th}$ of the input. Each of the 32 subbands has 12 consecutive samples assembled into blocks with 384 (32×12) input samples. All the samples in one block are scaled by a normalization procedure to have their absolute values between 0 and 1. After normalization, the samples are quantized using a bit allocation strategy provided by a psychoacoustic model, which attempts to find out which bands are masked, requiring fewer bits or no bits when compared with perceptible bands. The psychoacoustic model uses a 512-point fast Fourier transform in Layer I (1024 point FFT for Layer II). Some of the salient features for Layer I can be summarized as follows:

- The encoding quality is transparent at 384 Kbps, which means that, to a human ear, the compressed audio is not differentiable from the original input.
- Subband coding is employed by dividing the frequencies into 32 bands (12 samples/band), and each band quantized differently.
- Coefficients are normalized to extract the scale factor.
- For each block of 32 bands, the psychoacoustic model provides one of 15 quantizers.

Layer II introduces further compression to Layer I for a few reasons. First, the 1024-point FFT (compared with 512 in Layer I) provides an improved precision for the psychoacoustic process, thus allowing a better bit allocation strategy. Also, the scale factor processing is different, removing more redundancy in the scale factor data. Some of the salient features of Layer II when compared with Layer 1 are as follows:

- The encoding quality is transparent at 296 Kbps.
- Like Layer I, Layer II also employs subband coding using 32 channels.
- The quantizer has finer resolution compared with Layer I. This is because of the 1024-sample FFT instead of the 512-sample FFT.

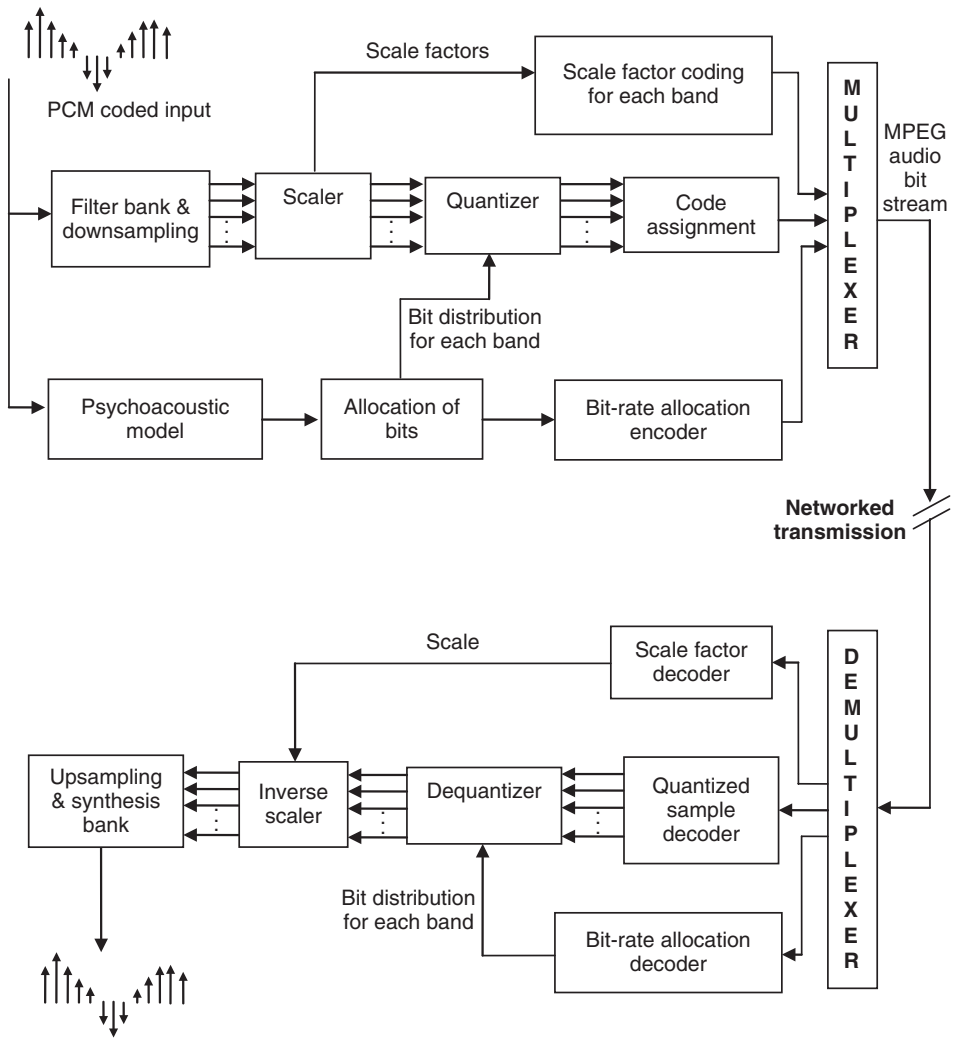


Figure 9-16 MPEG audio architecture for Layer I and Layer II. The encoder (above) divides the input into subbands, each of which gets quantized depending on the psychoacoustic analysis. The decoder (below) is less complex than the encoder.

The MPEG-1 Layer III audio coder is more complex compared with its predecessors. A block diagram is shown in Figure 9-17. The Layer III encoder is more popularly known as the *MP3* encoder. It has a hybrid filter bank providing a higher frequency resolution. The 32 subband outputs of the filter bank are input to an 18-point Modified Discrete Cosine transform (MDCT) module. The MDCT is a refined frequency transform that addresses problems the DCT has at boundaries between the windows used. Quantizing the frequency coefficients from a DCT in two windows independently and transforming it back to the time domain produces discontinuities between the samples in the time domain at the boundary between two windows. This

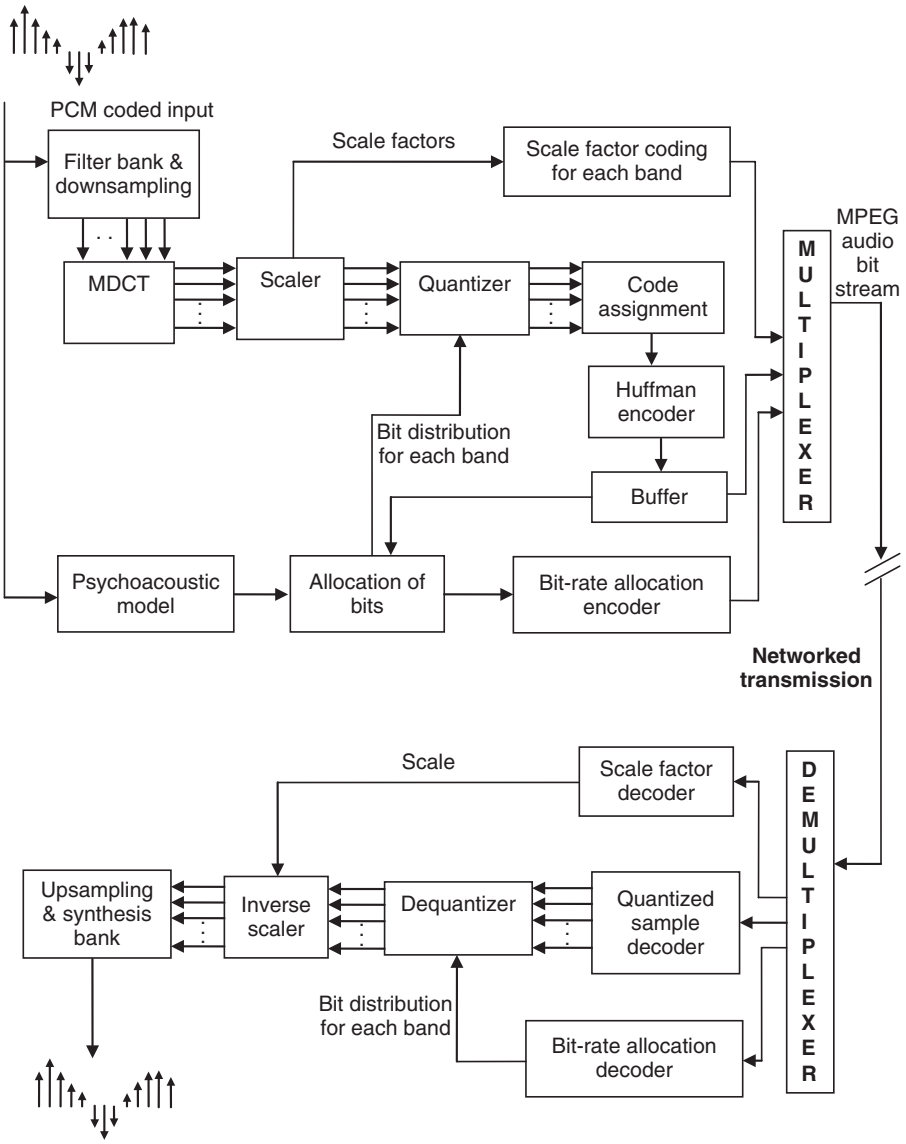


Figure 9-17 MPEG audio architecture for Layer III (MP3). The encoder (above) is more complex compared with the initial layer encoders because of the MDCT module and the entropy coder (Huffman). The decoder (below) is similar to the Layer I and Layer II decoders.

results in perceptible and periodic noisy clicks. The MDCT shown in the following equation removes such effects by overlapping frames by 50%. Compare this equation with the one discussed in Section 8 of Chapter 7.

$$F(u) = 2\sum_{i=1}^{i=N} f(x)\cos\left[\frac{2\pi}{N}\left(x + \frac{N/2 + 1}{2}\right)\left(u + \frac{1}{2}\right)\right], \mathrm{u} = 1, 2, \ldots N/2$$

The psychoacoustic processing module provides a better distributed bit budget to quantize the subband MDCT coefficients. Additionally, the quantized coefficients are entropy coded using Huffman coding for further savings in bit rates. Some of the salient features of the MP3 encoder are as follows:

- The output quality is transparent at 96 Kbps per channel.
- It applies a variable-sized Modified Discrete Cosine transform on the samples of each subband channel.
- It uses nonuniform quantizers.
- The quantized outputs are entropy coded (Huffman) resulting in a variable bit rate and requiring the need for buffering.

7.2 MPEG-2

MPEG-2 differs from MPEG-1 mainly because it supports a surround sound 5.1 channel input. This includes five channels (front left, front center, front right, back left, back center) and an optional low-frequency enhancement (*subwoofer*) channel. The multichannel support creates an improved realism of auditory perception along with visual imagery, for example when viewing HDTV or a DVD. The MPEG-2 audio coding standards are categorized into two groups: the MPEG-2 BC or backward compatible, which preserves compatibility with the MPEG-1 standards, and the AAC (Advanced Audio Coding), which is known as NBC or non–backward compatible. AAC is the audio-coding technology for the DVD–audio recordable (DVD-AR) format and is also adopted by XM Radio, which is one of the main digital broadcast radio services in North America. MPEG-2 BC is very similar to MPEG-1, except that MPEG-2 BC can encode different sampling rates. With the extension of lower sampling rates, MPEG-2 BC now makes it possible to encode two-channel audio signals at bit rates less than 64 Kbps.

The MPEG-2 AAC standard provides a higher-quality encoding for audio bit streams where compatibility with MPEG-1 is not a constraint. MPEG-2 AAC provides very good quality at data rates of 320–430 kb/s for a 5.1 multichannel system, which is less than half the data rate obtained by the MPEG-2 BC version. For stereo channel encoding, it can deliver high-quality encoding at bit rates below 128 kbps. The AAC encoder block diagram is shown in Figure 9-18. Compared with the MPEG-2 BC layers, the additional modules that aid in the compression are as follows:

- *Temporal noise shaping (TNS)*—After conversion to the frequency domain, the temporal noise-shaping module helps to control the temporal shape of the quantization noise.
- *Stereo intensity coding*—The stereo intensity module reduces the perceptual irrelevancy between the channels by combining multiple channels in the high-frequency regions.
- *The prediction module*—Adjacent audio frames might have high correlation after stereo intensity coding. The prediction module removes this redundancy.
- *M/S coding*—The M/S coding codes stereo channels. Instead of coding the left and right channels separately, redundancy is removed by coding the sum and difference between the channels.

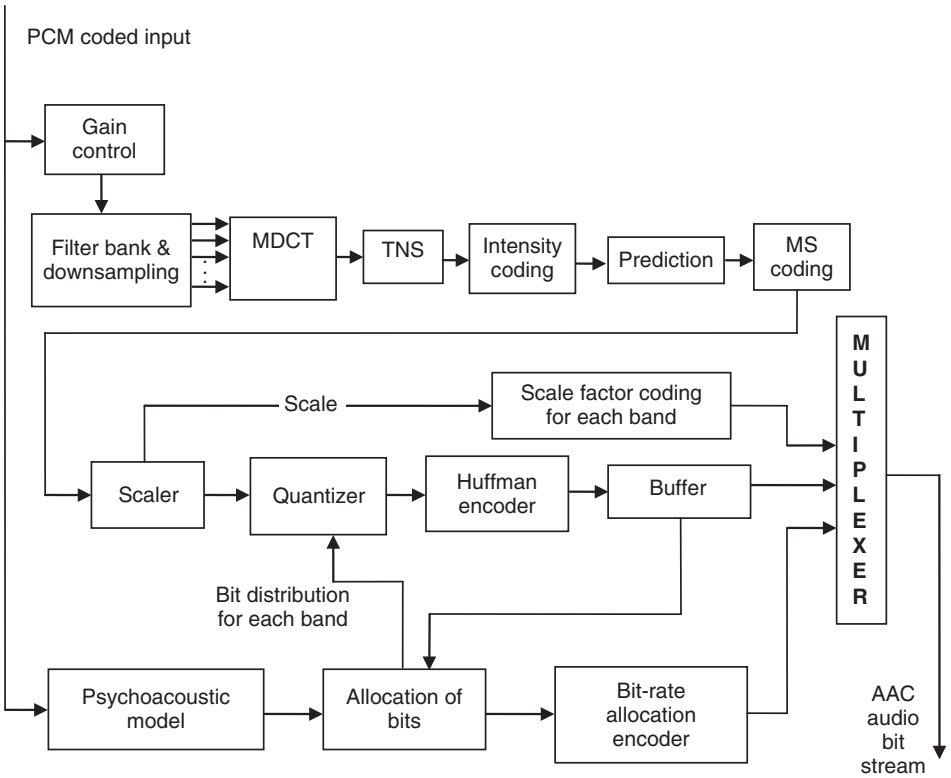


Figure 9-18 MPEG-2 AAC encoder. Most of the modules, such as the psychoacoustic, quantizer, and Huffman, are similar to the MPEG-1 layers. AAC gains more compression by temporal noise shaping (TNS), stereo intensity coding, prediction, and MS coding modules.

Further processing modules—quantization, bit allocation based on the psychoacoustic analysis, and variable-length Huffman coding—are very similar to those used in MPEG-1 Layer III.

7.3 Dolby AC-2 and AC-3

The Dolby encoders use the same psychoacoustic principles as the MPEG encoders. The Dolby AC-2 encoder has a low complexity operating at 128–192 Kbps per channel and is used in point-to-point delivery and cable, whereas the AC-3 encoder has a higher complexity operating at 32–640 Kbps per channel. The Dolby AC-3 block diagram is shown in Figure 9-19. As illustrated, the AC-3 encoder first employs a Modified Discrete Cosine transform and transforms the input channels to the frequency domain. When compared with the MPEG audio encoders, one of the main differences is the floating-point representation used for the frequency coefficients with one or more mantissas per exponent. The psychoacoustic model uses the exponents to compute a perceptual resolution for its analysis to find masked frequencies.

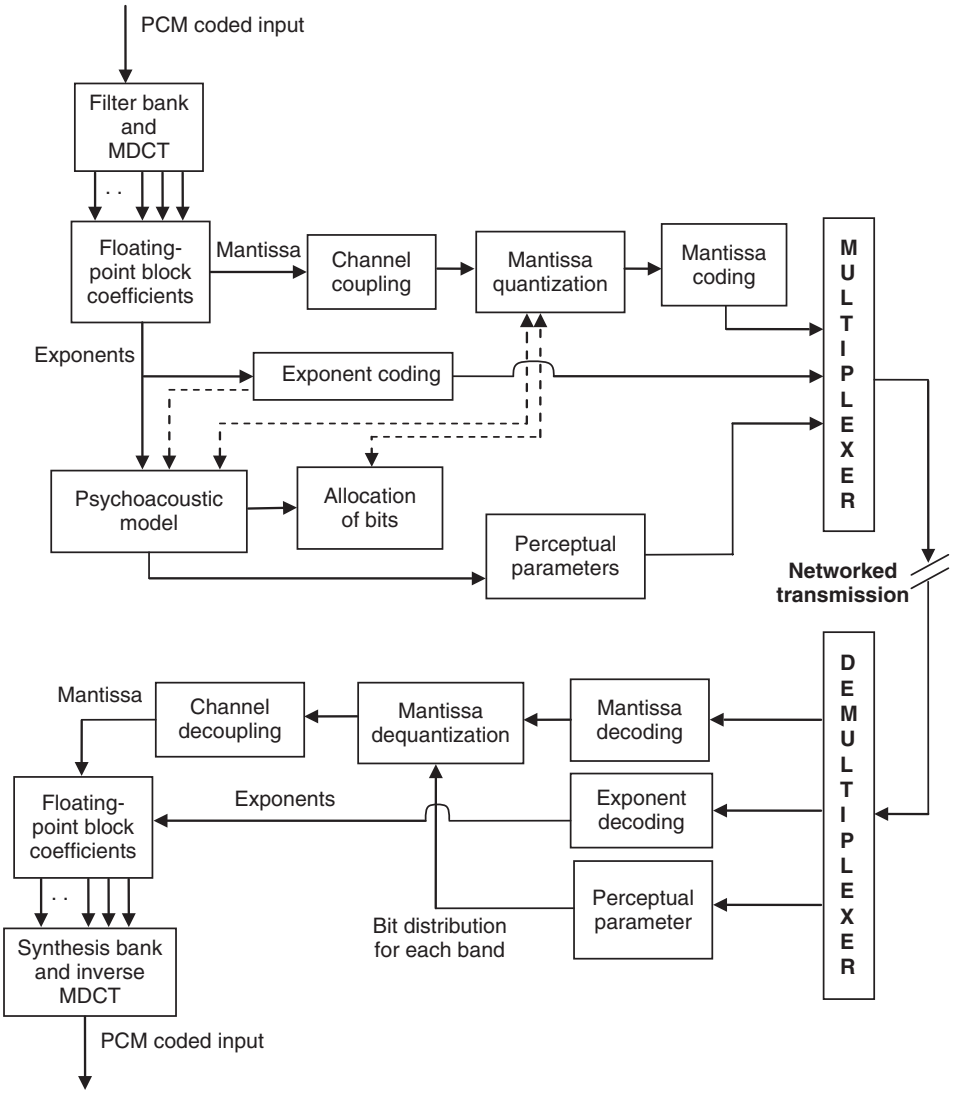


Figure 9-19 Dolby AC-3 architecture. The encoder (above) processing quantizes floating-point mantissa and exponent. The exponents are used in the psychoacoustic analysis. The decoder is shown below.

There is a tight connection between the exponent coding, psychoacoustic models, and the bit allocation, which can be seen by the hybrid backward/forward bit allocation.

The encoded exponents give the spectral envelope, which is used by the psychoacoustic module to help in the bit allocation for the mantissas for each band. Another distinguishing advantage of the Dolby AC-3 encoder is that it can automatically derive the quantizer information from the decoded exponents and limited perceptual parameters. This is unlike the audio encoders in the MPEG family, which need to transmit

the bit-rate allocation used to quantize all the bands. On the other hand, because only the exponents are used in the perceptual analysis, there is limited resolution in the perceptual analysis compared with the MPEG audio encoders.

7.4 MPEG-4

The MPEG-4 standard has numerous parts, where each part standardizes various media-related entities, such as video, audio, images, graphics, storage file format, and so on. Compared with the precursor MPEG standards, MPEG-4 differentiates itself as a true multimedia standard dealing with all media elements and Chapter 14 is dedicated solely to explaining the various parts of the standard. The MPEG-4 standard covers a broad set of issues and is divided into more than 20 parts. Of these, part 3 deals with audio and specifies audio bit streams and coding methods devised for low-band speech and high-band music. The two common speech codecs here include Harmonic vector excitation coding (HVXC) and code excited linear predication (CELP). HVXC supports speech bit rates as low as 2 Kbps. CELP works similarly to LPC coding as explained in Section 5 along with mutipulse excitation (MPE). The high-band coding methods include an enhanced version of MPEG-2's AAC. AAC's multiple codecs include the following:

- Low complexity advanced audio coding (LC-AAC)
- High efficiency advanced audio coding (HE-AAC)
- Scalable sample rate advanced audio coding (AAC-SSR)
- Bit sliced arithmetic coding (BSAC)
- Long term predictor (LTP)

HE-AAC uses spectral band replication to increase coding efficiency at lower bit rates. AAC-SSR splits the input audio data into four bands first and then each band is split using 32 or 256 sample windows. The two-stage demarcation allows appropriate enhancing of temporal resolution for high-frequency data, whereas the low frequency can still be encoded with higher spectral resolution BSAC. This is similar to AAC, except that the use of an alternative noiseless coding produces transparent sound quality at 64 Kbps and shows a graceful degradation at lower rates. MPEG-4 also specifies a text-to-speech synthesis module. A few of these audio standards in MPEG-4 have been addressed in more detail in Chapter 14.

7.5 ITU G.711

The ITU G.711 standard is the most common compression algorithm for telecommunication networks worldwide. It is designed for telephone bandwidth speech signals sampled at 3 KHz, and provides the lowest delay possible (1 sample) at the lowest complexity. It does a direct sample-by-sample nonuniform quantization of the PCM input signal, similar to the A-law and μ-law illustrated in Section 3.1. The G.711 scheme has lower processor utilization due to lower complexity, but a higher IP bandwidth requirement. It was primarily used for video telephone over ISDN and supports a bit rate of 64 Kbps.

7.6 ITU G.722

The ITU G.722 standard was designed to transmit 7 KHz voice or music and is often used in videoconferencing systems where higher audio quality (compared with the G.711) is required. The voice quality is good but the music quality is not perfectly transparent due to a 7 KHz sampling rate. Frame delays are tolerated and it supports bit rates from 48 to 64 Kbps. G.722 operates by dividing the signal into two bands—high pass and low pass—which are then encoded with different modalities. The G.722 standard is preferred over the G.711 PCM standard because of increased bandwidth for teleconferencing type applications.

7.7 ITU G.721, ITU G.726, and ITU G.727

The G.721 standard was established in the 1980s and was a precursor to the G.726 and G.727. All of these are also used for telephone bandwidth speech and differ from the previous ITU standards by using adaptive differential pulse code modulation techniques (Section 3.3). These standards show how a 64 Kbps A-law or μ-law PCM signal (G.711) can be converted to a 40, 32, 24, or even 16 Kbps signal using ADPCM. These correspond to 5, 4, 3, and even 2 bits per sample.

7.8 ITU G.723 and ITU G.729

The G.723 and G.729 are model-based coders with special models of speech production (Section 5). These coding standards provide good-quality speech with only moderate processing requirements and time delays. The voice models also perform well in the midst of random bit errors. A few notable features of these codecs are as follows:

- All these standards are based on the Code Excited Linear Prediction (CELP) model.
- The G.723 and G.729 perform compression/decompression of 8 KHz speech signals and convert input samples into 16-bit code words yielding 12 code words every 240 samples.
- The G.723 codec operates in one of two bit-rate modes: 5.33 Kpbs or 6 Kpbs. The G.729 standard offers a voice data rate of 8 Kbps.
- The standards also have provisions to deal with frame loss and packet loss while communicating over the Internet.
- G.723 is also part of the H.324 multimedia standard for communication over POTS (plain old telephone service) with a modem.

7.9 ITU G.728

The G.728 standard is a hybrid between the higher bit rate ADPCM coders (G.726 and G.727) and the lower bit rate model-based coders (G.723 and G.729), which are based

on code excited linear prediction models. Some of the salient features of the G.728 codec are as follows:

- It uses 8 KHz speech data as input, and has a low delay but fairly high complexity.
- It operates at 16 Kbits per second and is considered equivalent to the performance of G.726/G.727 at 32 Kbps.
- The G.728 coders are based on a Low Delay Code Excited Linear Prediction model (LD-CELP).
- It is used in ISDN video telephony, performing very robustly in the presence of noise and random bit errors.

7.10 MIDI

Musical Instrument Digital Interface, or MIDI, first appeared in the early 1980s as the digital audio and CD industry exploded. With the advances in digital audio, musical instruments were also being manufactured to play digital music. As musical instrument manufacturers began to manufacture electronic instruments that could create digital sound, it became clear that there was a language needed that these musical instruments could interpret themselves and also use to talk to one another. MIDI was established as an international standard with the involvement of music instrument manufacturers such as Roland, Yamaha, Korg, and Kawai among others, who later released MIDI-capable digital instruments. The MIDI system specification consists of both hardware and software components, which define interconnectivity and communication protocols and bit streams for electronic synthesizers, sequencers, personal computers, rhythm machines, sound card machines, and other musical instruments. This allows a variety of instruments to "talk" with each other. The interconnectivity defines standard cable connectors and input/output circuitry between two machines or instruments.

The MIDI protocols do not transmit audio waveform signals, but a digital description (as explained in Section 6) of the music performance in the form of multibyte messages. These messages are received by the MIDI sequencer and processed asynchronously in real time. These MIDI message signals are of two kinds—channel messages and system messages. Channel messages use one of 16 channel identifiers where each channel is used to separate instruments, somewhat like tracks in a multitrack mixer. The ability to multiplex 16 channels on a single wire allows several instrument messages to be sent on a single MIDI connection. The channel messages control the instrument in a variety of ways to do the following:

- Switch notes on or off.
- Send channel pressure messages—used to control the output intensity. There are also aftertouch messages (polypressure messages) that are sent in some instruments to indicate changes while a note is being played.
- Control messages are used to manage effects like tremolo, vibrato, sustain,and so on.
- Special pitch-wheel messages are used to change the frequency of all notes.
- Program change messages are used to change the instrument of one particular channel to another instrument.

MIDI did enable electronic music to flourish by giving a way for musicians to combine the sounds of different electronic instruments. However, there are a few drawbacks. First, the MIDI format does not contain actual digitally encoded sounds but only instructions that are interpreted by devices to render sounds using its own library of cached sounds. As a result, the same file might sound different on different devices. Just as different musicians might play the same note in different ways, different makes and models of sound cards can vary in the sound they reproduce to emulate a musical instrument.

Another drawback of MIDI is that it seems limited in the depth and detail of musical expression that can be created with it. This is because it has a limited sound resolution and time resolution. The sound resolution is limited to 128 values (7 bits), which is also very rough because the dynamic range of sound from an instrument can be perceived very finely by our ears. Timing is very important to get a musical melody just right. Because MIDI stores instructions that are executed sequentially, encoding simultaneous instructions is not accurately achievable and delays of a few milliseconds are possible. As a result, smoothly varying sounds, such as produced by a violin or a flute, do not always sound as perfect as the original when encoded using MIDI. MIDI is more suited to discrete sound-producing instruments such as a piano.

8 EXERCISES

1. [**R02**] While performing audio encoding and decoding, the complexity of the encoder is not the same as the decoder. Which one is more complex and why?
2. [**R03**] The earlier speech standards such as G.711 treat the audio signal as a simple waveform. Waveforms achieve compression by statistical analysis.
   - How does G.711 achieve compression?
   - Compared with the other waveform standards, G.711 achieves a superior quality. What assumptions about audio signals does G.711 make and how are these assumptions used?
3. [**R03**] Consider an application where you want to set up a music database. What kind of audio-encoding scheme (from DPCM, perceptually encoded audio, MIDI) would you incorporate and why?
4. [**R04**] Both the visual image/video encoders and the psychoacoustic audio encoders work by converting the input spatial or time domain samples to the frequency domain and quantize the frequency coefficients.
   - How is the conversion to the frequency domain different for visual encoders compared with audio encoders? Why is this difference made for audio encoders?
   - How does the quantization of frequency coefficients in the psychoacoustic encoders differ from that used in the visual media types?
   - Why is it necessary to transmit the bit-rate allocation in the bit stream for audio encoders? Does this have to be done in the beginning, toward the end, or often? Explain. How do the visual encoders convey the bit-rate allocation?

5. [**R04**] This question deals with your understanding of loudness and the decibel scale that is used to measure loudness.
   - What do you mean by decibel? Is it an absolute or relative scale?
   - If the amplitude of a signal is higher, does it necessarily mean it is louder?
   - How does loudness relate to frequency? If you consider signals at 100 Hz, 1 KHz, and 10 KHz, all at 50 dB, which one is louder?
   - Figure 9-7 shows the threshold curve in quiet. This gives the decibel levels at which the sounds at all frequencies are just audible. We can extrapolate this and think of curves that show equal loudness perception for different frequencies. How might these curves look? These are known as Fletcher Munson curves of equal loudness. Go to you favorite search engine and search for information related to these curves.
6. [**R05**] The MP3 or MPEG-1 Layer III encoder is considerably more complex than the preceding layers.
   - Identify and explain the functionality of the modules that adds to this complexity.
   - Why does MP3 do better than Layer I and Layer II?
   - One striking feature of the MPEG-1 encoder, which makes the overall architecture more complex to manage during encoding, is the feedback arrow from the output of the Huffman coder back into the psychoacoustic bit-rate allocation module. Why is this needed?
7. [**R04**] In perceptual audio encoding, masking is primarily used to control compression.
   - What is masking? Give examples of three different masking examples.
   - How is the quiet threshold curve used to evaluate masking in frequency?
   - For the frequency distribution shown in Figure 9-20, which ones require more bits and which ones require less or no bits? Arrange frequencies in increasing order in terms of bit requirements.
8. [**R06**] Vocoders use a model of the human voice tract to analyze, encode, and synthesize human speech. One of the primary steps in this process is separating speech into *voiced* and *unvoiced* parts.
   - What are *voiced* and *unvoiced* parts of speech?
   - Suggest an algorithmic strategy that you can use to distinguish between voiced and unvoiced parts of the signal.
   - Normally when coding, the input signal is divided into uniform segments and each segment is decided as being voiced or not. For this process, is it better to have longer or shorter segment lengths?
   - What happens when a sound is distributed at the boundary between two segments?
   - Can you think of any sounds that are harder to say voiced or unvoiced?

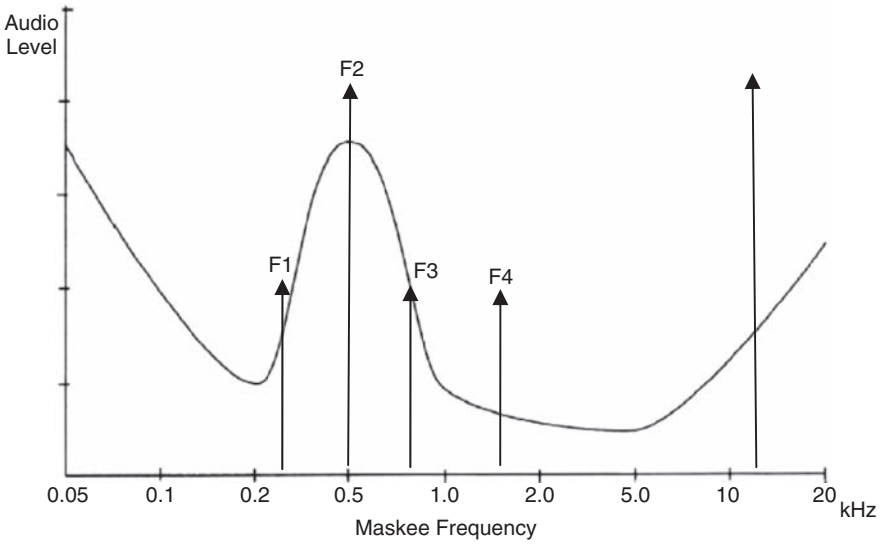


Figure 9-20

9. [**R06**] The Dolby AC-3 encoder also uses psychoacoustics to encode audio signals like the MPEG encoders.
   - What is the main difference in the frequency domain representation of the two encoders? From the block diagrams shown, can you see what might be semantic differences in the bit streams?
   - How is the psychoacoustic processing different in the two encoders? Which one is better?
   - Mention what factors affect the output of the AC-3 encoder when compared with the MPEG MP3 encoder. When will the AC-3 encoder give better bit rates when compared with the MP3 encoder and vice versa?
10. [**R05**] Model-based audio compression (LPC) and event list representations encode audio in very similar ways. Both analyze the input sound and try to fit a parameterized model. The extracted parameters are then used to synthesize the decoded sound. As such, they might represent in the same category. Mention a few salient points that necessitate differentiating the two methodologies.
11. [**R04**] Event list representations and structured audio are used to compress music sequences by semantically describing them, rather than compressing the time/frequency domain characteristics of the signal. The impressive compression ratios obtained here would naturally lead you to wonder why we cannot compress other sounds accordingly.
    - Why do you think speech can or cannot be compressed as event lists?
    - Instead of event list descriptions, what other descriptions are suited to describe speech? (*Hint*: Think in terms of parts of speech or even text.)

- How is this different (or similar) to vocoders that use LPC models for compression?
- The advantage of having a speech compressor that encodes by describing speech rather than other standard audio-coding mechanisms is obviously compression gains. Describe the limitations you see in this and, thereby, what applications you think it will be useful for.

12. [**R03**] The MIDI standard is used to standardize semantic descriptions of music. At the heart of the MIDI standard are communication protocols that send messages.
    - How many different kinds of messages are there?
    - What is the similarity and difference between these two messages: *Channel Pressure* message and *Aftertouch* messages? Which types of instruments would use these?
    - Why is the program change message necessary?
    - Sometimes each synthesizer manufacturer defines Manufacturer's System Exclusive messages (SysEx). These messages are commonly used to send non-MIDI data over a MIDI connection. What might these messages be useful for?

PROGRAMMING ASSIGNMENTS

13. [**R04**] Source code that allows you to read a pulse code modulated signal file and play it using Microsoft's Direct Sound technology is available in the Student Downloads section at www.cengage.com. Sample sound PCM files with descriptions (sampling rate, sample size) have been provided there.
    - Modify the program and vary the sampling rates to see how audio sounds at the correct versus incorrect sampling rates.
    - Modify the program to take two files and play them back to back. Assume that they are of the same sampling rate.
    - Now implement algorithms to fade out and fade in one file into another.
    - If your sound player was always *fixed* to play at 8 KHz, what would you do to play sounds sampled at higher or lower sampling rates?

CHAPTER 10

Media Compression: Graphics

Synthetically generated objects can be represented graphically and this is a topic relevant to the field of computer graphics. Whereas the early days saw the development of 2D objects only, the emphasis is now on 3D objects. These objects involve vector representations and have been at the heart of multimedia development since the beginning of multimedia. In the early 1990s, most of the multimedia CD-ROM content, including games, educational content, presentations, and so on, involved many aspects of 2D graphics and animations along with sound, video, and other media types. Games that involved 2D graphics, animations, and interaction have been used for a variety of entertainment on PCs and dedicated game machines, such as Nintendo, Sega, and now SONY PlayStations and Xboxes. Graphics and animations now play a de facto role in creating advertisements, presentations, graphs, games, user interfaces, and so on. A variety of authoring tools for professionals and novices allow creation of 2D graphics and animation content—Adobe Photoshop, Adobe Illustrator, Microsoft PowerPoint, Microsoft Word, and Macromedia's authoring tools. Although not an exhaustive list, it does indicate that graphical objects have become an integral part of representing information and interacting with it. Figure 10-1 shows an example of a 2D vector illustration.

3D representations have also gained in popularity and are now used for a variety of applications, including games, computer-aided design and manufacturing, advertising, 3D featured animations in the movies, special effects industry, medical, defense, museum archiving, and so on. The use of 3D objects in all these industries has evolved from the use of simple 3D objects and art forms to more sophisticated and precise models that replicate real-world objects to an unprecedented exactness. Figure 10-1 shows an example of a 3D representation. The process of creating such complex 3D objects has never been easy, but the advances in both 3D-scanning technologies as well as

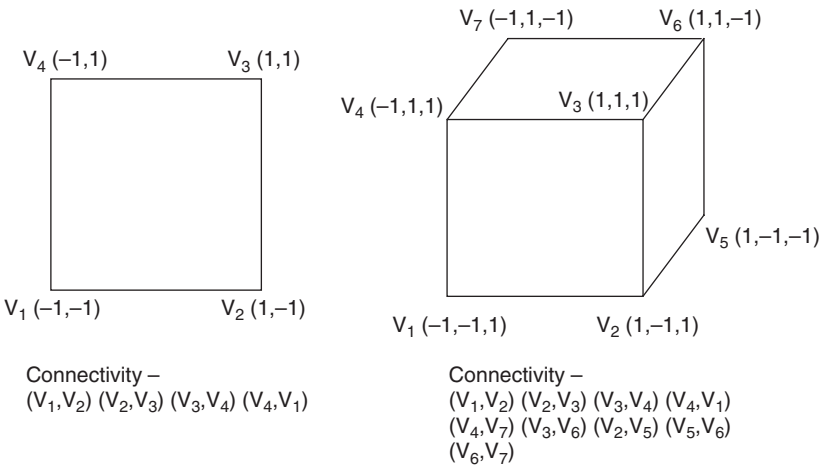


Figure 10-1 Example of a 2D vector representation (left) and 3D representation (right). These vector representations provide more precision and potential to interact and hence are unlike the other visual media types, such as images and videos.

authoring software used to create 3D representations have made it easy to create 3D models. Furthermore, the advances in graphics technology and user interfaces have made it easy to visualize and animate 3D objects. For example, software applications such as Autodesk Maya used in entertainment, Autodesk AutoCAD, and Metrix Imageware Surfacer used in design and manufacturing all now have mature features and interfaces to help create, edit, and animate 3D content. Also, with the affordable costs of such devices, graphics hardware is now a standard part of all PC desktops, workstations, and laptops. All this hardware and software evolution has not only increased the use of 3D graphical objects, but has also increased the representational details and the achievable realism.

From a distribution point of view, it is important to note that in the early 1990s, 2D content was initially only distributed on CDs in the form of CD-ROMs. These vector-represented graphics objects and animations had modest memory requirements and hence were rarely compressed. However, the proliferation of the Internet and the increased complexity of the content created since then have imposed compression requirements on the 2D and 3D content. Today almost all applications that distribute 2D/3D graphics and animation do so in a compressed format. Proprietary standards include Adobe Shockwave/Flash content (formerly Macromedia) and Viewpoint Experience Technology, but there are also standards now in place for 3D, such as VRML, MPEG-4, X3D, and JAVA 3D.

This chapter doesn't discuss 2D graphics compression in detail, but considers them a simpler case of 3D graphics. Also, although you can represent 3D objects in many different ways, this chapter concentrates on the most common representation, which consists of 3D triangles because polygonal representations have become a de facto standard for representation in computer graphics. Section 1 details the need for graphics compression by illustrating memory requirements of typical graphical objects in different

industries. Section 2 discusses some of the general features that relate to graphics compression and discusses the different ways of representing 3D objects. Section 4 also describes the choice of triangulated meshes as standard 3D representations and gives an overview of the different types of 3D graphics compression techniques. Sections 5, through 8 illustrate examples of algorithm in each category—topological surgery, progressive meshes, and wavelet-based compression as well as a few issues in progressive encoding of 3D objects. Finally, Section 9 discusses standards around 3D compression, which include the X3D, MPEG-4 (in part), and Java 3D standards.

1 THE NEED FOR GRAPHICS COMPRESSION

Detailed geometric 2D and 3D models have become commonplace in most computer graphics–related industries, and are distributed using the Internet. For most applications, the realism required in terms of a visualization experience or an interactive experience is achieved by increasing fine details in geometric representation. You can represent graphical objects in a variety of ways, such as polygons and triangles, splines, constructive solid geometry, and so on. These are discussed in Section 2. The objects can be created using a variety of software tools that take simple geometric primitives, such as a sphere, a cube, or a plane, and produce complex, detailed models by applying versatile modeling operations. The modeling operations provided depend on the underlying 3D representation, for example manipulation of point, face locations, smoothing, and extrusion are common mesh operations, whereas control point manipulation and stitching are common operations on spline and patch representations. In addition, detailed 3D representations can also be obtained by range scanning systems, which create sampled 3D points of an object's surface. The points are then assembled together into one of the above-mentioned representations. Whether interactively created starting from basic primitives, or approximated from scanned data, the resulting complex representations (which are normally polygonal meshes) lead to the following practical problems:

- *Storage and networked transmission*—The storage requirements for complex meshes can be formidable. Delivering the voluminous geometry data is also a serious problem, especially when the data is in a form that can be manipulated and edited. In networked multiplayer games, for example, the size of the geometry can interfere with tolerated real-time latency. Compression is needed to minimize the storage space and resolve the undesired latencies.
- *Level of detail*—3D models are visualized by rendering, the complexity of which is measured as a function of the number of polygons rendered. To improve rendering performance, it is desirable to have several different versions of the object from coarse approximations to detailed representations, so that when objects are far away, coarser representations are rendered, and detailed representations are necessary only when the objects get closer to the viewer. These hierarchical representations are useful for rendering as well as for progressive transmission, when coarser approximations can be transmitted first, followed by details to be added later.

- *Mesh simplification*—Often, meshes created by scanning processes have many points that are distributed all over the object's surface. Frequently, these points don't relate to any geometric detail and need to be either removed or merged with adjacent neighboring vertices to simplify the representation. For instance, a planar patch can be represented with very few triangles, even if it contains thousands of points. Such simplification operations might need *remeshing* of the data to create simpler representations.

The techniques used to address the above-mentioned issues are closely related to the underlying representation used for 3D objects. Although you can represent 3D objects in many ways, as discussed in Section 2, here we consider the most common form of representation in terms of triangular meshes. Polygonal models are normally represented in the form of quads and triangular meshes, where triangles are the atomic graphical units used for a variety of animation- and rendering-related tasks. When these meshes have to be transmitted over networks for purposes of visualization and interaction, it presents serious challenges, as illustrated in Figure 10-2, to efficiently control the storage capacities and real-time needs of the application.

Graphics data type	A simple 2D graphic	A character used in a 3D game	Models in 3D movies	CAD/CAM model
Number of polygons (normally triangles or quads)	Less than 500	4000–5000	20,000–50,000	2,000,000
Approximate number of vertices	250	2500	25,000	500,000
Approximate file size in bytes (uncompressed)	10 KB	100 KB	1 MB	30 MB
Transmission times for one second of data (56 Kb modem)	1.43 seconds	14.28 seconds	142 seconds	4286 seconds
Transmission times for one second of data (780 Kb DSL)	0.1 seconds	1.06 seconds	10.2 seconds	307 seconds

Figure 10-2 Examples showing storage space, transmission bandwidth, and transmission time required for uncompressed 3D objects, assuming that the data consists only of geometry with no textures

Next, we discuss representations of graphical objects and algorithms commonly used to compress these representations. There are 2D graphical objects and 3D graphical objects with 2D normally considered as a simple case of 3D. You can represent graphical objects in many ways, depending on the application and precision

of measurements involved. Section 2 and Section 3 enumerate some of the commonly used representations. Of all these, polygonal representations have become a choice for most industries because polygonal rendering is supported by hardware graphics cards on computers, laptops, game consoles, and so on, making it easier to create imagery. In addition, many graphics software programs have intuitive interfaces and powerful operations to create 3D content using polygons. We first discuss 2D representations in Section 2 and then extend the analysis to 3D representations in Section 3.

2 2D GRAPHICS OBJECTS

In 2D, the only geometric elements are points, regions, and curves. These are represented as vector coordinates. For visualization, these entities get rendered on an image grid that consists of $n \times m$ pixels.

2.1 Points

Points can be represented by their Cartesian coordinates (x,y). A pixel with these coordinates can be highlighted to render a point on a graphics display. Points can be compressed by quantizing these values. A common approach is to use integer coordinates in an n-bit range.

2.2 Regions

Regions can be defined by the list of points they contain. Enumerating a list of points to represent a region is a simple representation but quite inefficient in terms of storage. A number of schemes exist to compress this information. A simple way to achieve compression is to consider the underlying grid of 2D points and encode the grid representation. For example, one way might be to run length encode 2D scans of segments, which can be encoded by listing the beginning point and end point in each segment. Effectively, what is preserved is the list of boundary points of the 2D region, one line at a time. Alternatively, a region can be represented hierarchically by a tree, with the root representing the smallest square bounding the region, the next level representing a subdivision of the shape into four squares, and so on until the finest level of resolution is achieved. This representation is called a quad-tree. A more compact representation can be obtained by observing that regions are fully defined by their bounding curves, which we discuss next.

2.3 Curves

A straightforward way to represent a curve is by an ordered list of coordinates $C = \{x_i,y_i\}$ with $i = 1 \ldots n$. Such a curve can be efficiently represented instead using a chain code. The starting point (or an arbitrary point, in the case of a closed curve) is represented by its coordinates, which requires 16 bits in a 256×256 grid, 8 bits for each coordinate. Successive points can then be represented by

a relative displacement vector. There are eight directions around a point, which takes 3 bits for each connected point. Another approach is to approximate the curve using polynomials of different degrees. The simplest such approximation uses lines (polynomials of degree 1), also called a polyline capturing a sequence of line segments by a list of points. Such an approximation can be achieved with any desired degree of accuracy. Although it is straightforward to generate a curve with line segments, there is no unique algorithm to infer a description from a given curve. An iterative algorithm proceeds by choosing a starting point, then moving the line along the curve until the approximation error exceeds a predefined threshold, then starting a new line segment, until the end point (or the starting point in the case of a closed curve) is reached. A recursive algorithm proceeds by drawing a line between end points and then computing the distance from every point to this line. If it is below a predefined threshold for all points, you are done; otherwise, choose the point with the largest distance to the curve, call it a breakpoint, generate two line segments—from start to breakpoint and from breakpoint to endpoint—and recursively apply the procedure to the two segments.

It is possible to use higher degree polynomials, such as degree 2. The approximations are then in terms of conic sections. The advantage is that fewer segments are needed because the segments are more descriptive. These, however, also have undesirable properties; A straight line might be well approximated by a set of curves, the representation might be unstable when the data is noisy, and the type of curve might change (a hyperbolic segment instead of an elliptic one). A better approach is to use B-splines. These are piecewise polynomial curves defined by a guiding polygon, no matter what the chosen degree of the spline is. They combine the expressive power of polynomials with the simplicity of polygons, which leads to efficient representation and compression. The mathematics behind B-splines, as well as the different variety of splines, is beyond the scope of this book.

3 3D GRAPHICS OBJECTS

Graphics are vector representations that can be created using a variety of software, which start from basic shapes and apply specific geometry operations depending on the representation. Additionally, detailed 3D representations can also be obtained by range scanning systems, which create sampled 3D points of an object's surface. The points are then assembled together into a 3D representation. Irrespective of how 3D models are computed, the resulting data is expensive to store, transmit, and render, which necessitates compression. In this chapter, we deal with 3D graphical objects (because 2D can be considered as a degenerate case of 3D), and furthermore, we assume that these 3D objects are represented by triangular meshes. Graphics can be represented in a variety of ways, such as polygonal, splines, constructive solid geometry, and so on. The following sections describe some of the popularly used representations and explain the choice of using triangular meshes. 3D objects are vector quantities. Vector representations were

introduced in Chapter 2. That has been extended here to show how 3D polygonal information is represented using vectors.

3.1 Polygonal Descriptions

3D objects are commonly represented by polygonal meshes. A polygonal mesh object is made of polygons, each of which, in turn, is specified by interconnections, called edges, among vertices. Each vertex is represented by coordinates in three dimensions. These are normally expressed as follows:

- An array of x, y, z positions for the vertices $v_1, v_2, v_3 \ldots v_n$.
- An array of index doubles for the edges $e_1, e_2, e_3 \ldots e_m$. Each edge e_i is expressed as an index double $<k,l>$, which implies an edge connection between vertices v_k and v_l.
- Although the preceding two points are self-sufficient to describe a polygonal mesh, often the mesh is further represented as faces by combining edges together that form a face. Each face f_i is represented as an n-tuple $<i_1, i_2 \ldots i_k>$ implying that the face is formed by connecting vertices $v_{i_1}, v_{i_2} \ldots v_{i_k}$. All faces are thus represented as an array of such n-tuples.
- Additionally, each vertex may have properties needed for its representation, display, or rendering—for example, a vertex normal specified by three coordinates $<n_x\ n_y\ n_z>$, a texture coordinate $<u, v>$, and a color per vertex coordinate $<c_x\ c_y\ c_z>$.

The examples in Figure 10-3 show primitives that build up into a mesh, where the mesh is represented by triangles (three-sided polygons), quads (four-sided polygons), and sometimes polygons of higher order. However, most applications that involve

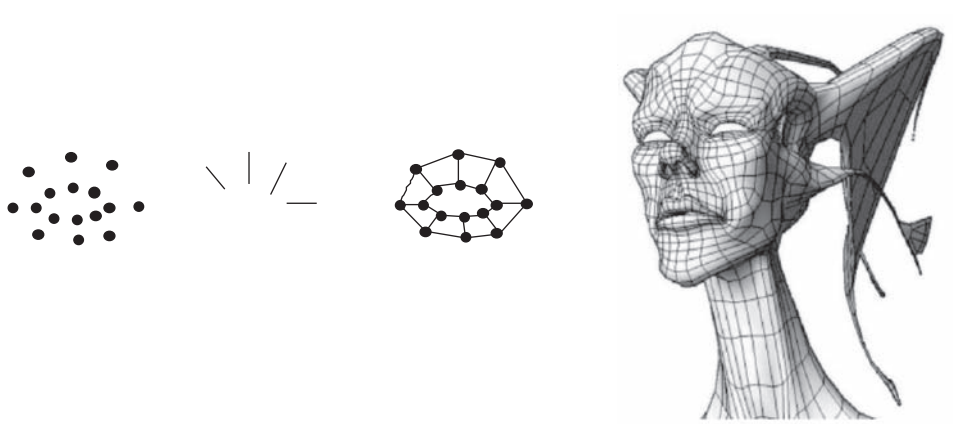


Figure 10-3 Polygonal description—points, edges, polygons, and an object. Traditionally, objects are composed of polygons consisting of triangles and quadrilaterals.

rendering, visualization, and manipulation of meshes often do so efficiently by reducing the representation into a triangular mesh. A triangular mesh consists of points, edges, and faces, where each face is represented only as a triangle.

3.2 Patch-Based Descriptions

Rather than using explicit *x, y, z* points to approximate a surface, smooth primitives tend to approximate a surface using mathematical surface descriptions—examples of these include Bézier patches and NURBs (nonuniform rational B-splines). One primitive gives a good local approximation of a surface and is normally not enough to describe a complex and articulated surface such as a face. Figure 10-4 shows examples of such patch-based descriptions.

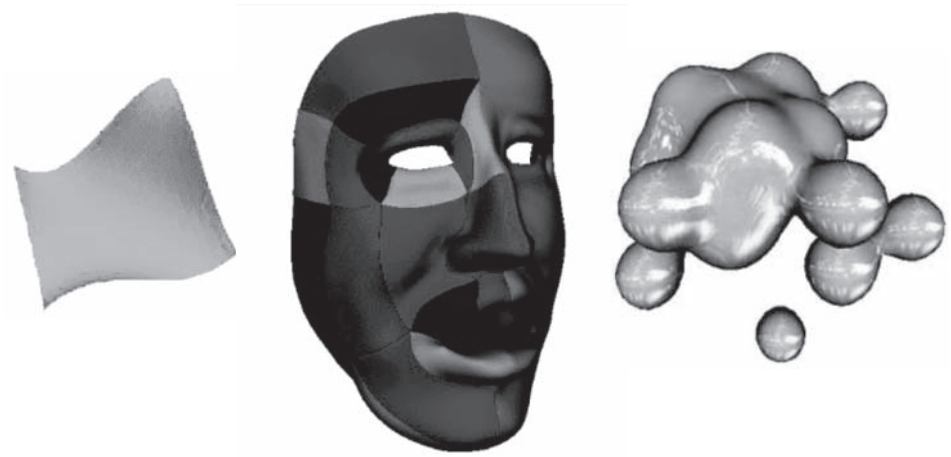


Figure 10-4 Nonsmooth primitive used to describe 3D objects—a surface patch, a group of patches forming a face, and metaballs

3.3 Constructive Solid Geometry

Constructive solid geometry (CSG) representations are used for modeling solids and are commonly used in computer-aided design applications. The modeling techniques using this representation are termed as *solid* modeling or *volume* modeling, when compared with the representations in section 3.1 and 3.2, which are termed as *surface* modeling. CSG representations start with simple parametrically defined primitive solids, such as cubes, spheres, cylinders, prisms, pyramids, and so on, and create complex surfaces by combining them using Boolean operators. These operations are unions, intersections, and differences.

CSG representations are popular in cases where mathematical accuracy cannot be compromised and in manufacturing where a modeler can use a set of simple objects to create complicated looking objects. Figure 10-5 shows an example of a CSG representation. However, CSG representations are not very popular in mainstream entertainment applications where the objects need to under deformations for animation.

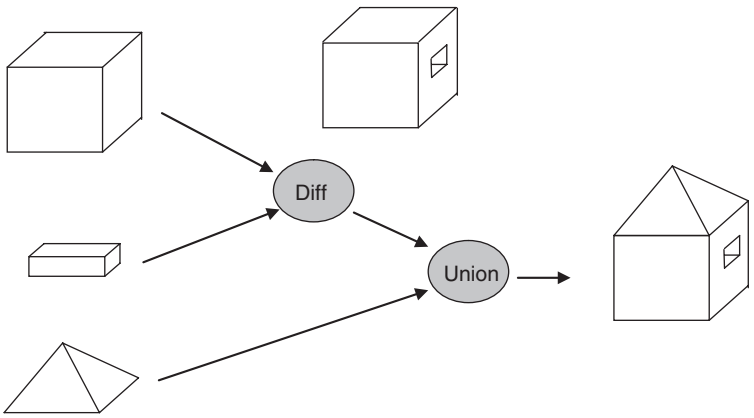


Figure 10-5 Constructive solid geometry example. The cube, the rectangular prism, and the pyramid are combined using different Boolean operations to create a fairly complex object in two steps.

4 GRAPHICS COMPRESSION IN RELATION TO OTHER MEDIA COMPRESSION

Graphics compression is a relatively new field compared with the other media types, such as image, video, and audio compression. Less attention has been devoted to 3D object and shape compression from research, industry, and standardization efforts, until recently when usage and complexities of 3D objects has grown in a variety of industries. For the purpose of discussion in this chapter, we limit our representation of 3D objects to polygons, as mentioned previously. Furthermore, because all standard graphics processing is defined on triangulated meshes, we only deal with triangular meshes in the remaining discussion.

Having understood the representation of triangular meshes mentioned in Section 3, an interesting consideration, then, is its evaluation of information content. To compress information in 3D meshes, it is important to recognize what constitutes information in the case of a mesh and, thereby, also understand what the redundancy here is. In 2D images, the redundancy lies in the similarity of local areas, and this manifests itself in generally low-frequency dominance for that area. Thus, compression can be achieved by approximating higher-order frequency coefficients with fewer bits in comparison with lower-order coefficients. In video, the redundancy occurs not only spatially in each frame, but also temporally from frame to frame. Motion compensation here helps to predict interframe motion that further helps approximate the redundancy. In 3D meshes, the information content lies in

- The position of the vertices
- The interconnectivity of the vertices to form triangles

The redundancy here can be categorized as representational redundancy and surface redundancy. Representational redundancy is inherent in how the interconnectivity is

specified. Normally, connectivity is explicitly specified using triangles where each triangle is represented by three indices into the vertex array. However, an edge is incident on two faces and, thus, gets specified twice. Moreover, a vertex is incident on a number of triangles, and each triangle thus repeats the vertex index. Such explicit connectivity representations need to be reduced. The surface redundancy occurs in the large number of vertex samples approximating a surface, whereas the surface might only need a few, as shown in Figure 10-6.

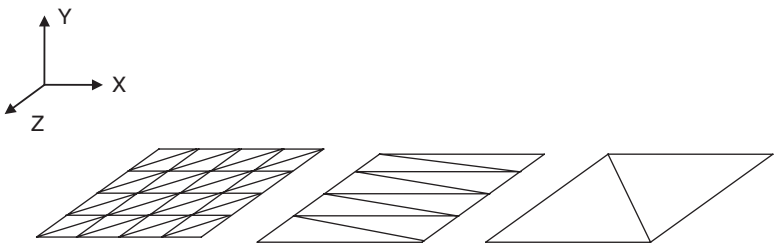


Figure 10-6 Various ways of triangulating a planar surface. The plane has the same geometry in all figures irrespective of the number of points, edges, and triangles used to represent it. In 3D, you want to represent the object with as few vertex samples removing the redundancy.

Recent methods in 3D compression can be divided into lossy and lossless techniques just like the other media types. Precision is the main requirement for CAD/CAM applications, which warrant lossless mesh compression, whereas 3D meshes used in the visualization process can tolerate precision loss. Additionally, each technique, whether lossy or lossless, can be further divided into four categories depending of the information that is compressed:

- *Direct compression of properties*—This technique is a lossy technique where data corresponding to vertices and other surface properties (normals, texture coordinates, vertex colors, and so on) are directly quantized. These techniques are often suboptimal, as they are not making use of any 3D coherency in the data set.
- *Connectivity encoding*—This technique mainly aims to reduce the connectivity information by exploiting coherency in the connectivity information. One of the main connectivity-preserving algorithms is topological surgery, which is explained in Section 5.
- *Polyhedral simplification*—Here, compression is achieved by reducing the number of vertices, edges, and triangles in the mesh, while at the same time maintaining the overall geometric shape. Progressive meshes are one class of representations that belong to this class and are explained in Section 6.
- *Subband-based surface compression*—The subband-based compression technique compresses the 3D data by quantizing the frequency of components in different 3D bands. A sample algorithm is explained in Section 7.

Representative algorithms of each compression technique are described in the next sections. In all cases, we assume that the triangular mesh is the underlying representation as followed by most graphics algorithms. If a mesh is not triangular, it can easily be

preprocessed and transformed into a triangular mesh. Additionally, the compression algorithms described in the following sections could also be used on nontriangular meshes at the expense of more complex data structures and bookkeeping. There are also surface appearance attributes other than geometry that are associated with a mesh. These might be unrelated to the mesh, such as a surface material indicator used during the rendering and shading process, or directly related to the mesh geometry, such as vertex color (*r, g, b*), vertex normal (n_x, n_y, n_z), and texture coordinates (*u, v*). All these attributes can either be derived (vertex normals) from the triangular mesh, or given separately to compress.

5 MESH COMPRESSION USING CONNECTIVITY ENCODING

A triangular mesh is represented by a set of mesh points and their interconnectivity. The mesh points are represented as 3D (*x, y, z*) positions and the interconnectivity is specified by edges. In such representations, the information is contained in the positions and the interconnectivity. Positional information is represented by a fixed number of bits (float, double, int) for each *x, y, z* coordinate for each vertex. The number of bits used depends on the desired precision and tolerable quantization error. The connectivity information in a triangular mesh is represented by three indices per triangle and can be substantial depending on the number of triangles in the mesh. Connectivity-encoding techniques aim mainly to reduce the redundancy in connectivity information. Before delving into connectivity information analysis, it should be mentioned that when dealing with good surface representations, there is no set formula to represent the number of triangles as a function of the number of vertices. On one side of the spectrum, we have very simple and well-formed structures such as a triangular grid, where the connectivity in a mesh is completely defined by the row and columns of the grid. A similar observation can be made for parametric surfaces like a sphere. Explicit representation of vertex interconnectivity by triangles is not necessary here and can be implicitly obtained as shown in Figure 10-7. Although such well-formed structures might be good for representing

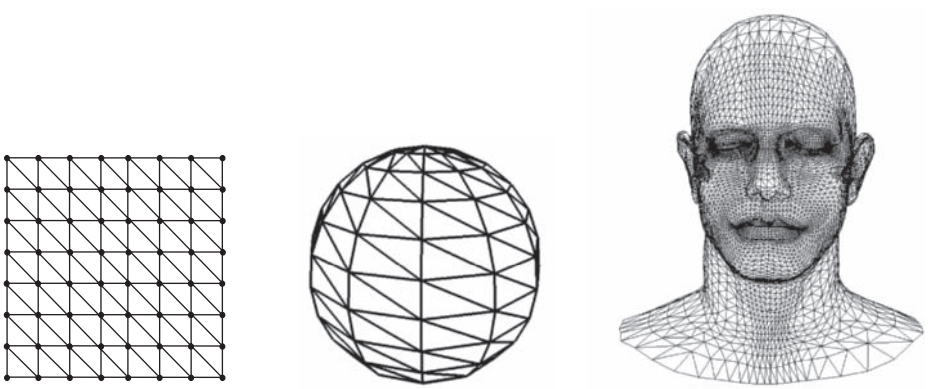


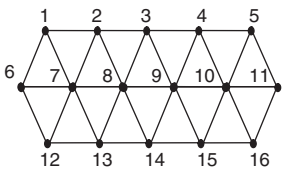
Figure 10-7 Example of triangular meshes. Left: A well-organized grid, where the edge interconnectivity can be derived from an ordered sequence of vertices and the number of rows and columns. Middle: A parametric surface where the mesh interconnectivity can also be derived. Right: An organic natural surface where triangles have to be explicitly specified.

terrains in 3D cartographic applications, 3D surface shapes in general cannot be approximated with such simplicity, which necessitates no explicit/implicit structure among the interconnections and each triangle needs to be represented by three vertex indices.

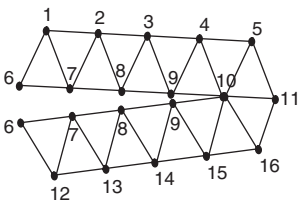
If there are n vertices in a mesh, each triangle is represented by three indices, where each index points to a vertex in the vertex array. The number of bits needed for each index is $log_2 n$. Statistically, there are twice as many triangular faces in a mesh as there are vertices. For n vertices and $2n$ triangles, the number of bits needed to specify the connectivity is $6n\ log_2 n$, or if averaged this amounts to $6log_2 n$ bits per vertex. Today it is not uncommon to have ten thousand vertices requiring roughly 84 bits per vertex, which can prove to be pretty substantial—840 Megabits in this case.

5.1 Triangle Runs

The redundancy in an explicit triangle-based connectivity representation can be greatly reduced by using triangle strips or triangle runs, where the connectivity can be implicitly defined for every triangle in the strip. This is so because, given a run, a vertex in a run needs to be combined with the preceding vertex and the following vertex in the run to specify a triangle. Triangle strips are commonly used in graphics standards like OpenGL and examples are shown in Figure 10-8. Generating triangle



Triangle mesh specified by 16 points and 18 triangles. Each triangle is a triplet of indices (1,6,7) (7,1,2) (2,7,8) (8,2,3) …(13,8,7) (7,12,13) (12,7,6)—a total of 54 indices.



Same mesh is cut open and approximated by a triangular strip with indices 6,1,7,2,8,3,9,4,10,5, 11,16,10,15,9,14,8,13,7,12,6—a total of 21 indices. The triangles are implicitly defined as a moving window of three indices—6,1,7 – 1,7,2 – 7,2,8.

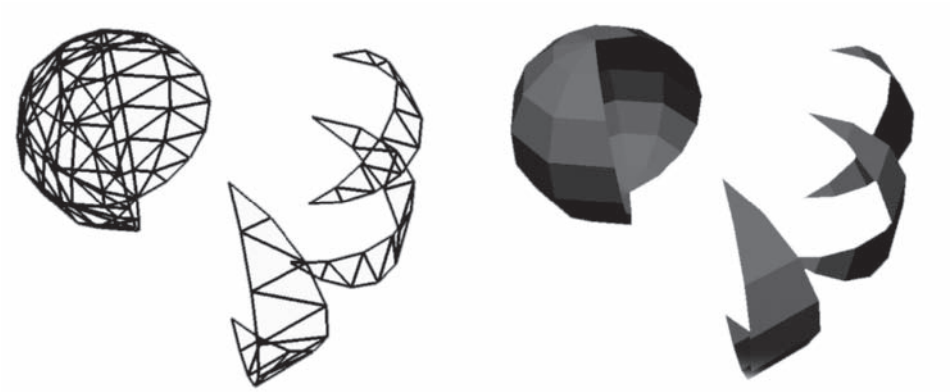


Figure 10-8 Triangle strips. Example of a sphere being cut into triangle strips. Each strip can now be described as a zigzag sequence of indices, wherein the triangle information is implicit rather than explicit specifying three indices per triangle.

strips is not a computationally easy process, but does make sense if you can build long strips, which reduces the explicit description of triangles from three indices per triangle to a list of vertices. Furthermore, triangle strip generation creates an order in which vertices need to be traversed. This traversal order can be used for further differential coding of vertex positions.

The objective of compressing connectivity in a mesh then translates to finding out all the triangular runs that the mesh can be decomposed into and then encoding these runs efficiently. Topological surgery is one method that cuts a mesh into runs and is described next. Ideally, one long run should provide the best compression; however, practically, the decomposition process into runs yields a graph of connected runs, which needs to be encoded. Finding the most efficient set of connected runs is nontrivial, and heuristics are often used to guide the process of computing runs.

5.2 Topological Surgery (TS) Compression Algorithm

Topological surgery was first proposed by Gabriel Taubin and Jared Rossignac, and is now part of the MPEG-4 standard that deals with 3D compression. It starts with a triangular mesh consisting of an array of vertex positions (x, y, z) and connectivity information per triangle and proceeds as follows:

- *Construct a vertex-spanning tree*—The original mesh can be viewed as a graph of nodes (vertices) and edges. A spanning tree created on this graph connects all the vertices with no cycles. The mesh can then be cut open along the spanning tree edges. This produces a graph of triangle runs. In this graph, which is also know as the triangle tree, each triangle can be thought of as a vertex in the graph that are connected by edges. Both the vertex spanning tree and the triangle tree needs to be encoded as explained in the steps that follow. The nonspanning tree edges form the interconnected edges in each run.
- *Encode the vertex-spanning tree*—The spanning tree, which shows how to cut the mesh, needs to be encoded. This encoded information is needed to perform the reverse process of reconstructing the mesh by the decoder. Encoding the spanning tree entails encoding a list of vertex indices that describe the spanning tree.
- *Compress vertex positions*—Originally, we stored the vertex positions as an array of (x, y, z) values. With the vertex-spanning tree created, there is now a traversal order for the vertices. Based on this traversal order, the vertex position of a future vertex can be predicted from the vertex coordinates of the current and past vertices (just like DPCM for audio signals). Only the delta values dx, dy, dz need to be stored now, thus lowering the entropy of the vertex array. These prediction errors, ordered according to the vertex tree traversal, could then be entropy coded using Huffman or arithmetic coding, as in the JPEG/MPEG image/video standards.
- *Encode the triangle tree*—A tree of triangle runs is created when the mesh is cut open (metaphorically) along the spanning tree edges. This tree can also be viewed as a graph of triangles, whose nodes are triangles and whose edges correspond to the marching edges in a triangle run. The nodes can be titled as *branching* nodes, *run* nodes, and *leaf* nodes. *Run* nodes and *leaf* nodes form a triangle run while branching nodes connect runs. Thus the tree structure is binary.

- *Compute and compress the marching pattern*—This part is a by-product of encoding the triangle tree. The runs on triangles can be efficient described now by listing only the vertex indices in a marching pattern until the run ends, either in a *leaf* node or another *branching* node.

The mesh reconstruction algorithm in the decoder works in the reverse way. Figure 10-9 illustrates an example of how the compression works.

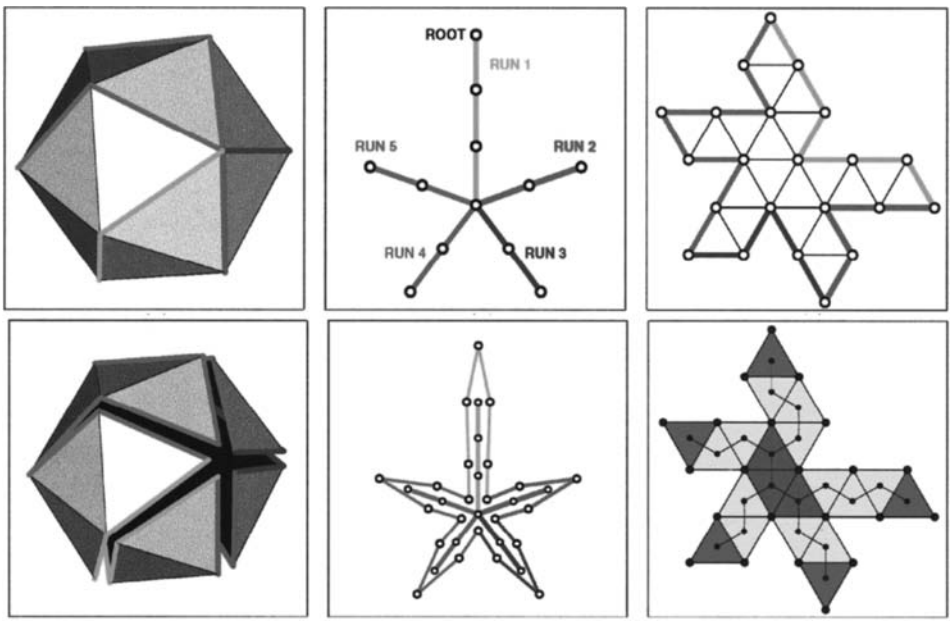


Figure 10-9 Topological surgery example (images courtesy of Gabriel Taubin). The top left figure shows a polyhedral object with a spanning tree (top middle) overlaid on the vertices and edges. This spanning tree can be used to open out the polyhedron as shown in the bottom left. This results in a flatten mesh (top right) which is then color coded to represent triangle runs (bottom right). The yellow color implies a triangle in a run, the red color indicates that the triangle is a leaf node and hence the end of the run. The blue color is used to represent triangles which function as nodes connecting the runs. See the color insert in this textbook for a full-color version of this image.

5.3 Analysis of Topological Surgery

Topological surgery is a lossless compression technique where the amount of compression achieved can vary depending on how the initial mesh is cut, or in other words, how the vertex-spanning tree is constructed. During its construction, the nodes (mesh vertices) can be visited in a depth-first or breadth-first traversal manner, or even a best-first heuristic combination of the two (like A* search). As far as encoding runs are concerned, you want to minimize the number of runs and increase the number of triangles per run. This reduces the branching information

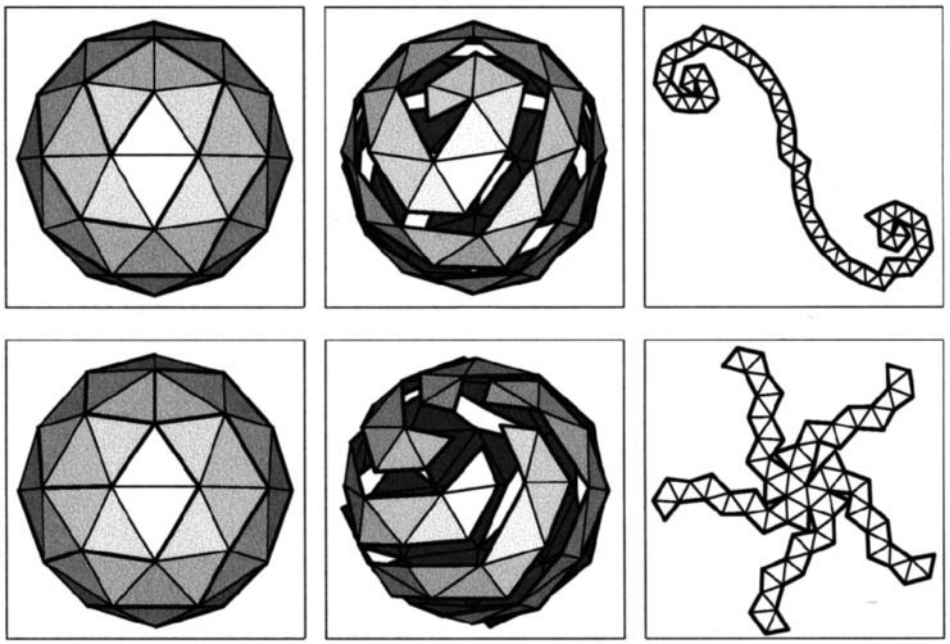


Figure 10-10 Different cutting strategies in topological surgery on the same object (images courtesy of Gabriel Taubin). The top row has created the vertex-spanning tree using a depth-first search strategy, whereas the bottom row has created it using a breadth-first search strategy.

needed during encoding the triangle tree and the vertex-spanning tree. An example of cutting strategies is shown in Figure 10-10, where the same object is cut using depth-first search and breadth-first search. You can see that the number and length of the runs differ in both cases and, in this specific case, the depth-first search traversal produces better results.

Another important decision to be made during the vertex-spanning tree construction, regardless of the graph traversal strategy, is the decision of which vertex to visit next. This choice of the next vertex to visit is not unique because a mesh vertex typically has a few neighbors. The problem can be better understood by analyzing the next step in the algorithm, where the vertex array is compressed by prediction-based techniques from the traversal orders output. Naturally, we want to traverse the mesh in a way that makes the predicted error minimal. A typical heuristic used here is the Euclidean distance to the next vertex from the current vertex. Given a choice of vertices, you would choose the closest vertex to the current vertex being expanded in the spanning tree. Statistically, 3D surfaces are locally very similar and, hence, the closest vertex would give the minimal prediction error. Figure 10-11 shows an example of a skull model where the choice of spanning-tree vertices was chosen depending on the Euclidean distance from the current vertex.

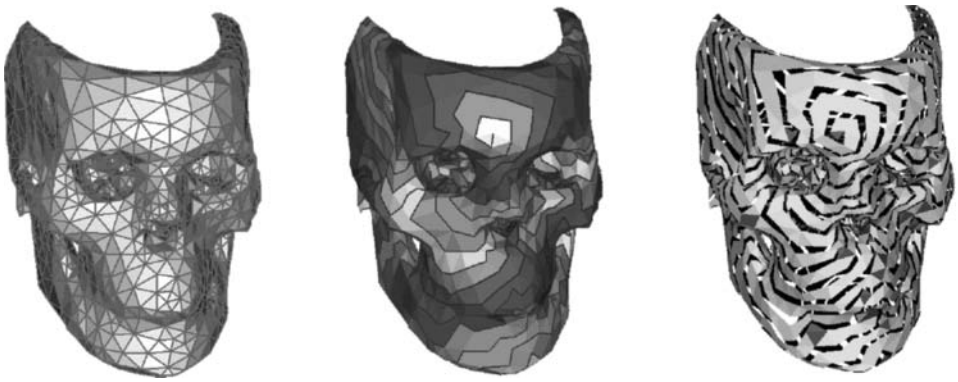


Figure 10-11 Topological surgery (images courtesy of Gabriel Taubin). Example on a skull mesh. The original mesh is shown on the left; the vertex spanning is shown in the center—the colors indicate the distance from the root node. The cut-open mesh is shown on the right. The right image shows color-coded run triangles (yellow), branching triangles (blue), and leaf triangles (red). See the color insert in this textbook for a full-color version of this image.

6 MESH COMPRESSION USING POLYHEDRAL SIMPLIFICATION

Polyhedral simplification techniques operate by iteratively removing vertices, edges, and faces from a mesh representation, such that the overall geometric perception of the object is still maintained. For example, if part of the object has a planar surface yet is populated by many vertices, some of these can be removed without affecting the geometric perception of the surface. However, if the surface does change, the goal of such techniques is to minimize the distortion caused by removing these vertices. Figure 10-12 shows a simple illustration of what this process entails.

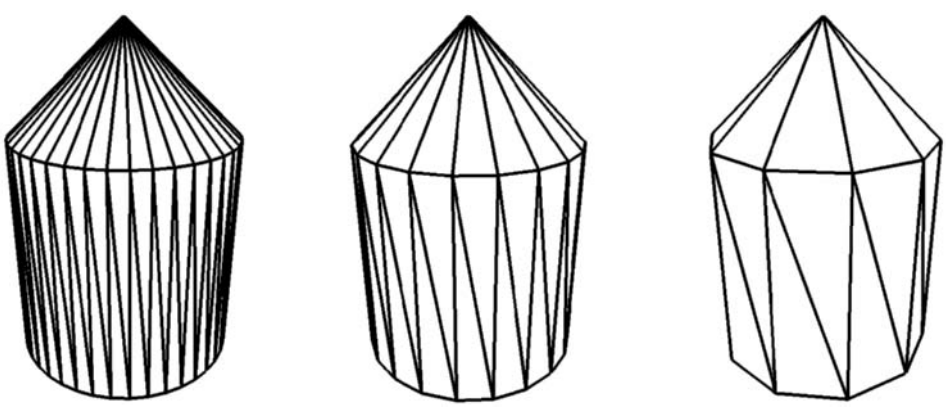


Figure 10-12 Example of a simplified polyhedron. The left figure shows a polyhedron that has been simplified into coarser representations. The objective of such simplification techniques is to minimize the distortion in perception of the 3D shape.

6.1 Progressive Meshes

Progressive meshes (PM) is a polyhedral simplification technique that was first introduced by Hughes Hoppe. In this form, the original detailed mesh is represented as $\hat{M}$ and is approximated as a coarser mesh M^0 together with a sequence of decimation records that indicate how to incrementally refine M^0 into M^1, M^2 . . . M^n, which is the original detailed mesh $\hat{M}$. At each stage, the decimation record stores the information associated with a *vertex split* operation. This is an elementary operation that adds an additional vertex to the mesh at each stage. The progressive mesh representation of $\hat{M}$ as a sequence of simple to complex meshes M^0, M^1, M^2 . . . M^n provides a way of continuously approximating the mesh at any desired level of detail or LOD.

As you can see from Figure 10-12, the coarser meshes are obtained by removing vertices and edges at each stage. The removal of these entities is obtained by using one of three fundamental transformations:

- *Edge collapse*—An edge collapse transformation $ecol(v_s, v_t)$ merges two adjacent vertices v_s and v_t into a single vertex v_s. As a result of this operation, the vertex v_t, the edge joining the two vertices (v_s, v_t), and the two adjacent faces $\{v_s, v_t, v_l\}$ and $\{v_s, v_t, v_r\}$ disappear in the process. The vertex v_s that still remains is given a new position.
- *Edge split*—An edge split operation introduces a new vertex on the edges and, correspondingly, also new faces.
- *Edge swap*—In an edge swap operation, two edges are swapped if it makes sense.

Overall, a single edge collapse transformation suffices to effectively simplify the mesh and is illustrated in Figure 10-13. The sequence of edge collapse transformations results in an initial mesh $\hat{M} = M^n$ being transformed into a simple mesh M^0.

$$\hat{M} = M^n \xrightarrow{Ecol_{n-1}} M^{n-1} \xrightarrow{Ecol_{n-2}} \cdots \xrightarrow{Ecol_1} M^1 \xrightarrow{Ecol_0} M^0$$

The vertex split transformation, which entails splitting a vertex into two vertices, thereby introducing another edge and two faces, is complementary to the edge collapse operation. The edge collapse and vertex split operations are, thus, invertible, resulting in the recovery of $\hat{M}$ from M^0 through a sequence of vertex splits, as shown in the following description.

$$M^0 \xrightarrow{Vsplit_0} M^1 \xrightarrow{Vsplit_1} \cdots \xrightarrow{Vsplit_{n-2}} M^{n-2} \xrightarrow{Vsplit_{n-1}} M^n = \hat{M}$$

As shown in Figure 10-13 (upper left), the vertex split transformation $vsplit(v_s)$ adds a new vertex v_t near vertex v_s and two new faces $\{v_s, v_t, v_{l1}\}$ and $\{v_s, v_t, v_{r1}\}$. This transformation also updates information on the mesh in the neighborhood and includes the new positions of vertices v_s and v_t, which can be represented as delta vector differences (dx_s, dy_s, dz_s) and (dx_t, dy_t, dz_t) from v_s. The faces and edges can be generated automatically given the current neighborhood state; however, the split

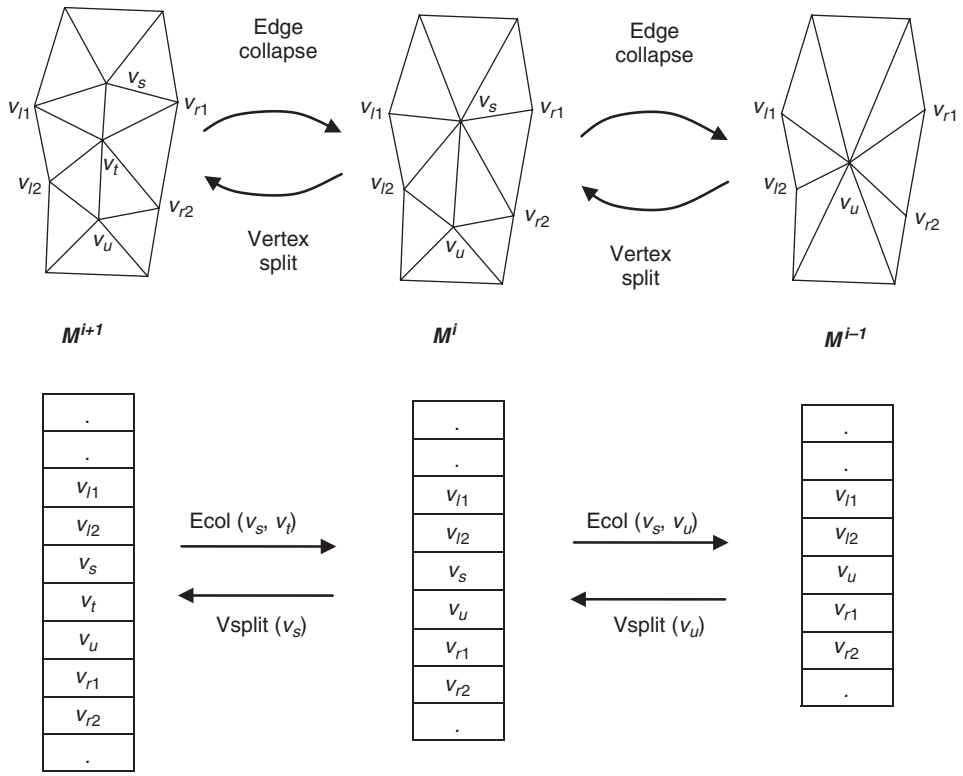
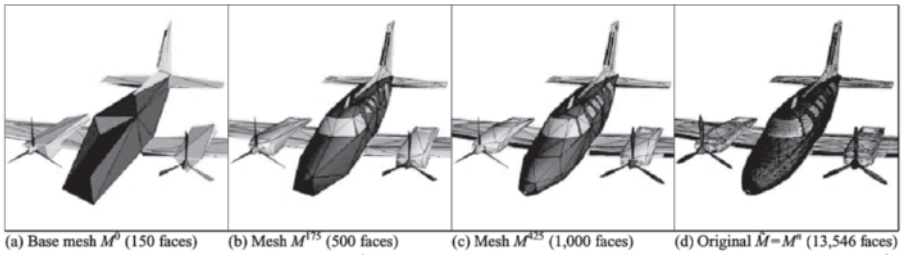


Figure 10-13 Transformations in mesh simplification in progressive meshes—edge collapse and the complementary operation vertex split

operation of v_s needs to contain pointers to the vertices (s, l_1, r_1). Thus, the amount of information that needs to be stored for a split is two difference vectors and three vertex indices. Note that if the surface-based attributes are represented, they also need to be updated.

Although progressive meshes start with $\hat{M}$ and create all intermediary representations down to M^0, compression can be achieved in a lossy and lossless manner. For lossy compression, the edge collapses can create smaller-sized meshes until the memory stamp of M^i is just under the required bandwidth requirement. For lossless compression, we need to encode the lowest level M^0 along with the continuous set of vertex split operations. Each vertex split operation, which consists of two vectors (dx, dy, dz) and a triplet index (s, l, r) can be efficiently encoded in two ways. A proper choice of the vertices chosen during the *ecollpse*(v_s, v_t) stage, which minimizes the local shape distortion, can ensure that the (dx, dy, dz) vectors have very small ranges and, hence, have less entropy, resulting in fewer representational bits. The triplet index, which tells you the vertex to split along with the left and right neighborhood vertices to create new faces, can be accomplished by merely encoding one vertex index suggesting the split vertex and a few bits (5–6 bits) for choosing one two-neighbor combination out of all possible two-neighbor permutations



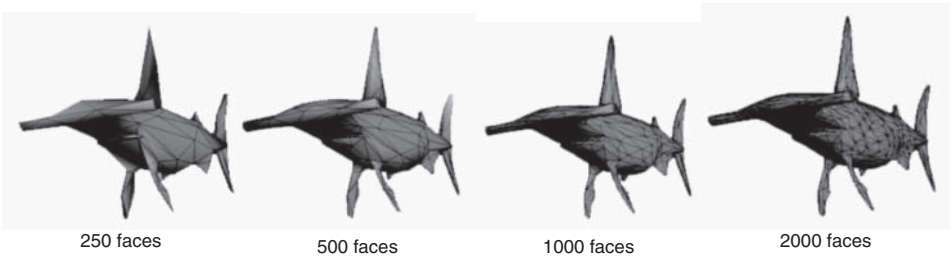


Figure 10-14 Progressive mesh approximations (images courtesy of Hughes Hoppe). The images show a sequence of approximations of stages obtained during progressive mesh generation. The left figure in each row shows the lowest resolution which increases as we move to the right.

for vertices around v_s. Figure 10-14 shows an example of meshes generated by progressive computations.

6.2 Analysis of Progressive Meshes

Progressive mesh representations simplify an original mesh by iteratively combining two vertices, which is termed as an edge collapse operation. The vertices chosen for the edge collapse change the underlying mesh and introduce an error. This error is reversible by applying a vertex split using the information cached by the edge collapse operation. However, the error does introduce a distortion in the perception of the shaded mesh object at each iteration. Because the choice of these two vertices is not unique, the objective of each iteration(s) should be to choose two vertices that result in a minimum error at each iteration over all possible iterations. This iterative evaluation can prove to be an expensive operation when the mesh sizes are large and also might lead to choosing vertices that might not end up being as important. For example, while viewing a progressive mesh from a viewpoint, vertices on faces facing away from the camera or vertices not in view neither influence rendering speeds nor give added details that are visible. In such cases, selectively refining vertices is helpful where vertices facing away from the camera or those that are projected out of the camera's screen space need not be considered and can be represented very coarsely. Even in the field of view, the distance of the surface from the camera can be used as a measure to control refinements—for example, the farther away the surface, the more coarsely it can be represented.

7 MULTIRESOLUTION TECHNIQUES—WAVELET-BASED ENCODING

This class of compression techniques works by considering the 3D model as a single surface that needs to be compressed globally rather than attempting local modifications for compression. The main idea is similar to the previous section in that attempts to approximate the surface by a coarse base representation are followed with refinements. However, the base representation and refinements are computed on the mesh as a whole in a frequency space producing detailed coefficients along the way at each stage. This is unlike progressive meshes and other remeshing techniques where local reasoning is used. In wavelet-based image compression, the image is approximated by various frequency bands, where each band gives added detail. In a similar manner, we can split a high-resolution triangular mesh into a low-resolution part and successive detail parts. For example, in Figure 10-15, the object shown can be approximated at various levels of detail.

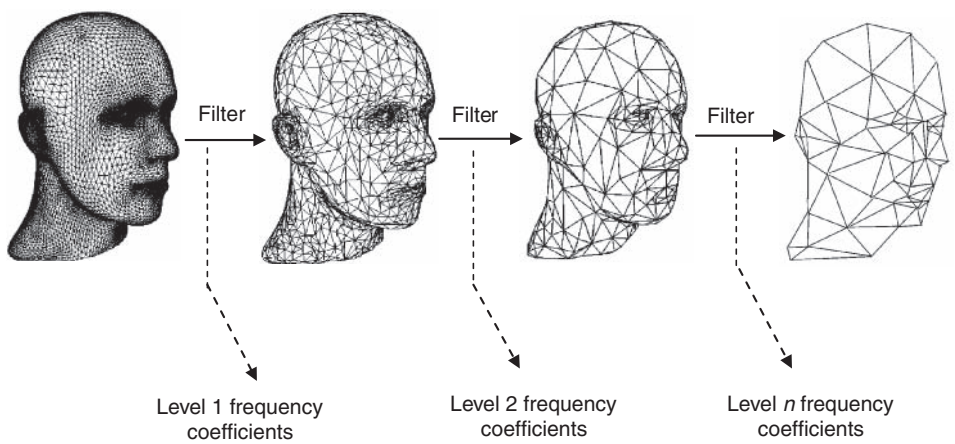


Figure 10-15 Multiresolution surface approximation example. The high-resolution surface shown on the left can be filtered successively using 3D frequency transforms to approximate lower-resolution meshes along with frequency coefficients. Quantization of the frequency coefficients would lead to compression.

The surface here is represented as a sequence of approximations at different resolution levels. The change at each level is transformed by a wavelet transform into a set of wavelet coefficients that express this difference. Compressing the mesh then reduces to storing these wavelet coefficients in a lossless or lossy manner. For a lossy compression, we need to appropriately quantize the frequency coefficients.

8 PROGRESSIVE ENCODING AND LEVEL OF DETAIL

From a pure information theory perspective, the most achievable compression always leads to the least space on disc and also to the quickest transmission. However, from a user's perspective, this might not be necessarily desired. Frequently the end user is

more concerned with getting a good perceptual quality mesh in a reasonable time instead of getting the most optimal mesh. This is true of any media type—of images, video, and audio. Progressive transmission, where data is sent in a coarse-to-fine way by reorganizing the encoding and bit stream, is one way to address this problem that delivers perceptually valid information to the user.

3D data is used in a number of different applications—from entertainment, to medical, to surveillance, to military, to cartography, and so on—where much of the interactivity and visualization speeds need to be real time or close to real time. The dense geometric representation necessitates compression, but for practical use the bit stream needs to be shuffled, sending a very coarse approximation first, followed by subsequent bits that allow addition of more details in a successive manner. Real-time applications that share data and render the dense mesh information cannot wait until all relevant compressed data is received, decompressed, and then rendered. They need to do so seamlessly on the fly from a coarse-to-fine manner at the needed frame rates, which is unlike traditional techniques, where all the data is available prior to rendering. The following list describes some examples where progressive encoding of meshes is necessary:

- *3D multiplayer games across a network*—The quick interactivity necessitates the use of lower-resolution models so that communication is faster. But as the 3D models and the 3D characters increase their resolution, better compression and smarter ways of distributing compressed data will be required to maintain the real-time rendering and interactivity.
- *Visualizing 3D data*—3D data is for scientific to commercial visualization. For example, Internet users now have the opportunity to visualize 3D models of products while shopping on the Web. Although all the compressed data delivery might not be instantaneous, it is more viable to have coarser representations delivered instantaneously, which users can view and interact with in a virtual setting.
- *Animating data*—This is a compression-related problem. Setting up animations is a time-consuming process and often the need to have dense mesh models to work with slows down the animation process and the creativity workflow of the animator. Animators prefer working with coarser models in a more interactive setup and have their animations gracefully transfer onto denser mesh geometry at render time.

Traditional encoding modes necessitate that the client receive all the data first, then decompress the data to be followed by rendering. Such techniques need to wait before all mesh data is received. The undesirable latency can be reduced by encoding and transmitting the data in a progressive manner where a computed coarse representation is entropy coded and sent first, followed by finer information that the decoder can add on to refine in mesh information, as shown in Figure 10-16. Some of the techniques discussed so far naturally lend themselves to progressive compression. For example, in progressive meshes, the compact M^0 mesh can be transmitted first followed by a stream of $vsplit_i$ information records. The decoder incrementally builds M^n from M^0 using the vertex split information, as illustrated in Figure 10-13. One drawback of this method is that, depending on how the vertex splits are ordered, the refinement at the decoder

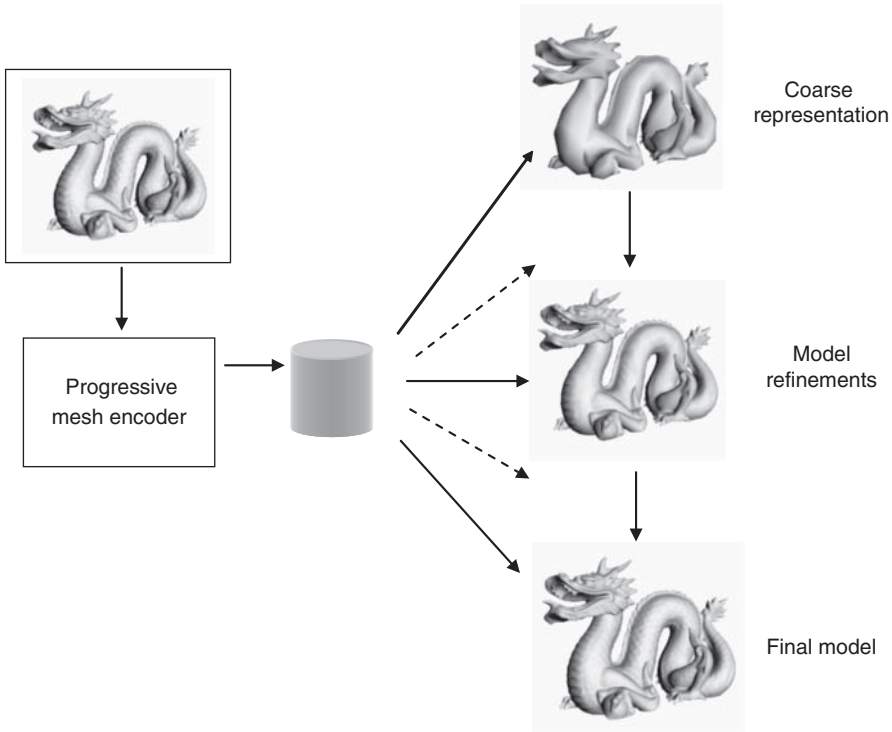


Figure 10-16 Progressive mesh encoding. The compressed bit stream is organized so that a coarse representation is encoded first, followed by refinement information, which increases the details on the decoded models.

might not seem spatially coherent and appear visually unaesthetic. The frequency space approximation approaches do not have these problems because refinements are added on a universal frequency level.

9 3D GRAPHICS COMPRESSION STANDARDS

Standards are established so that a compressed bit stream can be parsed and interpreted similarly on different platforms supported by different software. With regard to 3D graphics, compression formats were only recently introduced.

9.1 VRML

VRML stands for Virtual Reality Modeling Language: It was the first attempt to standardize specification of static and dynamic 3D scenes. Although it can perform any compression on the geometry, it was the first 3D visual extension to the World Wide Web and it is a precursor to all the compression formats discussed in the next few subsections. VRML descriptions have syntax and tags similar to HTML, which can be

interpreted and visually displayed in a standardized VRML browser. VRML was proposed in the mid-1990s and has already gone through two definition versions, VRML 1.0 and VRML 2.0. Although no longer popular, the VRML format has paved the way for 3D interactive standards such as X3D and MPEG-4.

9.2 X3D

X3D or extensible 3D is a software standard aimed at formalizing the Web-based as well as broadcast-based interactive 3D content. It has been put together by the Web3D Consortium and sets forth an XML-based file format along with interfaces to create an open standard for real-time 3D communication in a broad range of applications areas—engineering, scientific visualizations, multimedia presentations, medical, entertainment, and education. It is the successor to VRML and has all the description and animation capabilities, but improves on it by providing advanced application programming interfaces and 3D encoding formats. It consists of two parts: Part 1 describes the architecture and base components, and Part 2 describes the scene access interface. All together, it aims at the following:

- *Description*—Describe 3D objects and worlds that include geometric mesh descriptions, parametric geometry, and other attributes that relate to the visualization/rendering of the mesh—such as surface material properties, shaders, textures, lighting models, and so on. It also does the same for 2D objects.
- *Animation and interaction*—Provide standard means of using timers, interpolators to create continuous animations, deformations, and humanoid animations, and have interactivity to control the animations. It also supports camera navigation and physical simulation.
- *Scripting and programming*—Provide the ability to dynamically change the scene via scripting languages.
- *Encoding and compression*—This is provided at two levels, first where the actual 3D content made up of meshes is compressed and, second, at the compression of the scene description where the nodes, transformations, and animations are compressed using information theoretic considerations.

In general, X3D provides a broad standard to describe 3D objects that together form a scene, animate them, and compress them for real-time streaming/distribution reasons. Along with 3D, it also contains specifications of how other media types, such as video and audio, can be universally described. The standardized bit stream generated is intended to be used for a variety of hardware devices as well as software applications. Figure 10-17 shows an example of an X3D description file. The scene tag in the XML description pertains to the 3D geometry and its surface and rendering properties. The 3D geometry is specified in the IndexedFaceSet node tag, which contains a section for the vertex array (*coordinate point*) and another for the connectivity information (*coordinate index*). The vertex array is specified as an x, y, z coordinate, whereas the connectivity information is specified by a sequence of indices separated by a '−1' delimiter for all the polygons/triangles used. For conciseness, only a few coordinates and interconnectivity information is shown.

```
<X3D version='3.1' profile='Immersive' xmlns:xsd=
'http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation=
'http://www.web3d.org/specifications/x3d-3.1.xsd'>
<head>
    <meta name='filename' content='example.x3d'/>
    <meta name='author' content='authorName'/>
    <meta name='created' content='March 20 2003'/>
    <meta name='keywords' content='X3D VRML binary compression'/>
    <meta name='url' content='file://C:/multimedia/3d/example.x3d'/>
</head>
<Scene>
    <Viewpoint position='0.0 0.0 1.0' description='camera view'/>
    <Background groundColor='0.5 0,5 0.5'/>
    <Transform scale='0.25 0.25 0.25'>
      <Shape>
        <Appearance>
          <Material diffuseColor='1.0 0 0'/>
        </Appearance>
        <IndexedFaceSet coordIndex='0 3 4 -1 2 42 1 -1 3 4 7 -1 2 4 11 -1 2 14
16 -1 5 11 71 -1 12 35 67 -1 45 23 7 -1 3 44 56 -1. . . .'>
        <Coordinate point='0.2138 0.2339 0.09065, 0.1078 0.2532 0.1026,'
0.2138 0.2339 0.09065, 0.1078 0.2532 0.1026, '0.2138 0.2339 0.09065, 0.1078 0.2532
0.1026,' 0.2138 0.2339 0.09065, 0.1078 0.2532 0.1026, '0.2138 0.2339 0.09065, 0.1078
0.2532 0.1026,' 0.2138 0.2339 0.09065, 0.1078 0.2532 0.1026, '0.2138 0.2339 0.09065,
0.1078 0.2532 0.1026, . . . .'/>
        </IndexedFaceSet>
      </Shape>
    </Transform>
  </Scene>
</X3D>
```

Figure 10-17 X3D description of a static 3D geometry set in XML format. The section included within the scene tag corresponds to the actual geometric content listing color assignments, appearance information, and the actual coordinates. The Coordinate point tag gives the x, y, z floating-point representations of vertices, whereas the IndexedFaceSet tag gives the interconnectivity between the vertices. The list shows three indices separated by a −1, suggesting triangle vertex indices pointing to coordinate point vertices.

```
<X3D version='3.1' profile='Immersive' xmlns:xsd=
'http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation=
'http://www.web3d.org/specifications/x3d-3.1.xsd '>
   <head>
      <meta name='filename' content='example.x3d'/>
      <meta name='author' content='authorName'/>
      <meta name='created' content='March 20 2003'/>
      <meta name='keywords' content='X3D VRML binary compression'/>
      <meta name='url' content='file://C:/multimedia/3d/example.x3d'/>
   </head>
   <! ---- compressed binary data ---->
</X3D>
```

Figure 10-18 A compressed representation of the X3D description in Figure 10-17. The compressed binary data corresponds to a stream of binary bits that are obtained by compressing the scene information tag.

9.3 MPEG-4

MPEG-4 deals with a variety of media objects in terms of description, interaction, and compression of each media type into elementary bit streams and the delivery of standardized streams to end terminals. The overall architecture is the subject matter of Chapter 14. One of the streams relates to graphical objects in terms of representation, interactivity setup, and compression. Most of the 3D scene description and interactivity setup is an extension of VRML and is similar to X3D; however, MPEG-4 also specifies a 3D mesh-coding bit syntax in its MPEG-4 Visual, Version 2. The coding process is based on an *IndexFaceSet* Node similar to the one described for X3D in Figure 10-17. The main components of this node are the connectivity and the geometry. The mesh compression can be done in a lossy and lossless manner using the steps outlined in the topological surgery algorithm discussed in Section 5.2, which performs a differential quantization of the geometry and preserves connectivity by transforming the mesh into triangle strips. The data that results is then entropy coded. Figure 10-19 shows the pipeline for a 3D mesh encoder in the MPEG-4 standard.

MPEG-4's mesh compression provides a compression ratio of 30:1 to 40:1 without noticeable visual degradation. The bit stream provides much functionality besides lossy and lossless compression as mentioned in the following list:

- *Progressive or increment rendering*—It is not necessary to wait until the complete bit stream is received by the decoder to decode and render. The progressive bit stream formatting capability allows the decoder to decode and render the data incrementally as the data arrives at the decoder, thus minimizing latency.
- *Support for nonmanifold object*—The 3D topological algorithm used in MPEG-4 compression only works on manifold objects that are orientable.

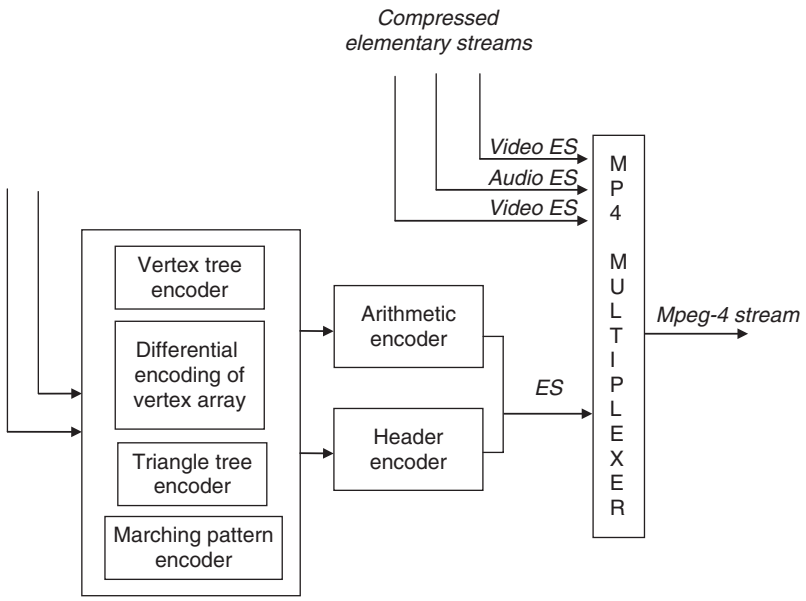


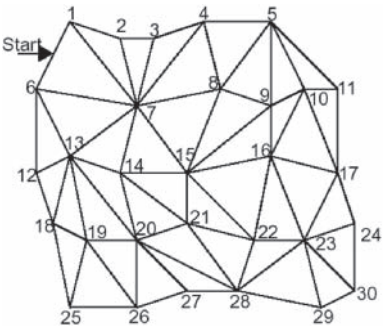
Figure 10-19 MPEG-4 3D mesh encoding. The mesh encoding process is based on the topological surgery algorithm discussed in Section 5.2. The resulting compressed bit stream, an elementary stream, is then multiplexed with other media elementary streams.

- *Error resiliency support*—In case of network errors or corruption during transmission, MPEG-4 scene decoders can still recover the 3D mesh because the encoding is performed on partitions that do not depend on others.

9.4 Java 3D

The Java 3D standard defines a programming API that makes it easy to specify a binary compressed geometry format. The compression API is based on the initial pioneering work of Michael Deering, which was a precursor to the topological surgery algorithm discussed in Section 5. First, the geometry to be compressed is converted into a generalized triangle mesh, where each triangle can be represented on average by 0.8 vertices instead of three vertices. This generalized technique addresses all kinds of triangle mesh representations efficiently, when the geometry does not come in the form of a perfectly regular rectangular mesh, which is often the case. Regular meshes are described well with triangle strips (see Section 5.1); however, irregular meshes can have many repetitions in triangular strips because vertices can be shared between varying numbers of triangles. A *mesh buffer* can be used to reference such repetitions in a triangle strip, which creates the generalized representation shown in Figure 10-20.

Once connectivity is defined, the data for each vertex is then converted to the most efficient representation format for its type, and quantized to as few bits as possible for the desired compression. The quantization is followed by



Generalized Triangle Strip:
R6, O1, O7, O2, O3, M4, M8, O5, O9, O10, M11, M17, M16, M9, O15, O8, O7, M14, O13, M6, O12, M18, M19, M20, M14, O21, O15, O22, O16, O23, O17, O24, M30, M29, M28, M22, O21, M20, M27, O26, M19, O25, O18

Generalized Triangle Mesh:
R6p, O1, O7p, O2, O3, M4, M8p, O5, O9p, O10, M11, M17p, M16p, M-3, O15p, O-5, O6, M14p, O13p, M-9, O12, M18p, M19p, M20p, M-5, O21p, O-7, O22p, O-9, O23, O-10, O-7, M30, M29, M28, M-1, O-2, M-3, M27, O26, M-4, O25, O-5

Notation Key: R = Restart, O = Replace Oldest, M = Replace Trailing, "p" = Push into mesh buffer. All vertices are numbered, the "-" indicates mesh buffer reference where –1 is the most recent pushed vertex.

Figure 10-20 Example of a generalized triangle mesh (image courtesy of Michael Deering). The figure on the left shows a generalized mesh. The list of vertices on the right (top) shows how the mesh could be coded as a triangle strip; note that the list of vertices needs explicit references including for those that are repeated. The right (bottom) list shows how the mesh could be coded as a generalized triangle mesh, where old vertices are pushed into a mesh buffer and referenced from the buffer. The buffer size is finite and, hence, the number of bits required to reference a vertex in the buffer is less than the number of bits required to reference it directly.

computing differences between successive vertices in an orderly manner. The differences reduce the entropy in the mesh information and are further compressed using a modified Huffman code. Color information is also compressed in a similar manner to positional vertex information. But normal information is compressed parametrically. Finally, the variable-length codes are combined using Java 3D's eight geometry commands, described in the following list, into a final compressed block.

- *vertex*—This command specifies the Huffman compressed delta encoded 3D position, two triangle strip replacement bits, a mesh buffer push bit, and might specify a normal and/or color coordinates, which is controlled by additional two state bits (bnv and bcv).
- *normal, color*—These are two independent commands to specify normal and color information, which is also Huffman coded using delta values.
- *setState*—This command is used to update the five state bits for the vertex triangle runs, color and normal.
- *meshBufferReference*—This command allows any of the 16 recently used vertices to be referenced as a next vertex.
- *setTable*—This command sets entries in the three entropy coded Huffman tables for position, normal, and color.
- *passThrough*—This command allows other unrelated data to be embedded in the compression stream.

- *NOP*—This command signifies no operation, but is mostly used to pad bits to be aligned to 32-bit boundaries, which is needed during run-time dispatching and fetching.

10 EXERCISES

1. [**R03**] Meshes can be compressed in a variety of ways.
   - Where is the information content in a mesh, and where is the redundancy in this information? How is this different compared with the other media types—images, video, and audio?
   - There are different algorithms mentioned in the text—topological surgery, progressive meshes, wavelet-based compression. Explain which aspect of redundancy that each reduces.
   - In the text, we have mentioned that mesh vertices also have other attributes, such as *vertex color, vertex normals, texture coordinates*, and so on. What are these properties and how are they used in graphics content creation?
   - If these properties are needed as an integral part of the mesh representation, they need to be compressed. Explain how the techniques discussed in the text can be extended to compress these parameters, mentioning in each case how and why a simple extrapolation of the geometric compression algorithm would work, and if not, discuss your other alternatives.
2. [**R04**] All the 3D compression algorithms discussed in the text start with polyhedral meshes that are triangulated. Although the text mentions that triangulated meshes are commonly used representations, it does not mention how to derive triangular meshes from a polyhedral mesh where the polygons are not triangles.
   - Why have triangulated meshes become so popular?
   - How do you convert an n-sided polygon into triangles? This is easy if the vertices are planar, but not so easy if the vertices are not coplanar. Derive methods in both cases.
   - In a triangulated mesh, if each vertex coordinate is represented as a *double* data type, and there are n vertices, what is the number of bits needed to store the mesh in terms of n? Assume that for n vertices, there are $2n$ triangular faces on average.
3. [**R03**] The text mentions a variety of compression algorithms. For the following application scenarios, mention which of these methods you would choose, giving reasons for your choice.
   - In production, you want to work with a low-resolution mesh to make your animations quickly, and then transfer animations to a high-resolution mesh.
   - You want to send a 3D object mesh across a network to a lithography machine, which is used to make an exact replica.

- To reduce rendering complexities in a 3D game, you want to insert a compressed representation of a mesh when not much detail can be seen from the camera's viewpoint.

4. [**R05**] The progressive meshes are polyhedral, simplification-based compression techniques, where a complex mesh is reduced to simplified representations at each stage by edge collapse operations until reaching a coarse-level mesh.
   - Are the set of representations generated unique? Explain.
   - Why do vertex splits require less information to encode?
   - What happens when a vsplit (vs) generates vt and two faces and the edge {vs, vt} is actually a boundary edge $\{v_s, v_t, v_{l1}\}$ and $\{v_s, v_t, v_{r1}\}$?
   - How many bits are needed on average to store one vertex split operation?
   - One application of this technique is geomorphs (name invented by the author), where coarser representations can be morphed into finer representations in a continuous manner. Can you suggest applications where this is useful?
5. [**R04**] The topological surgery algorithm aims to create runs.
   - The text mentions that for good compression, you want long runs with less number of runs in the triangle tree. Argue that this is true in general—that longer runs give you the best compression.
   - For constructing such long runs, which methodology would you use during the vertex-spanning tree construction—DFS or BFS? Why?
   - Either way, you have a choice of vertices to visit at each stage during the construction of the vertex-spanning tree. The text describes a distance-based heuristic. When would this be useful?
   - Can you think of a better strategy to visit vertices during the spanning tree build? Explain why it would perform better.
6. [**R07**] Although the text deals with 3D triangulated meshes, 2D graphics content uses 2D meshes.
   - Can the different algorithms discussed in the text be reduced to compress 2D meshes? How?
   - Whether 2D or 3D, normally meshes are built to look regular (where all the edges of the triangles are almost equal). Part of the reason is that regularity is desired when meshes deform during animations and are rendered. But in terms of compression, does regularity help, not help, or does not matter?
   - Given an input mesh, if the objective of a preprocessing software program is to make it regular prior to compression, what algorithm can you think of that will achieve this?
7. [**R07**] Another important feature about browsing any kind of information is being able to randomly access a region of interest or ROI like in the JPEG2000 standard. In case of large and dense 3D meshes, you want to see a very coarse representation and zoom into an area of a mesh where denser representations of the mesh are shown for the area to be focused.
   - Mention a few applications and use scenarios where the region of interest is useful.

- Most of the algorithms discussed in the text do not talk about random access of the compressed bit stream. How can they be extended to support random access?

8. [**R06**] A different, though related topic that is used to manage data in a graphics production pipeline is the usage of subdivision techniques based on the pioneering work of Catmull and Clark for surfaces. Here, a coarse-level mesh is defined or created and detailed meshes are approximated by rules of subdivision. Go the Web and read about subdivision techniques to answer these questions:
   - Subdivision rules are used to approximate 2D (Bézier curves, splines) and 3D data. Find out how the rules work for 2D and 3D data.
   - How do subdivision schemes relate to compression? Given a mesh, can you perform subdivision to achieve compression?
   - In the production scenario for creating 3D content, what does subdivision give you that some of the traditional compression schemes discussed don't?

9. [**R08**] Most of the techniques discussed revolve around compression of a static mesh. In animations, meshes undergo rigid and nonrigid transformations and deformations that cause the vertices to changes positions. In this question, you are asked how you might efficiently extend some of the algorithms discussed for static meshes to animating meshes assuming that interconnectivity is maintained. You have n frames of a mesh where the mesh is deforming in each frame—the vertex positions change, but the connectivity information remains the same. The objective is to efficiently compress the animation of n frames. *You must do better than compressing each frame statically*.
   - You could use topological surgery to compress the first mesh. How would you extend it to compress the following meshes? Outline which parts of the algorithm need to be computed at each frame and which parts don't. How can you store the recomputed parts efficiently?
   - In progressive meshes, you compress the mesh by edge collapse operations. Let's say you do this on the first frame and bring the mesh vertices down in count. You also do this for each of the following frames so that now you have low-count meshes for each frame having the same number of vertices. What are the advantages and disadvantages of doing so?
   - Suggest a modified scheme based on the discussed algorithms (or a new one) that efficiently compress an animation—make use of the temporal redundancy concepts by creating the following at each frame: I_{mesh}, P_{mesh}, and B_{mesh}.

10. [**R08**] The techniques discussed in this chapter are not efficient when it comes to compressing really large mesh data sets. One of the primary reasons is that the mesh needs to be read into memory prior to applying any compression algorithm on it. This is prohibitively expensive. For example, given accurate laser-scanning technologies, it is not impossible to create data sets that are 6–10 Gbytes. It would be more efficient to compress data as it is obtained from the scanning.

- First discuss what problems can occur because of writing such large amounts of data obtained by scanning.
- Suggest either how to modify the discussed techniques or suggest a new technique that could effectively deal with such data. Is your technique optimal in terms of the maximal compression achievable? Explain.

PROGRAMMING ASSIGNMENTS

11. [**R06**] One of the main needed computations of connectivity-preserving compression techniques is the creation of triangle strips. Topological surgery creates triangle strips by creating a vertex-spanning tree on the mesh, which can be created in a depth-first manner, breadth-first manner, or an intelligent search (like A*), that evaluates the next node in the tree based on a cost. In this assignment, you are asked to write a program to see how effectively such spanning techniques on a mesh might help create triangle strips that yield better compression ratios. We have provided triangular meshes on the accompanying CD. To that end, complete the following:
    - Write a program that takes a mesh and a start node and creates a vertex-spanning tree on it using DFS. Given a node, there is a possibility of choosing a number of neighboring vertices to expand. What criterion would you implement to choose the next vertex in your DFS search? What are the resulting triangle strips?
    - Repeat the preceding exercise by performing a breadth-first search (BFS). Which ones give you longer triangular strips? Longer triangular strips result in fewer bits to code the triangle graph, but argue why/if this is not necessarily the case for the overall compression.
    - Given a mesh, a variety of spanning trees can be computed, each of which result in a different triangle graph, different entropies for delta vertex computations, and, ultimately, different compression ratios. If your objective was to get the best compression ratio possible at the cost of time, what criteria would you use to create the vertex-spanning tree? Implement this and compare results.

This page intentionally left blank

CHAPTER 11

Multimedia Networking

Today, we see a convergence of various technologies in the areas of digital recording, home electronics, communication, digital entertainment, Internet access, wireless networks, and digital broadcasting. This has resulted in refined content creation practices and well-engineered digital distributions, thus enabling digital media to play a pervasive role in everyday life. So far, the chapters in this book have tried to show how well-developed media-capture devices, sophisticated software processes, and the advances in compression technologies have made it feasible to create a plethora of media information. Additionally, the accessibility of this content is made possible by digital networks. Telecommunication networks, Internet, digital TV cable networked distribution, wireless networks, and fiber-optic distribution are good examples where media information is constantly exchanged for communication, entertainment, and other businesses. Different applications impose requirements and constraints on how media information is communicated on networks of varying bandwidth capabilities. Modern computing methods and various industry and consumer applications have come to rely on digital networks and, in particular, multimedia networks geared toward transmitting multimedia data traffic. The constantly growing need for various multimedia-related communications has made networks one of the most active areas of research and development. This third part of the book covers introductory and advanced topics in multimedia communications.

Although a mature digital distribution infrastructure exists today, the plain old telephone service (POTS) was the primary focus of voice-band communications for years. POTS is a well-engineered network based on circuit switching of up to 3 KHz voice signals and provides real-time, low-latency, and high-reliability communication for low-bandwidth signals such as voice. However, POTS does not suffice for wideband communications that involve audio, images, video, and other media combinations.

Digital networks based on packet switching are better suited to move bursts of media data among communication devices. The evolution of packet-switching technology began (independently of POTS) in the 1960s with the work of many pioneers such as Ivan Sutherland, Bob Taylor, Lawrence Roberts, and Leonard Kleinrock, with funding from the Department of Defense. The first large-scale deployment of this was the creation of the ARPANET, which then grew into today's Internet. The term *Internet* refers to a global networked system that logically links together globally unique address spaces based on the IP protocol, and is able to support communications based on a family of TCP/IP-based protocols. This allows multiple independent networks of arbitrary size and complexities, such as the ARPANET, ground-based packet radio networks, satellite networks, and local area networks, to join together and communicate based on standardized protocols regardless of the different gateways and network architecture supported. The huge capital investments made by government, public, and private sectors have resulted in a pervasive network that allows media information interchange using many devices—computers, pagers, cellular phones, PDAs, and so on—and have made information access easy and nomadic.

This chapter starts with an overview of some of the common techniques and terminologies in digital networks. Section 1 describes the OSI architecture, Section 2 discusses the switching technologies used in local and wide area networks, and Sections 3 and 4 discuss communication modes and routing methods, respectively. After that, Sections 5 and 6 describe various multimedia-related requirements for networked distribution, and how to evaluate distributed multimedia performance. Finally, Section 7 describes protocols used for data communication and, in particular, details the protocols that were designed to help set up and direct multimedia traffic.

1 THE OSI ARCHITECTURE

The internetworked communication among computers can be a complex setup requiring data to be transmitted via different types of mediums using multiple types of equipment, each sharing different protocols and set up by different manufacturers. To reduce this complexity, the Open Systems Interconnection (OSI) standard was proposed by the International Organization for Standardization (ISO) in 1982. The OSI standard is a multilayered architecture as described in Figure 11-1 and consists of seven logical layers:

- *Physical*—The Physical layer is concerned with transmitting and receiving unstructured bit streams over any physical medium. It defines the electrical and mechanical properties of the physical interface at the signal level, for example connectors and cables, and also specifies the step-by-step sequences to be performed by the interfacing circuitry. Some examples of the Physical layer are RS-232-C and X.21.
- *Data Link*—The Data Link layer ensures uniform transfer of data across the physical medium by establishing, maintaining, and terminating a link. It is logically divided into two sublayers—a lower Medium Access Control layer, which mainly regulates the control of access to the physical medium, and a higher

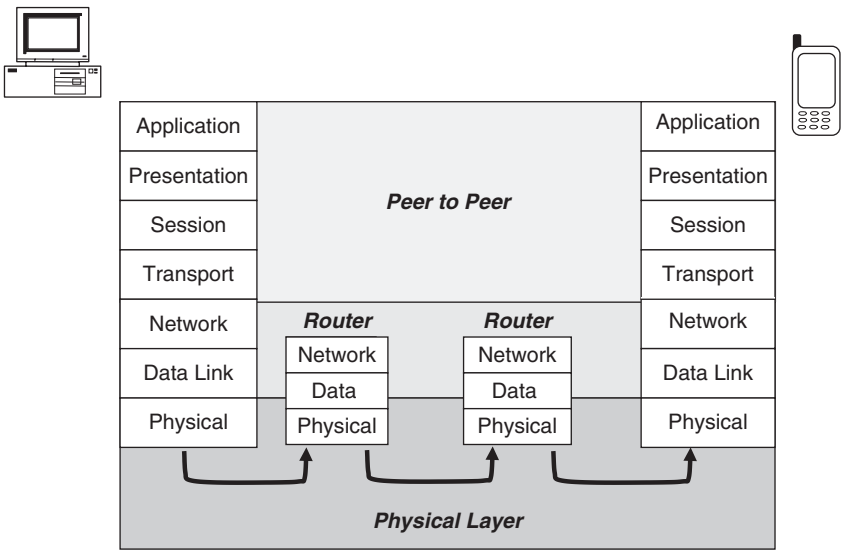


Figure 11-1 Communication using the OSI architecture. The OSI architecture consists of a seven-layer model, where each layer plays a part in packetizing and reliably transporting data from a sender to a receiver.

Logical Link Control layer, which provides flow control, detection, and retransmission of lost or dropped packets. Examples of the Data Link layer include Ethernet, Token Ring, High-level Data Link Control (HDLC), and Synchronous Data Link Control Layer (SDLC).

- *Network*—The Network layer provides the upper layers independence from the switching technology used to communicate data. It is responsible for the routing of data from one end to the other through a network, which can be packet switching or circuit switching. It also provides services for internetworking, error handling, and traffic congestion control. Examples of the Network layer include X.24 and IP.
- *Transport*—The Transport layer is responsible for reliable and transparent transfer of data between end points. It also supports services for end-to-end flow control and end-to-end error recovery. Examples of protocols in the Transport layer are TCP and UDP.
- *Session*—The Session layer provides a means for establishing, managing, and terminating connections between processes, for example a long file transfer between two hosts. It might also provide synchronization and checkpoints to restart service.
- *Presentation*—The Presentation layer performs data transformations to provide a standardized interface to high-level applications. It helps resolve the syntactic differences when the data differs from machine to machine.
- *Application*—The Application layer is the highest layer, which can be used by user applications, such as Telnet, FTP, HTTP, SMTP/MIME, and so on.

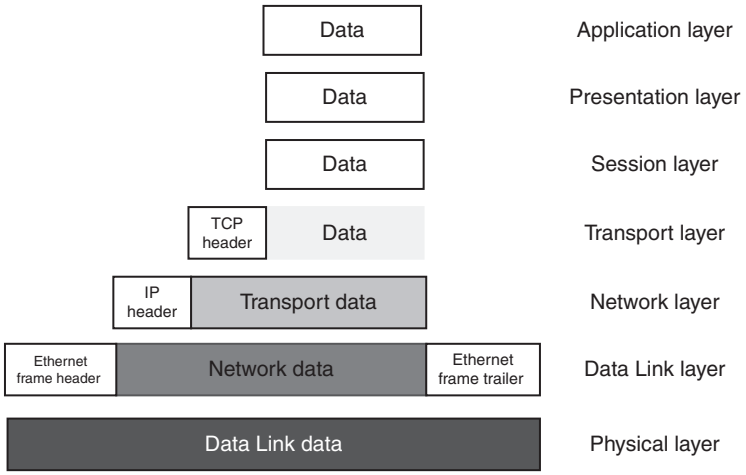


Figure 11-2 Construction of a packet through the OSI layers. The first three layers perform peer-to-peer management of getting data ready and setting up communication sessions. The subsequent layers are involved in transporting, routing, and, ultimately, delivering data through the physical medium.

The first three layers are peer-to-peer communication setup layers, where the top level nodes open, set up, and maintain connections for transporting the data. The actual transporting, routing, and overall management of data delivery are taken care of by the lower layers. Because data delivery might include hops across multiple networks via routers with errors occurring during transmission, the data itself needs to be encapsulated between specific protocol headers, such as TCP, IP, or Ethernet. Figure 11-2 shows a description of this anatomy, and the relevant protocols are discussed later.

2 LOCAL AND WIDE AREA NETWORKS

Networks set up for communication can be divided broadly into two categories based mostly on distribution over a geographic area: local area networks (LANs) and wide area networks (WANs). You can think of the Internet as numerous LANs interconnected by WANs. Although the ultimate purpose is the same—to communicate information packets—they generally involve different technologies, different data management styles, and different lower-level Medium Access Control (MAC) protocols for communication.

2.1 Local Area Networks (LANs)

Local area network (LAN) is a term used for a network when it is restricted to a small geographical area and has a small number of stations. Although there is no upper bound on the number of computers or on the size of the geographical area, the one

common feature that defines a LAN is a single, shared medium to transport data between computers. Hence, protocols to access, use, and relinquish control of the medium are of primary importance in any LAN. These protocols typically reside in the Medium Access Control (MAC) sublayer in the OSI-specified Data Link layer. The Data Link layer, as mentioned earlier, is logically divided into two sublayers:

- *Medium Access Control (MAC)*—This lower sublayer is responsible for assembling frames of data at the source and dissembling them at the destination. It performs error correction and node addressing and regulates access to the physical medium using a variety of protocols.
- *Logical Link Control (LLC)*—This higher sublayer performs error control and flow control, while interfacing with the higher OSI layers. Its task is also to perform addressing for the MAC layer.

The MAC layer connects a node to the physical medium and governs access to it. A variety of protocols exist, which are incorporated into standards that control access to the physical medium and can be divided into the following subcategories: round-robin, reservation, and contention.

2.1.1 Round-Robin

The shared network is organized in a logical ring topology, and each computer or node connected to the network is given an opportunity to transmit in a sequential manner, as determined by the logical ring structure. If a node takes over the network to transmit to another receiver on the network, it must let other nodes know of its network control, transmit its data, and then relinquish network control. The access to the network is now open and the next node in the logical sequence is given the opportunity to transmit data. The control of handing out turns in the ring structure can be centralized or distributed. The IEEE 802.5 Token Ring protocol is an example that adopts this round-robin strategy to share the network and is illustrated in Figure 11-3.

In a token ring, a special bit pattern called a token is used to control access. The token circulates around whenever no communication is occurring. When a node needs to transmit, it captures the token, changing the network state from idle to busy and temporarily preventing other nodes from transmitting. When transmission is completed, the node releases the token, permitting other nodes to seize the token. Implementation of this protocol requires that each node be able to *listen*, where a node is looking for a token if available, else simply forwarding the data, and a *transmit* mode when it has detected the token, seized it, and now transmits its own data. The token ring supports data rates at 4 Mbps or 16 Mbps over shielded twisted pair of wires. Larger data rates are possible with fiber-optic cables; however, because of their relatively fast transmission, the source nodes absorb the token. To resolve this, the Fiber Distributed Data Interface (FDDI), capable of 100 Mbps, was designed as a successor to the original token ring. It supports a dual-ring topology with a primary ring for data transmission and a secondary ring for fault tolerance. FDDI supports both synchronous and asynchronous modes.

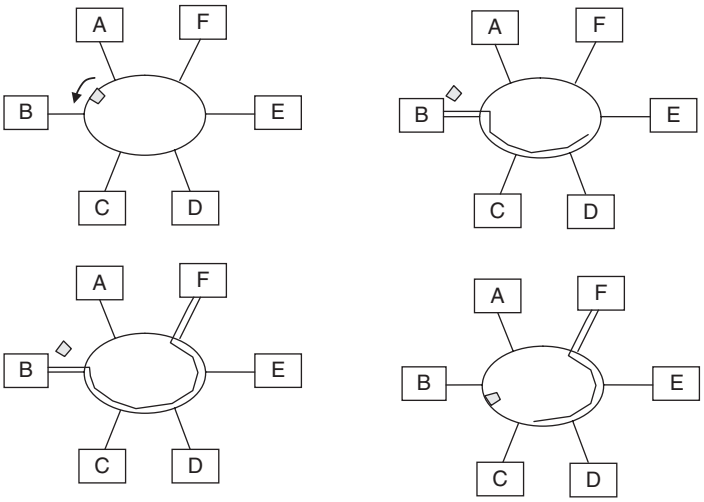


Figure 11-3 Token ring example. Six nodes are connected in the form of a ring. In the upper-left image, a free token is floating in the ring with no data transmission. In the upper-right image, B grabs the token and starts to send data. In the lower-left image, B transmits data to F. In the lower-right image, B finishes transmitting data and releases the token.

2.1.2 Reservation

Another approach to sharing a medium is to assign an explicit reservation scheme or an on-demand reservation scheme where the time on the medium is shared by dividing it into slots, also known as time division multiplexing. In an explicit scheme, each node is assigned reserved time slots to transmit data. Time slots can also be reserved on an on-demand basis for extended or indefinite time periods. A node that has been allocated a particular time slot is allowed to transmit its data during that allotted time slot only, and not any other time slot. Thus, a central controller is needed to manage time slot allotments or assignments to prevent multiple nodes from transmitting at the same time. Shared satellite channel is an example of this scheme. An example of a time-based reservation scheme is illustrated in Figure 11-4.

2.1.3 Contention

Unlike the previously discussed protocols, which impose a control to access the medium, another method is to impose no control. Every node connected to the medium is allowed to transmit, if it finds that the medium is free for transmission. Here, the node that wants to transmit data must listen to the network to sense whether there is data flowing—also termed as carrier sense. If there is no data flowing, the node is free to transmit data. If there is data flowing, the node cannot transmit and should wait a random amount of time before sensing the medium again to check whether it is free for transmission. This lack of simple centralized/decentralized administration can have multiple nodes transmit data at the same time, causing a collision, for example if two nodes simultaneously probe and find the network to be free, both will transmit data, causing collisions. To resolve collisions, each node also listens to its transmission and

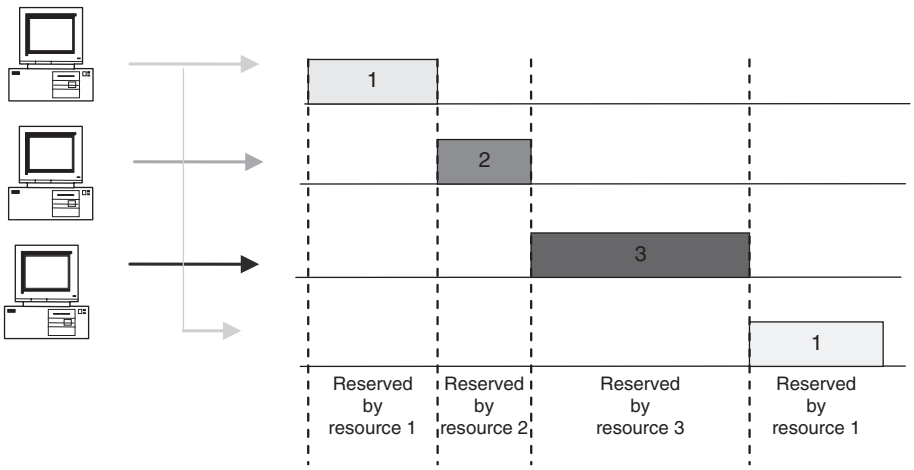


Figure 11-4 Time-based reservation

compares the received signal with the transmitted one. If they are different, there was a collision and the node stops transmitting data and resumes the process after a random delay. The IEEE 802.3 Ethernet is an example of this method, which uses Carrier Sense Multiple Access/Collision Detection (CSMA/CD).

2.2 Wide Area Networks (WANs)

Wide area networks cover wider geographical areas and connect various LANs together so that computer nodes on one LAN, which are normally limited to a room, a building, and so on, can communicate with computer nodes located in other LANs. The most well-known WAN is the Internet. WANs are typically built using routers that connect various public and private leased lines and often the physical medium is composed of many different types of wires, fibers, or other lines that provide switching technologies to establish a route. The protocols used to establish connections on such nonshared mediums normally employ switching technologies and use different protocols from the broadcast ones used in a MAC layer, as is the case with LANs.

2.2.1 Circuit Switching

In circuit switching, an end-to-end circuit is established in a dedicated manner between the sender and the receiver nodes for the duration of the connection and also for a specified bandwidth. The Public Switched Telephone Network (PSTN) is an example of a WAN operating by circuit-switching technology. The PSTN was initially put into place for voice communications, but has also been used for data transmission and forms the basis of all modem-based communications, Integrated Services Digital Network (ISDN), and Asymmetric Digital Subscriber Line (ADSL). To support multiple users and multiple data rates on the same line, it often resorts to an instance of time division or frequency division multiplexing.

The data rates the circuit-switching technologies provide are very low and are only sufficient if the demanded data rate by a user is constant—for example, in data-only

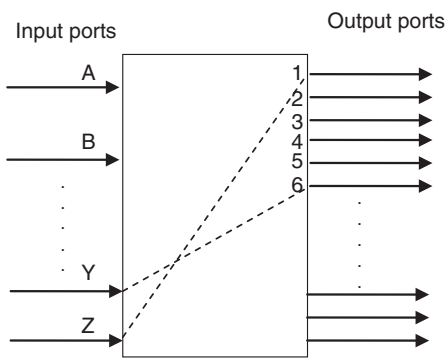
transmission or constant bit rate multimedia video and audio traffic. However, if the data rate is variable, especially in real-time multimedia communications, circuit switching is inefficient.

2.2.2 Packet Switching

Packet switching is more efficient for varying data rates and real-time communications because higher data rates can be accommodated. Prior to transmission, the sender node breaks the data into smaller sections encapsulated into packets. Each packet usually has 1000 bytes or less, with a header that carries necessary information, such as the destination address, routing information, and so on. Packet-switching technology normally follows one of two approaches: connectionless and connection oriented, and is explained further in Section 4. Packet-switching networks provide higher data rates compared with circuit-switching networks; however, their performance starts to deteriorate when the network gets congested. Normally, different arbitration policies are enforced to avoid this situation. Examples of protocols that are implemented for packet switching are X.25, which is the most common, and IP. These normally reside in the Network layer and also perform extensive error checking to ensure no data corruption occurred during transit.

2.2.3 Frame Relay

Frame relay is another simple connection-oriented setup that provides capabilities of both switched virtual connections (SVCs) as well as permanent virtual connections (PVCs). Modern networks with high-bandwidth capacities, such as fiber optics, have lower error rates, making it unnecessary to add on extra bits per packet for error detection and recovery. These high-speed, low-error networks now allow high-speed parts of an established virtual route between two nodes to be time shared or multiplexed for different route connections, thus increasing efficiency. Frame relay, therefore, provides a means for statistically multiplexing many logical connections (virtual circuits) over a single physical transmission medium. Since many logical connections co-exist on a single physical line, each connection needs to be identified with a unique identifier that will allow for data to be logically tied to a connection. The data link connection identifier (DLCI) performs this function. This is illustrated in Figure 11-5. From a user's perspective, frame relay is a



ROUTING TABLE			
Input port		Output port	
Port	DLCI	Port	DLCI
Y	*m*	1	*n*
Z	*p*	6	*q*
..	..	..	..

Figure 11-5 Principles of frame relay. Input ports Y and Z are mapped to output ports 1 and 6, respectively. This setup is for a particular session of decided routes.

multiplexed interface to a packet-switched network. Frame relay includes all the operations provided by X.25, except that there is reduced error checking, which mean no acknowledgements, flow control, or error control because these mediums have minimal loss and corruption. However, the error detection, correction, and flow control can be optionally performed by a higher layer such as the Transport layer. Frame relay is, thus, a simplified X.25 protocol with minimal services. Frames of data multiplexed can have a length of up to 1600 bytes. Bad frames are discarded. Data rates for frame relay are higher, for example T1 supports 1.5 Mbps, whereas T3 supports up to 44 Mbps.

2.2.4 Asynchronous Transfer Mode

Most of the different switching technologies discussed previously have some characteristics in common—first, they support variable packet sizes and second, the transmission can occur synchronously, especially in circuit-switching networks, because a dedicated physical or virtual link is established from sender to receiver. In such cases, a connection is established between two nodes before data is transmitted and abandoned when the transmission is complete. The two nodes, thus, use resources and allocate bandwidth for the entire duration, even when they are not transmitting any data, and even if the connection is set up as a multiplexed connection. Asynchronous Transfer Mode technology solves this by first statistically multiplexing where packets of all traffic routes are merged together into a single queue and transmitted on a first-come, first-served basis, rather than allocating a time slot for that route when no data is flowing. This maximizes the throughput compared with synchronous mode transmissions. ATM networks, thus, make it feasible to support high-bandwidth multimedia requirements.

Another important distinction is that ATM adopts only small, fixed-length packets, called cells, which provide the additional benefit of reducing latency in ATM networks, as shown in Figure 11-6. ATM networks can provide high speed and low delay

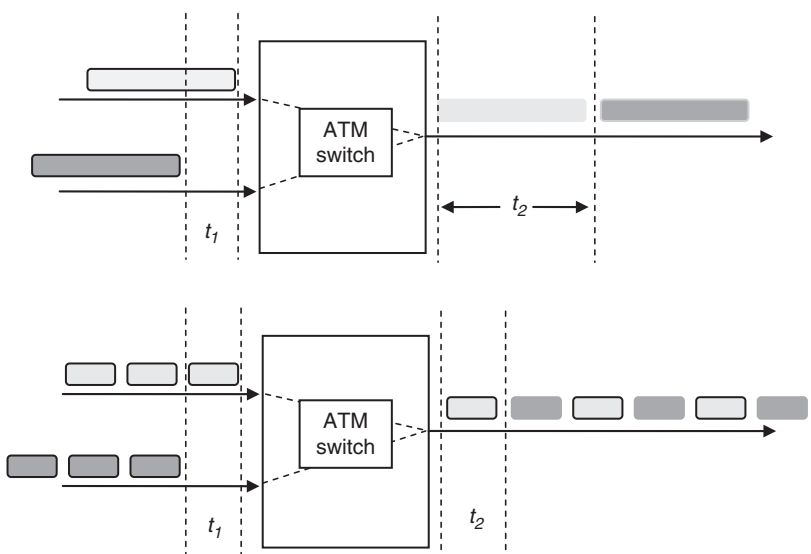


Figure 11-6 Smaller packets reduce latency during packet switching

and, hence, can support up to 2.5 Gbps. ATM switching also flexibly supports various technologies, such as IP, DSL, and wireless, and is capable of providing a guaranteed bandwidth and performance (or Quality of Service) levels.

3 MODES OF COMMUNICATION

Multimedia communication, or, more generally, information communication, can be divided into three categories as mentioned in the following sections: unicast, multicast, and broadcast. Sometimes multicast and broadcast are equivalent, but we make a distinction, needed especially because of real-time needs imposed by multimedia. The main difference can be seen at the lower network levels, where routing takes place to communicate data.

3.1 Unicast

Unicast is derived from broadcast and refers to a setting where one sender node sends information to one receiver node. This can be achieved by setting up a dedicated link between a sender and receiver, for instance in circuit-switched telephony, or in a connectionless link such as sending an e-mail to a specific receiver. Unicast servers work by providing a media stream to each user, an architecture that is hard to scale for many simultaneous users.

3.2 Multicast

Multicast is a term used for information delivery to a group of destinations simultaneously over a network. This is more efficient than unicast because it delivers messages to each of the receivers by sending only one copy of the information. The network duplicates packets only when necessary, thus conserving bandwidth. Implementing multicast routing is much harder. In the multicast mode, each multicast receiver can receive information in a unicast manner, with one explicit packet sent to each receiver, or in a more sophisticated way where the network resources duplicate packets along different routes toward the end receivers.

3.3 Broadcast

A sender node can also broadcast a message on a network. In this case, the message packet is not intended for a specific receiver node or a set of receiver nodes but to all the nodes on a network. The receivers have to be "listening" to a broadcast rather than the previous cases where data is delivered specifically to them. Correspondingly, in practical implementations, broadcast protocols use different ports to listen. Broadcasts of live and prerecorded multimedia data to an unlimited set of receivers over an IP network is a challenging task, especially because all the intra- and intermedia synchronization needs to be maintained in the presence of network delay, jitter, and congestion.

4 ROUTING

The network can be viewed as a graph of interconnected nodes, where an edge between two nodes can be interpreted as a physical link. The nodes can be on a LAN or WAN. If one sender node on this network wants to send data to a receiver node, the goal of routing then becomes to determine a path from the source node to the destination via a minimal number of intermediary nodes that can be used to deliver information packets. An example of a route is shown in Figure 11-7 Routing is the main task of the Network layer, with the intermediary nodes acting as routers. It involves two problems:

- Finding an optimal or a good path in the routing graph. This can be difficult with changing network loads and even changing network topologies.
- Once a route is established, getting all the information packets though the intermediary routers to the receiver in an efficient manner.

4.1 Approaches to Routing

Routing from a source to a destination can take place over two different modes: connectionless mode and connection-oriented mode. In a connectionless mode, the path-finding algorithm is executed every time a packet needs to be delivered. Consequently, the path-finding algorithm needs to make use of current knowledge of network topology to find different hops to progress toward a destination, and each

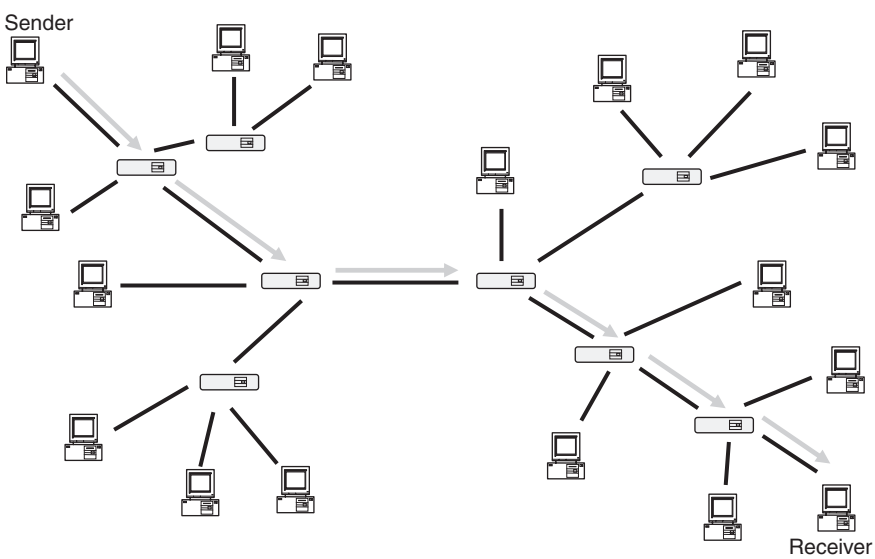


Figure 11-7 Routing example. A typical network consists of numerous computer nodes connected via router nodes organized as a connected graph. Given a sender and a receiver, routing algorithms have to determine an optimal path from the sender to the receiver node.

packet might find its way independent of other packets. For this process to work, every packet needs to carry the final destination IP address, so that each intermediary node can evaluate where to send it next. Connectionless routing protocols, such as IP, have several salient features:

- They are robust in the case of node failures and changing topology. If new nodes are added, routes are dynamically computed. The failure of a node does not affect the overall routing.
- Connectionless protocols tend to perform efficiently for short connections. In larger networks, packets might not get to their destinations as efficiently.
- Setting up networks becomes easy using connectionless protocols.

In connection-oriented modes, an explicit or implicit path from the source node to the destination node has to be set up once before data transmission begins. All the packets then follow the same path in the node graph from source to destination. Connection-oriented networking protocols include X.25, ATM, and frame relay. Some of the salient features of these routing protocols are as follows:

- Prior to transmission, an explicit/implicit path needs to be set up. This is done by the sender sending a connection setup packet to the destination. This special packet leaves a trace of the routing information at each node on a path, which includes a connection identifier (for that transmission session) mapped to a corresponding output port.
- All the subsequent transmission packets follow the same path, which amounts to a port lookup in a mapped table at every intermediary node. Hence, the routing can be efficiently done at run time, with a simple lookup as opposed to executing a path-finding algorithm while routing every packet in connectionless routing.
- Because there is a defined path in such methods, a statistical model of network traffic at any given time can be determined. This information can be used to control access to the network and avoid network congestion. For example, a new call to set up a route can be rejected if the network load is high.

4.2 Routing Algorithms

Routing is the main task of the Network layer. Together with the Network layer, the Transport layer protocols have been defined to get data from a source node to a destination node in a reliable manner. The two main approaches to routing are discussed in the following sections.

4.2.1 Static Routing

The overall route from a source to a destination in static routing is created using fixed paths. Routes for all source-destination pairs are precomputed, either programmatically by a central network hub or input by a network administrator. This source-destination route information is then broadcast to every node in the network, enabling each node to create a table with entries that resolve an outgoing link given

a sender and a destination. Any packet to be routed contains a destination address or a connection identifier (in case of connection-oriented networks). A node can quickly look up and resolve the outgoing link to forward the packet to the next node. Therefore, routing occurs efficiently and systematically. However, static routing methods are not fault tolerant. In case of node failure, packets will not be rerouted, and the sender nodes will have to wait until the affected pathway is repaired. Node failure in such cases results in request time-outs. Another drawback is computing optimal routes. Optimal routes can be computed and defined once, but would need to change every time the network topology changes. Static routing is, therefore, unattractive for large and dynamically changing networks, but is useful for small subnetworks and also to suggest default routes in a network.

4.2.2 Adaptive Routing

In adaptive routing, also known as dynamic routing, the path-finding algorithm alters the path that a packet takes through the network in response to changing network conditions. The changing conditions might include dynamically varying network topology as new nodes are inserted or obsolete nodes are removed and also local congestion and traffic conditions in parts of the network. IP is a common adaptive routing method used.

4.3 Broadcast Routing

Broadcast routing is a specific case of routing where a sender needs to broadcast a message to all the nodes in the network. One straightforward manner is to send the message packet to each destination node. But this requires that the source node have a complete list of all the destinations. Moreover, it turns out to be bandwidth wasteful. Another simple manner to achieve a broadcast is by flooding the network, where at each node every incoming packet is sent out to every outgoing packet, except the one it arrived on. The packet duplication followed here can easily congest the network (hence the name flooding), and there has to be a way to delete duplicate packets. For instance, each router can keep track of sent packets in a sequence and delete packets if they arrive again. The most efficient way to broadcast messages is by using a Spanning Tree Broadcasting protocol. A spanning tree, shown in Figure 11-8, consists of links in the network that includes all the nodes but has no loops. If each node knows the incoming/outgoing ports that belong to the spanning tree, broadcast can be achieved by each router only forwarding packets on the outgoing spanning tree lines. Using a spanning tree is most efficient, but requires that a spanning tree be computed and communicated to all nodes in the network, which is nontrivial under changing network topology and node failure.

5 MULTIMEDIA TRAFFIC CONTROL

Streaming and real-time distribution of media content typically requires large network bandwidth as well as timely delivery, decoding, and rendering of the media data on the client side. Multimedia files or real-time streams tend to be voluminous, and need

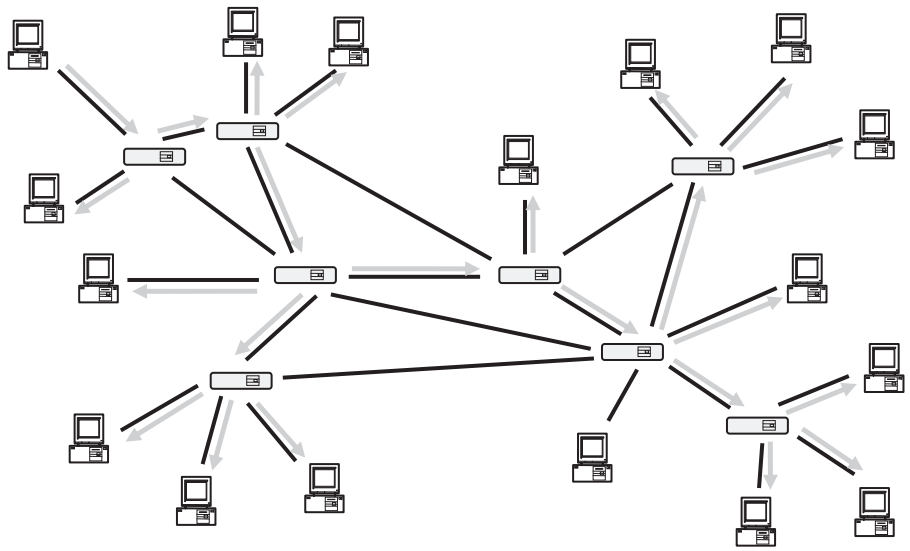


Figure 11-8 Spanning tree broadcasting. Every node knows whether it is on the spanning tree. To broadcast a message, all intermediary nodes need to copy and forward messages only on their spanning tree edges.

compression techniques to reduce the amount of data to be distributed. The compression mechanisms produce varying bit rates that result in traffic bursts during distribution. For multimedia traffic, the following bit rate categories are generally defined:

- *CBR*—CBR or constant bit rate is intended for real-time applications that require tightly constrained delay and jitter, for example voice communications or watching a live sports event over the Internet.
- *Real-time VBR*—Real-time variable bit rate is defined to also categorize real-time applications that need constrained delay and jitter but in this case, the sources can produce short data bursts.
- *Non-real-time VBR*—Non-real-time variable bit rate is used to classify multimedia traffic for non-real-time applications, which have "bursty" traffic characteristics.
- *UBR*—Unspecified bit rate is intended for non-real-time applications, which are tolerant to delay and jitter. Normally, UBR service does not guarantee on-time delivery and provides no synchronization to play in real time.
- *ABR*—Available bit rate is a term used for multimedia traffic where a sender can adjust or remain adjustable to the bit rate needs provided by the network. For example, applications might have the ability to reduce their bit rate if there is congestion on the network and vice versa.

It is easier to devise data-delivery strategies that need synchronization for constant and unchanging throughput as compared with variable bit rate delivery. For instance, you could compute a statistical measure of the latency between the sender and the

receiver nodes on the network and, thereby, maintain a small but fixed playout buffer that allows the media delivered to the client node to be rendered in an uninterrupted manner. Sometimes though, even with CBR networks, the throughput might vary with time due to the following reasons:

- *Node or link failure*—When a node fails, data needs to be rerouted along other nodes and links, causing traffic to increase on those routes.
- *Network congestion*—Congestion describes the state when the demand for the network's bandwidth capacity exceeds the physical availability of bandwidth. Thus, congestion happens when too many packets are present in a subnet or part of a subnet when the throughput decreases with increasing load. For example, on a 10 Mb/s network with 100 computers, if 50 computers are simultaneously transmitting data to the other half each at 1 Mb/s, the network demand of 50 Mb/s will exceed the network capacity of 10 Mb/s, causing congestion and resulting in lost or dropped packets. Traffic policing mechanisms need to be put in place for controlling congestion.
- *Flow control*—This control is imposed during end-to-end delivery to prevent loss of data at the receiving end due to slow receiving nodes. If the sender node transmits data quicker than a receiver can buffer and consume data, it will result in buffer overflows, ultimately causing loss of data. For example, on a network with a large 1 Gb/s capacity, if a fast computer was sending data at 200 Mb/s to a slower receiver capable of consuming data at 10 Mb/s, packets would not be picked up in time by the receiver, resulting in data loss. Flow control protocols need to be established here to control effective flow of data.

5.1 Congestion Control

Congestion control aims to provide a mechanism to control the volume of data packets flowing in a network. These methods to automate and shape network traffic generally borrow concepts of bandwidth classification, queuing theory, monitoring and policing flow of packets, congestion management, and fairness to network access. One standard way to manage traffic is by regulating the data bursts and redistributing them uniformly over time. For example, suppose a video stream produces variable data rates resulting in 140 packets the first second, 10 packet in the second second, 30 packets in the third second, and 140 packets in the fourth second. Instead of these bursty packet rates, a traffic-management algorithm could ensure that 1 packet is inserted into the network every 12.5 milliseconds.

Two predominant ways to shaping traffic are the leaky bucket and token bucket. These methods result in lowering the overall delay and reduce packet loss.

5.1.1 Leaky Bucket

The leaky bucket algorithm is used to control the rate at which a sender node injects packets into a network. It manages the burstiness generated by multimedia data packets and injects them into the network at a steady rate as opposed to an erratic rate of high and low volume flows. A good working analogy to this method is controlling automobile traffic entrance into a freeway. Although cars (packets) may come at any

rate to enter the freeway (congested network), a signal monitors and allows only one car (packet) to enter the freeway (network) every few seconds. This allows a steady flow and lowers the chances of a freeway traffic jam.

In this method, a policing mechanism is put in place between the node and the network. The packets to be released into the network are first kept in a bucket with a hole at the bottom. The bucket has a capacity and queues at most N packets. If a packet arrives when the bucket is full, the packet is discarded. Packets (bytes) leak through the hole into the network at a constant rate or *t* packet (bytes) per second, thus regulating the traffic into a steady flow, as shown in Figure 11-9.

5.1.2 Token Bucket

The token bucket method differs from the leaky bucket in that the leaky bucket imposes a hard limit on the data or packet rate being transmitted, whereas the token

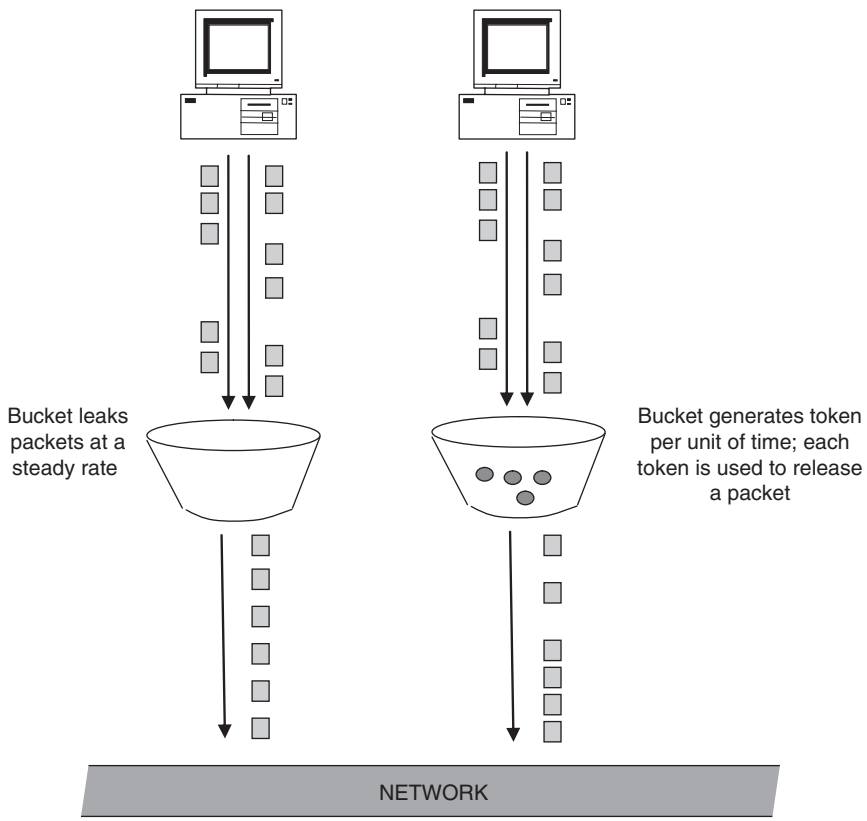


Figure 11-9 Traffic congestion control. The left figure illustrates the leaky bucket algorithm when packets are released at a steady rate to the network. The right figure illustrates the token bucket where packets are released to the network if tokens are available in the bucket.

bucket allows a certain amount of burstiness while still maintaining an upper limit on the average packet rate being transmitted.

The algorithm is illustrated in Figure 11-9 and works by adding a token into a bucket every t seconds. The bucket has a capacity to hold N tokens. After it reaches its capacity, no additional tokens are generated. When a packet of n bytes arrives, n tokens are removed from the bucket and the packet is released to the network. If n tokens are not available, the packet is buffered until additional tokens get added over time. This method allows bursts of up to N bytes and over a longer period, the output rate is limited to $1/t$ bytes. Initially when a computer is ready to transmit and tokens are available in the bucket, the transmission begins by a burst of packets (or bytes) because tokens are available. After consumption of all the tokens, the network traffic gets regulated. If C is the capacity of the token bucket (number of tokens), M is network throughput, and R is the steady state rate, then maximum time for which a burst lasts T_{burst} is given by

$$\begin{aligned} \textit{Effective Flow Rate} &= (M - R) \\ T_{Burst} &= \frac{(M - R)}{C} \end{aligned}$$

5.2 Flow Control

Flow control is the process of overseeing the data transmission rate between a sender and a receiver on a network. This differs from congestion control, which is used for controlling the flow of data packets over a network when congestion has occurred. Flow control is a contract between a sender and a receiver node on the network. Flow control is necessary to control cases when a fast sender sends data to a slow receiver, which can happen when the receiving node has a heavy traffic load or less processing power compared with the sending node.

5.2.1 Closed Loop Flow Control

In closed loop flow control, the sender is able to get information about the network congestion status and whether the packet being sent by the sender reaches the intended receiver. This information can then be used by the sender node in a variety of ways. For example, the sender might adapt its transmission activity by either increasing or decreasing it depending on network conditions. ABR is one place where closed loop flow control is effectively used.

5.2.2 Open Loop Flow Control

In open loop flow control, the sender has no feedback from the receiver. Flow control in such cases is achieved by allocating resources by reservation. For example, in ATM networks, resource allocation is made at connection setup and the resources effectively help control the flow of traffic. This simple means of flow control is the preferred flow control mechanism used by CBR, VBR, and UBR multimedia.

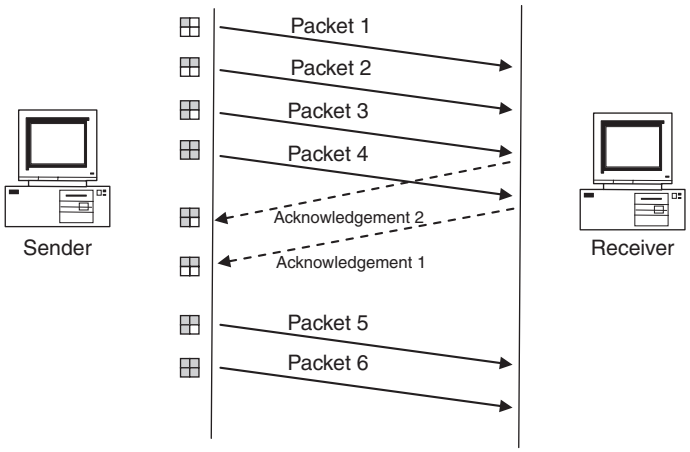


Figure 11-10 Flow control using the sliding-window mechanism. The sender (left) has a window of four packets. It transmits packets until the window is full and waits for acknowledgements from the receiver (right). When acknowledgements are received, more packets can be transmitted.

5.2.3 Transmit Flow Control

Transmit flow control is the process used to control the rate of data transmission from a sender so that the data can be received successfully by another terminal. One standard way to achieve transmit flow control is by using the sliding-window mechanism. The sliding-window algorithm allows the sender node to transmit data packets at its own speed until a window of size W is used up, as illustrated in Figure 11-10. The sender then stops sending further packets and waits to receive acknowledgements from the receiver for the sent packets. The received acknowledgements open up the transmission window when the sender can transmit more data. The sliding window is used in the TCP protocol to ensure flow control.

6 MULTIMEDIA NETWORKING PERFORMANCE AND QUALITY OF SERVICE

Any multimedia application or presentation that requires data communication between two nodes defines performance requirements from the network to maintain the integrity of the presentation. This might be real-time requirements as in video teleconferencing, streaming live video over the Internet, or static examples where movies and music are downloaded prior to being played. Networking performance is important when it comes to delivering multimedia data because the data is normally very large, needing intramedia (video frame rate) and intermedia (video and audio) synchronization. A variety of factors can affect the packet delivery from the source sender to the destination, mainly defined by average throughput, error rate, and latency.

6.1 Throughput

In networked communication, throughput can be defined as the effective rate at which a network sends or receives data, and is measured as the amount of bits transferred over a period of time. Throughput is a good measure of the channel bandwidth capability and is measured in bits/second or packets/second. Factors that affect the throughput are network congestion, bottlenecks in a network, or node failure. Network congestion and bottlenecks can result in dropped packets because the intermediary router buffers might be full when the packet gets to it. The receiving node in such cases might want to detect a dropped packet and signal a source to resend it (for example, in TCP) or ignore the dropped packet in the interest of maintaining real-time throughput (for example, in UDP). Node failures during a connection-oriented session also result in dropped packets because its input and output lines are not actively transmitting data.

6.2 Error Rate

Error rate or error ratio measures the total number of bits (packets) that were corrupted or incorrectly received compared with the total number of transmitted bits (packets). This is measured in terms of bit error rate (BER) or packet error rate (PER). Data corruption can occur inadvertently due to switching problems at the lower Physical layer, dropped bits and packets during transmission, misdirected packets, packets combined together, or collisions on the same medium, and so on. Errors can also be deliberately induced if intermediary nodes drop packets to maintain a contracted bit rate service. The receiver node should have enough information to detect the error and request a retransmission if necessary. In fiber optics, BER ranges from 10^{-8} to 10^{-12}, whereas in satellite-based transmission, the BER is of the order of 10^{-7}. To help reduce and correct errors, error-correction codes are also communicated for the data packet, which allows the receiver to check whether the packet was received correctly and, if necessary, request retransmission by sending a negative acknowledgement (NACK) back to the sender. The sender learns about errors in two ways: receiving a NACK from the receiver or the sender times out when a positive acknowledgement is not received.

6.3 Delay or Latency

The process of transmitting data packets from sender to receiver is not instantaneous and happens in a small, though finite, time interval. The delay is a very important issue to multimedia communications, as it heavily impacts the real-time needs and overall design architecture of a networked application. Delay, also called latency, is normally measured as an end-to-end delay or a round-trip delay and is the total time required for a sender to send a packet and receive an acknowledgement from the receiver. It can be described as the sum of network delay and interface delay.

- *Network delay*—Network delay is composed of transit delay and transmission delay. Transit delay is caused by the time needed to send data on a physical connection between the sender and the receiver. Transmission delay is the time needed to transmit a packet through the network as a result of processing

delays at intermediary nodes to look up routing tables, compute local routes, and also buffer management.

- *Interface delay*—An interface delay is incurred between the time a sender is ready to begin sending and the time a network is ready to accept and transmit the data. This might happen due to policing imposed by congestion control mechanisms.

Although latency does need attention, it can be worked around by suitable buffering strategies when the latency is constant. A larger problem is caused if the latency itself varies during the transmission. This is known as jitter. Depending on how the latency varies, network traffic can be defined to be asynchronous, where there is no upper bound on the latency, synchronous, where there is a small upper bound on the latency, or isochronous, where the latency is constant. It is more desirable for multimedia applications to have isochronous or even synchronous delays, where an appropriate playout buffer at the destination can be used to resolve the latency issue.

6.4 Quality of Service

Quality of Service, also termed QoS, indicates how well a network performs with multimedia applications, regardless of whether the network is conforming to the required traffic for the application. It effectively defines the capability of a network to provide a level of service to deliver network packets from a sender to a receiver(s). When data packets are transmitted from a source to a destination, the route can consist of multiple networks and make use of many network resources. There will always be multiple users transmitting across the network that might overload the network. Some applications transmitting data might be more critical than others and need to get data packets routed quickly and efficiently. The need for this shared network to provide deterministic and priority-based delivery services is even more relevant in real-time applications. Such deterministic delivery services demand that both the sender and the network infrastructure have capabilities to request, set up, and enforce the delivery of the data packets. It should include a dedicated bandwidth, bounded latency, controlled jitter, and less error rate during the transmission. Collectively, these services are referred to as bandwidth reservation and Quality of Service (QoS).

QoS works by slowing low-priority packets down and even completely discarding unimportant packets under overloaded network conditions. This leaves room for important packets to flow through the network and reach the destination as quickly as possible. Once the resources along the route from sender to destination are each made aware of the importance of packets flowing through and also how much data they can enqueue, they can act as traffic shapers by first transmitting the more-important packets and delaying less-important ones. Different applications have varied QoS requirements. Some sample requirements include the following:

- Streaming media might impose QoS requirements on a guaranteed throughput and low latency but a more tolerable jitter.
- Videoconferencing or IP telephony, which are real-time applications, might require stricter limits on jitter and maximum delay, but could tolerate errors.

For example, if an error is detected, there should not be a need to retransmit the packet and cause delay problems for future packets.

- Online games where multiple players play together across a network need more real-time QoS constraints with bounded latency and jitter.
- A remote surgery, which is a health-critical application, might need a guaranteed level of connection availability compared with other requirements.
- An application that backs up data files does not need any strict bounds on throughput or even latency and jitter, but needs to ensure no errors in transmission.

Services that require a defined minimum level of bandwidth and can withstand a maximal latency are known as *inelastic* services. Real-time applications will demand inelastic services from the network. On the other hand, *elastic* services take advantage of all the bandwidth available to them, however small it might be. An example of an elastic service can be seen in an application that performs data backups. A better categorization of QoS levels are *best-effort* service, *differentiated* service, and *guaranteed* service. These services and sample protocols to implement them are discussed next. When the requirements to provide a certain level of QoS is justified, the sender and receivers typically enter into a contractual agreement with network providers, which specifies that the network will provide a guaranteed level of throughput, latency, error, and overall performance. This normally mutual agreement by the sender/receiver and the network is achieved by prioritizing traffic flow in favor of the higher QoS applications on the network.

6.4.1 Best-Effort Service (Also Called Lack of QoS)

Best-effort service is also termed as *lack of QoS*. The best-effort service provides basic connectivity with no guarantees or priorities to packets in transit across the network. This is best characterized by a First In First Out (FIFO) queuing algorithm, which takes effect when the congestion is experienced. FIFO does not differentiate between flows and is the default queuing algorithm on intermediary nodes and routers. It makes no decision based on packet priority, order of arrival, and any buffer allocation but simply routes each packet in order of arrival. Packets get dropped if the buffer at the router is full, which is possible when congestion is experienced.

6.4.2 Differentiated Service

This is also called *soft QoS*. In this model, packets are marked with a priority, which is used to give partial preference while routing of packets at intermediary nodes. The traffic for some packets is, thus, different from the rest, hence the name differentiated service. This provides faster handling, more average bandwidth, and lower average loss rate compared with the best-effort service. This service works by having routers employ various queuing strategies to customize performance according to the requirements marked by the priorities. The priorities are set in the packet headers depending on the protocol used. For example, in the IP protocol discussed in Section 7.1.1, 6 bits in the header are used for differentiated services. Different queuing strategies are used to shape and forward packets, some of which include priority queuing (*PQ*), custom queuing (*CQ*), and weighted fair queuing (*WFQ*).

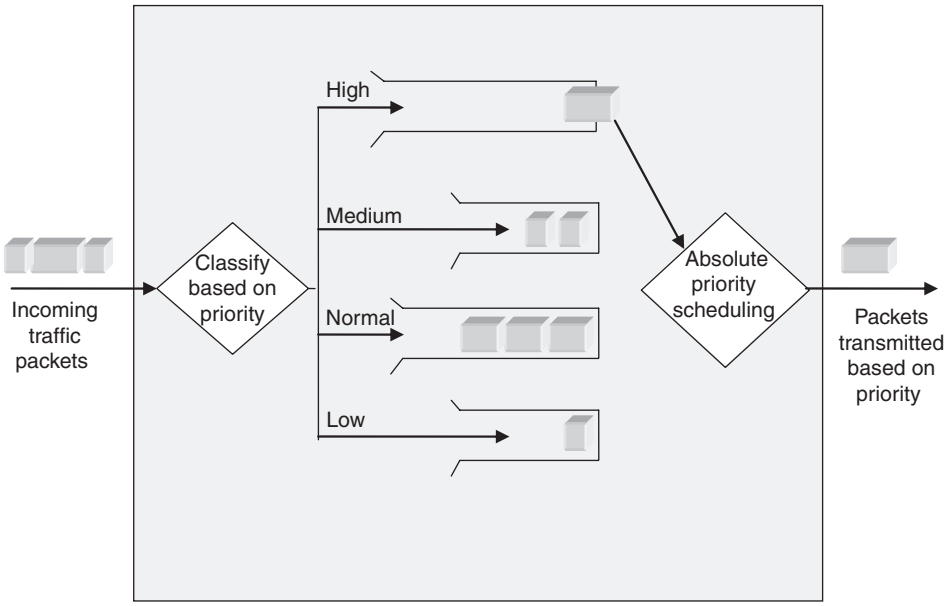


Figure 11-11 Priority queuing at an intermediary router. Incoming packets are put into four queues depending on their priority. At each transmission instance, the high-priority queue gets dealt with first, followed by the medium-priority queue if the high priority queue is empty, and so on.

In *PQ*, the incoming packets at the intermediary routing nodes are put into one of four queues—high, medium, normal, and low. When the router is ready to transmit a packet, it first searches the high queue for a packet. If the high queue is empty, it advances to the next medium queue, and so on, as shown in Figure 11-11. Therefore, *PQ* ensures that important traffic gets the fastest handling at each point where it is used. *PQ* is meant to be used on low-bandwidth links. The queue sizes or buffers have to be preconfigured and do not adapt to network traffic.

The queuing algorithm in *CQ* places incoming packets in one of 17 queues. The first queue holds system messages, such as Keep-Aives, signaling, and so on, and is emptied with weighted priority. The router services the remaining queues 2 through 17 sequentially. At each queue, a preconfigured number of bytes are allowed to be transmitted and then servicing moves on to the next queue. This feature ensures a guaranteed throughput for all applications by ensuring that no application achieves more than a predetermined proportion of overall capacity, as illustrated in Figure 11-12. Like *PQ, CQ* also has to be statically configured and does not automatically adapt to changing network conditions.

WFQ's queuing algorithm is designed to automatically adapt to changing network traffic conditions. *WFQ* achieves this by sorting the incoming packets in a weighted order of arrival of the last bit, to determine the transmission order. Using the order of arrival of the last bit emulates an unbiased multiplexing in time, hence the name fair.

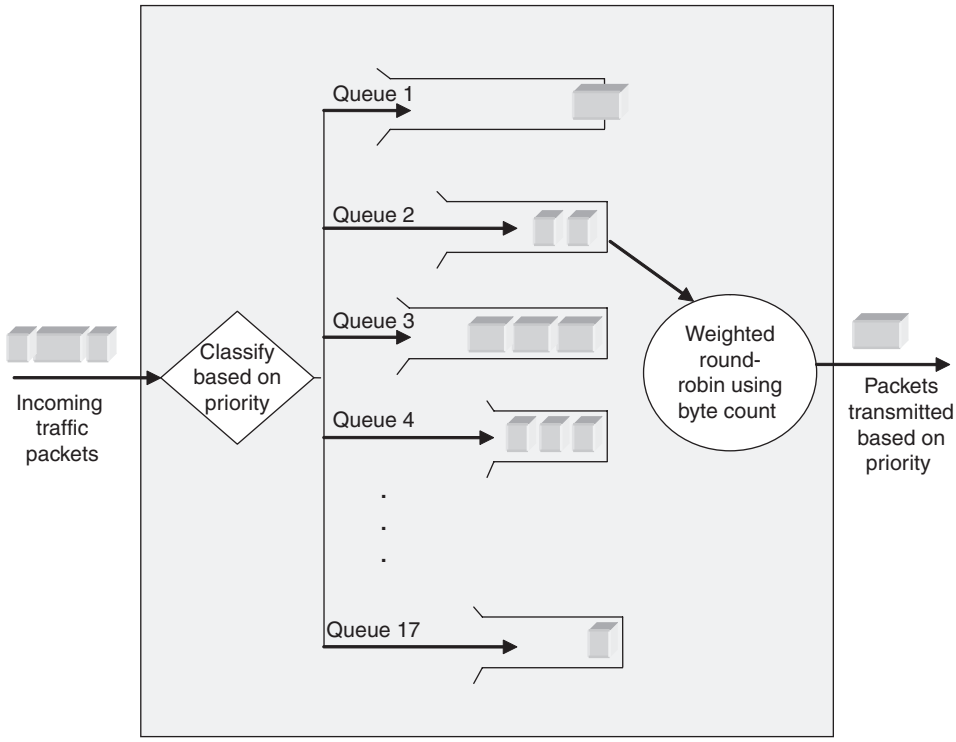


Figure 11-12 Custom queuing at an intermediary router. Traffic is handled by assigning a predetermined queue length to each class of packets. The 17 queues are then handled in a round-robin manner, which guarantees limited throughput to all classes of packets.

6.4.3 Guaranteed Service

This is also called *hard QoS*. In this service, packet delivery is guaranteed at any cost. There is an absolute reservation of network resources that are dedicated to packet delivery. Guaranteed services are provided using the Resource Reservation Protocol (RSVP) discussed in Section 7.2.4. This service can be effectively achieved in closed networks. It can also be used in larger, open broadband networks that make up different service providers, for example when you search and browse the Internet. But in such cases, core routers would be needed to accept, maintain, and tear down numerous reservations. This behavior does not scale up with the number of nodes and traffic on the Internet.

Deciding which type of service (among best-effort, differentiated, and guaranteed) is appropriate to deploy in the network depends on several factors:

- The application or problem that the customer is trying to solve. Each of the three types of service is appropriate for certain applications. This does not imply that a customer must migrate to differentiated and then to guaranteed service (although many probably eventually will). A differentiated service—or even a best-effort service—might be appropriate, depending on the customer application requirements.

- The rate at which customers can realistically upgrade their infrastructures. There is a natural upgrade path from the technology needed to provide differentiated services to that needed to provide guaranteed services, which is a superset of that needed for differentiated services.
- The cost of implementing and deploying guaranteed service is likely to be more than that for a differentiated service.

7 MULTIMEDIA COMMUNICATION STANDARDS AND PROTOCOLS

The requirement for standards in communication should be obvious. During a transmission, the data might be communicated between two end devices, intermediary routers, data link devices, and physical media—all of which might not be in the same subnet and might have different manufacturers. For effective communication, well-defined bit streams with appropriate packet headers are necessary so that all involved devices and software interfaces can correctly parse packet traffic and take necessary actions. The OSI architecture is a high-level and general-purpose logical abstraction that aims at such standardization. Specific protocols are further necessary to instantiate the abstraction for defined communication requirements. A network protocol is a formal set of conventions, rules, and structures that govern how computers and computing devices exchange data over a network. Many technology vendors and research institutions formalize such protocols over years of development and testing. Some of these protocols are described in the following sections by grouping them into two large subsections—one that deals with general protocols and another that deals with multimedia-specific protocols, which address the synchronization, real-time, and other needs that are core to bursty multimedia traffic.

7.1 General Protocols

This subsection first discusses general protocols used in communication that do not necessarily take care of requirements imposed by real-time media.

7.1.1 Internet Protocol (IP)

The Internet Protocol (IP) is a Network layer protocol that represents the main methodology of communication over the Internet. IP has two primary responsibilities:

- To provide a connectionless method of data packet delivery and to do so on a best-effort basis. There is no guarantee of data reaching the destination.
- To provide fragmentation of large data into packets and its consequent reassembly when the packet traverses data links with different maximum transmission unit (MTU) sizes, for example ATM.

The IP addressing mechanism is indispensable to how routing occurs in this protocol. Each node on a network is assigned a unique address (32 bits in IPv4), which is further divided into two main parts—a network number and a host number. The network

number is a globally decided unique identifier by the Internet Network Information Center (InterNIC) if the network is to be part of the Internet and the host number is local to the network and decided by a local network administrator either automatically or statically. When data is communicated from a sender to a receiver, each packet contains both the source and destination IP addresses, which are used to forward the packets. Because of the connectionless nature of the forwarding, packets can arrive via different routes and, consequently, out of order. It is up to a higher-level protocol (for example, TCP, TCP/IP) to reorder them back together in the correct sequence.

7.1.2 Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) is one of the oldest and most robust protocols developed for reliable transmission. It is a connection-oriented transport layer protocol, which provides reliable data transfer from a sender to a receiver regardless of the size of the datagram or the packet. Although connection oriented, it is normally used in conjunction with the packet-switched connectionless IP to form TCP/IP (see section 7.1.4) to also reliably deliver data over connectionless networks such as the Internet. For a TCP connection, the sender node and receiver node allocate a buffer called a window, which is used to ensure data delivery. Flow control is established by limiting sending data in the window to the receiver. Once the receiver acknowledges the received data, the window is moved to the next set of bytes in the data.

The TCP header contains the source and destination ports, sequencing identifier, acknowledgement identifier, and other fields necessary for transmission and to ensure that the data is reliably received by the receiver. Unlike its counterpart—Universal Datagram Protocol, discussed next, which can start immediately sending data—TCP requires a connection to be established prior to sending data, and the connection must be terminated upon completion of transmission. To establish a connection, TCP uses a three-step agreement to open a TCP connection from a client to a server, the server acknowledges, and, finally, the client also acknowledges. This is followed by data transmission. Some of the salient features of data transmission are as follows:

- *Every packet has both the source and destination addresses*—This is needed for the source node to know where to send the data, and for the receiver node to know where to send acknowledgements.
- *In-order packet reception*—Each packet sent by the sender is given a sequence number that is tracked by the receiver and also acknowledged by the receiver by sending an ACK message back to the sender. If the sender does not receive an ACK message within a predefined time from the receiver, it retransmits the packet, assuming that the packet got lost or dropped en route. This sending and ACK reception is done for a group of packets at a time, which lie in the logical window described previously. Therefore, all sent packets are kept track of, and delivery is ensured.
- *Error-free data transfer*—Each packet has a checksum to verify with a high degree of certainty that the packet arrived undamaged, despite channel interference. If the calculated checksum of the received packet does not equal the transmitted packet's checksum, the packet is dropped and no ACK is sent.

- *Discarding duplicate packets*—If, for some reason, a packet is retransmitted because the sender did not receive an acknowledgement and the receiver receives duplicate packets, these can be detected by the in-order sequence number and discarded.

Although TCP ensures reliable transmission, the overhead of retransmission is normally a hindrance to maintaining real-time synchronization of multimedia data and, hence, typically not preferred for multimedia data communication. The User Datagram Protocol discussed next is the protocol of choice in such cases.

7.1.3 User Datagram Protocol (UDP)

The User Datagram Protocol (also termed as the Universal Datagram Protocol) is a connectionless protocol that sends each packet during a session along different routes. UDP does not provide the reliability and in-order data transmission as TCP does because data packets might arrive out of order or be dropped by the network and no acknowledgements are sent by the receiver. However, as a result, UDP is much faster compared with TCP and efficient for the transmission of time-sensitive data, such as in the case of synchronized multimedia data, and is normally a preferred Transport-layer protocol for multimedia data.

As shown in Figure 11-13, UDP is a minimal, message-oriented protocol that provides a simple interface between the Network layer below and the Session layer above it. If any sequencing is needed using UDP, a higher layer, such as the Application layer, has to augment logic to ensure the in-order delivery.

7.1.4 TCP/IP Family

A number of communication protocols fall collectively in this family—TCP over IP, UDP over IP, and so on. These are the main technical foundations of communication over the Internet, which make use of the connectionless packet communication protocols offered by IP and augment it with higher-level features depending on the needs of communication. For example, TCP itself is a reliable connection-oriented protocol to transfer data on a network; however, to deliver data on the connectionless Internet, it has to ride on IP and provide reliability, which includes in-order delivery and error-free delivery, with the IP protocol. Thus, this family of protocols has the commonality of using IP as a Network layer to communicate data while providing support from the higher layers. The TCP/IP families of protocols include general data transfer protocols, such as FTP, HTTP, IMAP, SNMP, SMTP, TELNET, and so on, as well as media-related protocols, such as HTTP, RTP, RTSP, RSVP, SCTP, and TRIP.

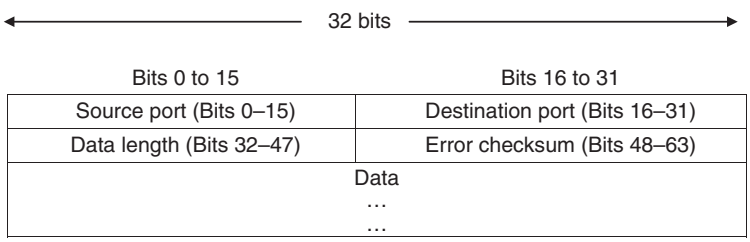


Figure 11-13 UDP header

7.2 Media-Related Protocols

In this section we enumerate a few commonly used protocols to access and stream media. Apart from the general protocol requirements such as those imposed by UDP or TCP, you also have media related requirements imposed by needs such as real time delivery, resource reservation requirements for guaranteeing a certain QoS level, identifying the media type that is being communicated and so on.

7.2.1 Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol is an application layer protocol. It provides the swiftness necessary to distribute media information without real-time needs. It has been in use by the WWW since the early 1990s and has become the de facto standard for transferring World Wide Web documents. HTTP works in a request-response manner between a client and a server. The originating client, typically a Web browser running on a computer, cell phone, PDA, or other end terminal, requests a document residing at a server location. The destination server, typically called a Web server, responds by locating the document and sending the information back to the client. The information might be individual files of any kind or a collection of text, images, and other media organized in the form of an HTML document. The client initiates the request by establishing a TCP connection. The request can go via many intermediary proxy servers, gateways, and tunnels to get to the destination Web server. Upon receiving a request, the destination server sends back a response with a header having a status line, either "*HTTP*/1.1 200 *OK*" or *Error*, and a body, which is either the requested data file or an error message regarding the data. HTTP defines many methods to initiate and respond between HTTP clients and HTTP servers as outlined in the following list:

- *GET*—The *GET* method specifies a request for a particular resource or file. This is the most common type of request method on the Internet.
- *HEAD*—The *HEAD* method also specifies a request similar to *GET*; however, it has no response body. This is useful to retrieve information such as metadata in the response header without the overhead to transport all the content.
- *POST*—The *POST* method submits user data from the server to the client with the data included as the body of the request.
- *PUT*—The *PUT* method uploads the data at the server side.
- *DELETE*—The *DELETE* method removes the data wrongly inserted by *PUT*.
- *TRACE*—The *TRACE* method is used to trace back the received request to the client. This is useful for the client to know how any intermediary servers are adding or changing the request if at all.
- *OPTIONS*—The *OPTIONS* method is used to return the HTTP capabilities of the server back to the client. This is used to check the Web server and its functions.
- *CONNECT*—The *CONNECT* method is used by intermediary proxy servers to tunnel the request through to the ultimate destination Web server.

7.2.2 Real-Time Transport Protocol (RTP) with RTCP

The original IP protocols used on the Internet provided delivery of data packets on a "best-effort" basis. Although this might suffice for non-real-time applications such as e-mail and FTP, it certainly is not suited to real-time multimedia applications. The Real-Time Transport Protocol (RTP) along with a monitoring Real-Time Transport Control Protocol (RTCP) were designed to solve the real-time needs for multimedia traffic, such as video and audio streams in applications, video-on-demand, videoconferencing, and other live streaming media. RTP was originally designed for multicast and supports distribution to multiple nodes. RTP runs on top of UDP, which provides efficient delivery over connectionless networks. It is preferably run over UDP rather than TCP because TCP's reliability achieved by in-order delivery of packets does not address the real-time needs of multimedia traffic if packets are lost or held up along the route. UDP, on the other hand, delivers packets without any heed to order. Though this is desirable, it does create problems when packets arrive out of order and destroy the stream synchronization. RTP, therefore, must create its own time stamping on each packet and sequencing mechanisms to help the receiver synchronize and render the information in sequence. Therefore, RTP needs the following additional parameters to be specified:

- *Payload identifier*—The payload, which refers to the packetized data, needs to be supplied with an identifier indicating the media type it contains as well as the encoding scheme. This helps the receiver in appropriately funneling data to the appropriate decoders and processing units when the application contains many different media streams.
- *Time stamp*—The time stamp records the timing information necessary to decode and play the media data in a proper timing order. Depending on the standardized compressed stream being used, this can include either or both of the DTS (Decode Time Stamp) and the CTS (Composition Time Stamp). The time stamp information is set by the sender depending on the information obtained by the higher layers, and helps not only intramedia, such as playing a video at the proper frame rate, but also intermedia synchronization, such as playing audio and video together.
- *Sequence number*—This feature is used to supplement the time stamp information. It is incremented by one for each RTP data packet sent, allowing the receiver to reconstruct the packets in the sender's original order when they arrive out of sequence.
- *SSRC ID*—This synchronization source identifier suggests the dependencies of the multiple media streams that are being sent by the sender and need to be synchronized—audio and video streams are a common example.
- *Optional CSRC ID*—This identifier, which identifies contributing sources, such as multiple cameras providing different streams in a videoconference, can be optionally specified.

An example of an RTP packet header incorporating this information is shown in Figure 11-14. Along with RTP, the RTCP supporting protocol controls and maintains the Quality of Service. RTCP works by initially creating a channel connection between the sender (server) and receiver (client), monitoring the data

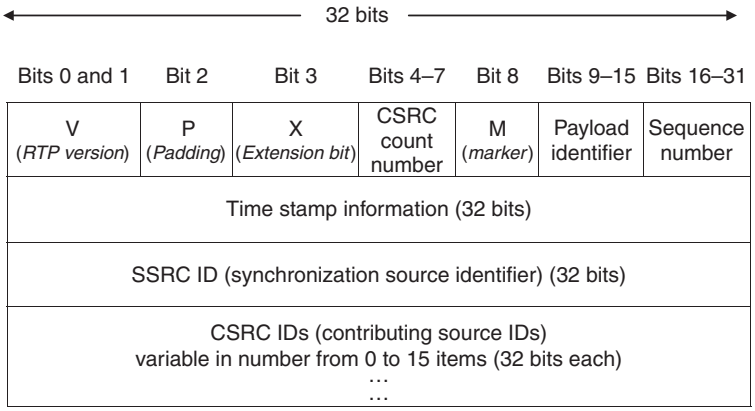


Figure 11-14 RTP packet header. 32-bit aligned header showing the semantics.

transmission and conveying information to the sender. Although the data packets sent by RTP and RTCP belong to the same session, they go to different ports at the same IP address.

7.2.3 Real-Time Streaming Protocol (RTSP)

The Real-Time Streaming Protocol (RTSP) was developed by the Internet Engineering Task Force (IETF) in 1998 and is useful in streaming media, as opposed to downloading the entire media content. In the fledgling days when bandwidths were low, the only option to transmit multimedia data was to download the entire content at the receiver's end, and then view the content, or have the receiver node set aside a buffer that, when filled to a certain extent, would allow uninterrupted playback. The buffer space required here can be large if a smooth playback is desired in the presence of low bandwidth and jitter. Nowadays, with broadband connections and higher compression standards, multimedia data can be viewed almost instantaneously by streaming the data.

RTSP is designed for streaming communication between a client and a stored media server and achieves real-time communication by issuing simple commands like play, pause, stop, and so on and, therefore, allows time-based access to files on a server. RTSP normally rides on top of RTP as the transport protocol for actual audio/video data. RTSP requests, which are sent over HTTP, include the following:

- *Describe*—A Describe request is sent to a server address, normally an RTSP URL, to obtain the presentation description such as media stream types (audio, video, graphics) and other particulars pertaining to the media, such as image size, frame rate, compression codec, and so on.
- *Setup*—A Setup request sent by a client imposes a transport protocol to communicate the media stream data, for instance RTP and RTCP. The Setup request also includes the port number for the transported media to be delivered. This has to be done prior to requesting and delivering any media

information. The server usually follows with a response confirming the chosen parameters and can fill any unspecified information in the request to standardized default values, for example receiving port number not specified. After setup and during a session, the client periodically sends an RTCP packet to the server, to provide feedback information on the current QoS in progress.

- *PLAY*—This request causes the media stream(s) from the stored server to be delivered to the client for playing.
- *PAUSE*—A *PAUSE* request temporarily stops the delivery of the media stream(s) so it can be resumed at a later time by another PLAY request.
- *RECORD*—This request allows media stream data to be sent from the client to the server for storage.
- *FAST FORWARD and REWIND*—These requests, as the names suggest, move the media state forward or backward in time.
- *TEARDOWN*—A *TEARDOWN* request terminates the real-time session that was established between client and the multimedia server and frees all session-related data and resources from the server.

An example illustrating the use of RTSP is shown in Figure 11-15.

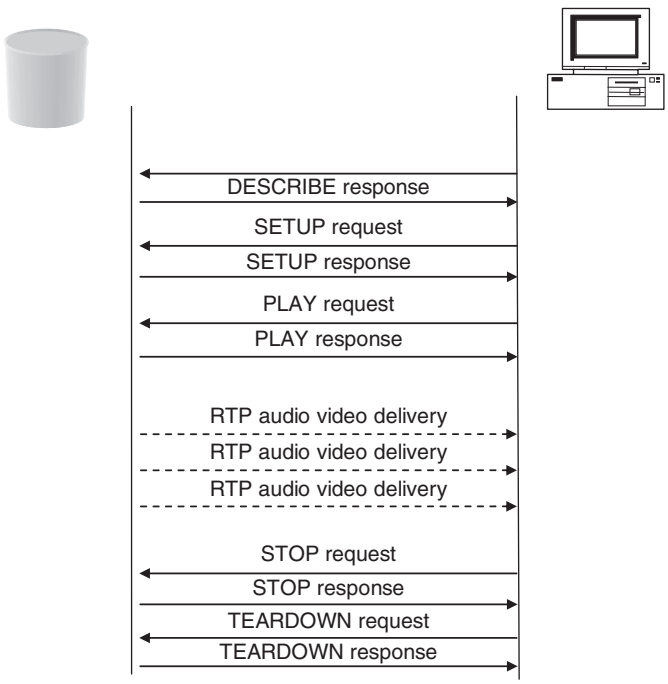


Figure 11-15 RTSP protocol in session. The session starts with "describe" and "setup" commands. Once "play" is requested, media data is delivered using the RTP protocol. The session is ended by a "teardown" request.

7.2.4 Resource Reservation Setup Protocol (RSVP)

The Resource Reservation Setup Protocol is a setup protocol where a receiver node reserves network resources during a communication session so as to efficiently monitor and guarantee the desired QoS for unicast and multicast multimedia communication over an integrated services Internet. The protocols defined previously, although they might monitor QoS such as in RTP, do not address control and rectification of QoS issues. RSVP works by applying two messages—a *Path* message and a *Resv* message. A *Path* message is initiated by the sender to the receiver(s) and contains information of the path and nodes along the path to the receiver. This is followed by one or more *Resv* messages from the receivers to reserve the intermediary node resources. Some of the main features of RSVP can be categorized as follows:

- RSVP requests resources for traffic streams in only one direction—from sender to one or more receivers. There can be multiple senders as well depending on how the application chooses to broadcast.
- In itself, RSVP is not a routing protocol, but works with the routing protocols to transmit data along allocated resources, which might be routers by themselves.
- In RSVP, the receiver in a transmission flow initiates and maintains the resource reservation for that flow by making RSVP a receiver-oriented protocol. A reservation might merge with an existing reservation when two receivers attempt to serve the same resource, for instance a router, for the same session in a multicast distribution. The merged reservation has to maintain the highest QoS needs among all merged reservations.
- RSVP maintains a *soft reserved state* for each resource, meaning the reservation at each node needs a periodic refresh from the receiver host. This is necessary for two reasons: One is to automatically adapt to dynamic changes in the network when nodes go offline, or routes are dismantled, and the second reason is to control and distribute the bandwidth so that one reserved node does not use up neighboring bandwidths.

RSVP provides various flavors or styles to allocate intermediate resources. These might fit different application QoS requirements and can also be extended as per needs. An example of RSVP in process is shown in Figure 11-16.

7.2.5 Telephony Routing over IP (TRIP) for Internet Telephony and VOIP

The Telephony Routing over IP is an administrative domain protocol that helps clients reach destinations between location servers. The primary function of a TRIP client, called a location server, is to exchange information with other location servers. The information includes the reachability of telephony destinations, information about intermediary gateways, and routing information. Additionally, TRIP can also be used to exchange attributes necessary to select correct routes based on gateway attributes. This is used to define synchronization and transport methodologies for the communication session. TRIP's operation is independent of any particular telephony-signaling protocol (SIP or H.323) and can, therefore, be used as a routing protocol or any of the signaling protocols. TRIP runs over a reliable transport layer protocol like TCP or UDP and, hence, does not need any support for error correction, packet defragmentation, and so forth.

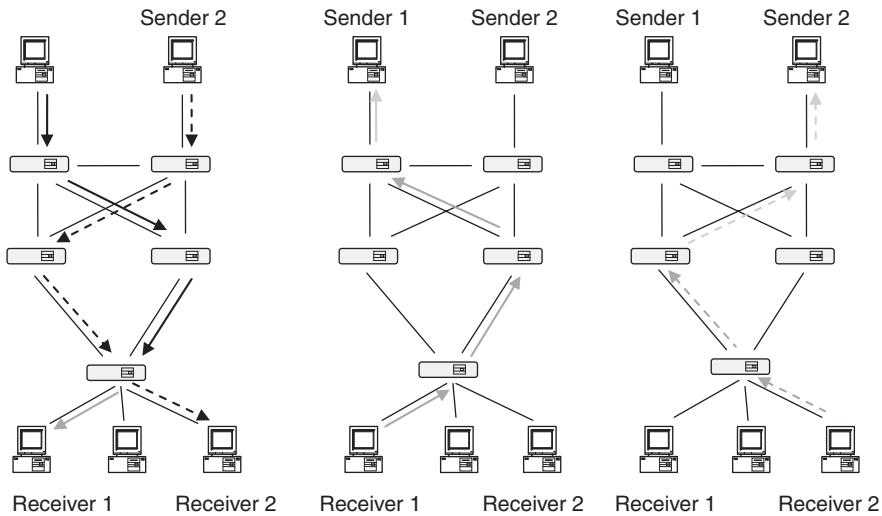


Figure 11-16 RSVP in process. Sender 1 and Sender 2 send PATH messages to Receiver 1 and Receiver 2 (Left). Receiver 1 sends a RESV message along the received path back to Sender 1 (Center). Receiver 2 sends a RESV message along the received path back to Sender 2 (Right). Both the PATH messages can be merged at intermediary nodes.

7.2.6 Session Initiation Protocol (SIP)

The Session Initiation Protocol is an Application-layer control protocol that has been set up to establish, modify, and terminate sessions in Internet telephony. Although designed for Voice over IP applications, it is not limited to it, and is also used for multimedia-related communications, such as videoconferences, multimedia distribution, and so forth. SIP transparently supports name mapping and redirection, which increases personal mobility so users can maintain an external visible identifier regardless of their network location. SIP is a client/server protocol where the sender client node connects to another receiver client node in a multimedia application by initiating a request to a server(s). The server(s) responds by address forwarding that ultimately reaches the receiver client. There are three types of servers—*proxy server*, whose job is to forward a client's call request, *redirect server*, whose job is to return the address of the next hop server, and *location server*, which finds the current location of the clients. The clients can invoke the following commands:

- *INVITE*—Invoked by a client (caller) to get other clients (callees) to participate
- *ACK*—Invoked by client (callee) to acknowledge an invitation
- *OPTIONS*—Invoked by client to inquire about media capabilities without establishing a session or call
- *BYE*—Invoked by client (caller or callee) to terminate a session
- *REGISTER*—Invoked by client (caller or callee) to send a user's location to the SIP server

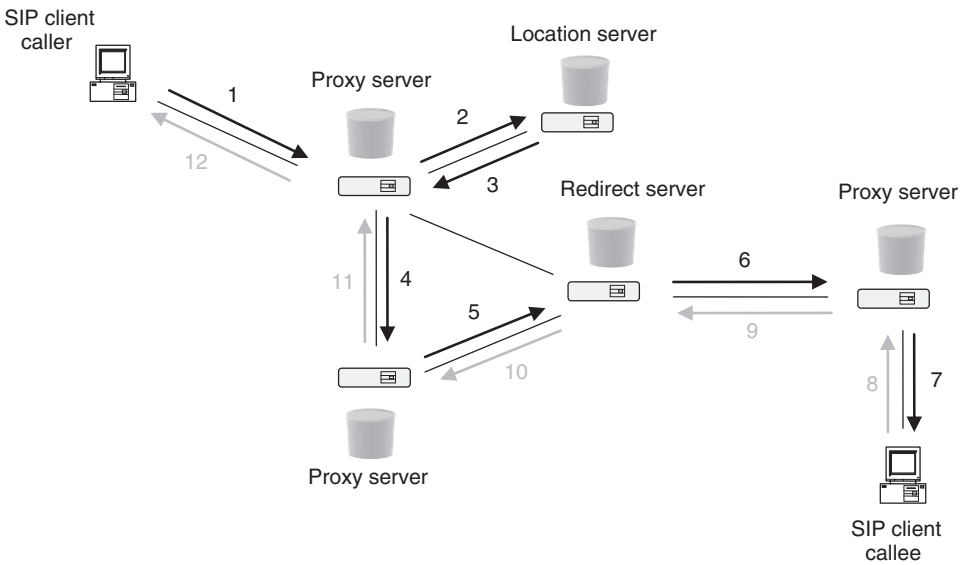


Figure 11-17 SIP example. The caller and callee clients are registered on the network user proxy servers. The client caller can reach another client to be called by tracing the registered locations through the network. The sequence traces of the INVITE message originating from the caller client are shown numbered in black, whereas the ACK message responses generated by the callee client are shown numbered in gray.

An example illustrating the usage of SIP is shown in Figure 11-17.

In a SIP session, callers and callees are given SIP addresses. Users can register their SIP addresses with SIP servers, which allows the flexibility for users to be anywhere on the network. When making a SIP call, a caller first locates the appropriate server and then sends a SIP *Invite*. The *Invite* reaches the callee via a chain of proxies and the callee sends an *Ack*. Prior to setting up the session, the caller can query for the media capabilities of the caller using the Options command. This can also be done via the Session Description Protocol described in the following section.

7.2.7 Session Description Protocol (SDP)

The Session Description Protocol describes a multimedia session that needs to be in session between a caller client and a callee client. A caller client must include this information when inviting the callee client during the initiation of communication, such as in the SIP *Invite* command. The description information includes the media stream types used in the session (audio, video, and so on) and its capabilities, destination address (unicast or multicast) for each stream, sending and receiving port numbers, stream type indicators, and other things necessary to communicate media information. When a caller client sends the SDP information in the *Invite* message, the callee client can choose to change it to fit its communication expectations.

8 EXERCISES

1. [**R02**] What are the advantages and disadvantages of the different MAC protocols when network access is controlled either by round-robin or time-sharing strategies and when network access is not controlled?
   - When does token ring outperform Ethernet?
   - What would be more useful for transmitting video or audio data?
   - Like reservation-based time division multiplexing, you could also think of frequency division multiplexing. How does this work and what are its advantages and disadvantages over time division multiplexing?
2. [**R03**] The following statements are either true or false. State them accordingly, giving reasons for your answers.
   - Connection-oriented networks are capable of doing better flow control compared with connectionless networks.
   - TCP is a better protocol for real-time media transfer compared with UDP.
   - In the OSI model, the most consistent and complete levels of agreement found among network manufacturers should be in the Transport and Network layers.
   - Connectionless networks are preferred for real-time multimedia communications compared with connection-oriented networks.
   - In connection-oriented networks, the number of total bytes transferred is less than that of connectionless networks.
   - Pay-per-view on cable television—which means only those who pay or agree to pay for a program can view the transmission—is a multicast protocol.
3. [**R02**] The following questions deal with issues that relate to Quality of Service (QoS) in different multimedia applications.
   - What does Quality of Service (QoS) mean?
   - Give three possible ways in which QoS is measured, briefly explaining each.
   - Consider these application scenarios of multimedia communication:
     Video-on-demand
     Teleconferencing

   What are the main differences in the Quality of Service requirements in these cases?
4. [**R02**] Transmission error is common in networked communication.
   - What is error rate and how is it measured? Mention why errors occur.
   - How does a receiver detect errors?
   - How does a sender detect errors?
   - What course of action can a sender/receiver have in a connection-oriented network to detect and correct errors? How does this change in a connectionless network?
5. [**R03**] Routing of data can be achieved via explicitly or implicitly setting up connections between a sender and receiver or via a completely connectionless manner.
   - What is the main difference between connectionless and connection-oriented networks?

- How is packet delivery achieved in both cases?
- Give a sample protocol used in the industry for both cases.
- Which is more reliable? Give reasons.
- Which type is usually used for real-time multimedia communications?
- Which one has a better ability to control network flow?

6. [**R03**] Network congestion refers to a state when a network throughput goes down due to multiple senders on a network simultaneously sending data.
   - What is meant by network congestion and how does the leaky bucket protocol try to control it?
   - When this protocol is used, you normally see a burst of packets/data followed by a steady stream of packets/data. A computer has a maximum packet size of 1000 bytes and produces data at 10 million bytes/sec. The bucket size is 1 million bytes and the network maximum throughput is 50 million bytes/sec. How long can a burst at maximum speed last?
7. [**R03**] The token bucket scheme limits the length of time a traffic burst can enter a network. The token arrival rate is 4×10^6 bytes/second and the maximum output data rate is 3232 megabits/second.
   - What bucket capacity is needed to limit the burst to 10 milliseconds?
   - Using this bucket capacity that you have computed, how long will it take the sender to send 2,010,000 bytes of data when the bucket is 25% full?
8. [**R04**] A computer accesses large video files over a widely distributed network from two computers *A* and *B*. With *A*, it uses a connectionless protocol to get the data, whereas with *B*, it uses a connection-oriented protocol. The path from a source computer node to the destination computer *B* is five hops long, that is, there are 5 links (with 4 intermediate switches). Each link has a transmission capacity of 1 Mbps (10^6 bits per second). For the following questions, ignore processing and queuing delays at the switches and propagation delays along the links. Also ignore any headers that are added to the message/packets.
   - A single 1 megabit message is sent from source to destination. How long will it take until it is fully received at the destination, that is, until the last bit of the message reaches the destination?
   - Now suppose that the message is broken up into 10 packets of 100 kilobits (or 10^5 bits) each. The packets are sent out from the source node one after the other, with no delay between one packet and the next. How long will it take until the last packet is fully received at the destination?
9. [**R04**] QoS requirements are needed to ensure a guaranteed network performance for applications.
   - What is QoS and how is it measured?
   - How can QoS be effectively implemented for large-band networks?
   - Is your method applicable to narrowband networks? If not, mention a few ways in which narrowband networks could implement QoS.

This page intentionally left blank

CHAPTER 12

Wireless Multimedia Networking

In the last decade, wireless communication has become a practical and reliable mode to transfer information between various devices. These wireless networks can be local, as used in homes, offices, metropolitan Wi-Fi networks, or larger cell phone communication networks. Overall, the ubiquitous emergence of portable information terminals, such as cell phones, Blackberry, and laptops, along with the growing wireless bandwidth has significantly altered the way we communicate and access information at work, at home, or just about anywhere. For example, cities are wirelessly connected using Wi-Fi networks and cell phones can be used over 3G networks to access videos, voice, and other communication services. Wireless Internet access is available at an ever-growing number of public places, such as airports, cafés, corporate campuses, university campuses, and so on. Wireless communications has become a large and integral part of information access in our lives.

Although wireless networking and information access might seem to be a relatively recent development today, wireless communication is, in fact, not a new concept, and is the same as radio communication, which has been around for more than a century. It is virtually the same technology that is responsible for television and radio broadcasts, except that it works bidirectionally. Radio communication is accomplished using specific lower frequencies of the radio spectrum that carry over larger distances. Active research began only 25 years ago to pursue usage of different frequencies of the infrared (IR) spectrum for smaller distance communication. The popularity of wireless communication systems can be understood by noting the advantages they have over wired systems. By far, the most important advantage is the mobility of the devices so that information can be accessed from anywhere within a coverage area. In addition, connecting becomes easier because there is no additional cost of rewiring, adding a communication device, or removing one. Devices can be installed anywhere without breaking

down walls or tearing up streets or sidewalks to lay down network cables. The bandwidths that wireless connections now provide give users mobility and flexibility to access informational services virtually anywhere a wireless link is available. Deploying many applications in large areas where information needs to be accessed is relatively easy using wireless technology, for instance traffic monitoring, security camera installations, and so on. Some of the applications that currently use wireless transmission include streaming video and audio, such as MP3, digital financial transactions, gaming, multimedia messaging services (MMS), long-distance learning, entertainment, and so on.

The preceding chapter dealt with various higher-level and lower-level protocols to access wired media and communicate information. This chapter begins by explaining how wireless communication fundamentally differs from wired communications in Section 1, and explaining the history of wireless communication development in Section 2. Section 3 goes on to explain the basics of wireless radio communications, including the different demarcations of the radio spectrum, the modulation techniques used in communication, and different methods to allocate wireless medium access for multiple users. Sections 4 and 5 summarize the different wireless standards, including the cellular/mobile phone generations (1G, 2G, 3G), the wireless LAN standards (the IEEE 802.11 family), and other wireless standards such as Bluetooth and cordless phones. Section 6 addresses common problems and hindrances to communication applications posed by wireless networks. Section 7 discusses Quality of Service factors in wireless communication. Finally, Section 8 concludes with a brief discussion of the 4G standard which is being designed to support global roaming across multiple wireless networks.

1 WIRELESS VERSUS WIRED TECHNOLOGY

The obvious difference between wireless and wired technologies is that the latter has direct/indirect physical connections set up between all the communicating components, whereas wireless technologies do not. Wireless components send and receive information along various frequencies using antennas. The only difference, then, relates to how various components end up accessing the medium and sending information across the medium. In wired technology, information is communicated by voltage switching on the physical interconnected wires. In wireless technologies, the information is modulated at different frequencies and transmitted. All the other higher-level protocols in the Open Systems Interconnection (OSI) architecture, TCP, UDP, HTTP, and so on, are very similar in both wired and wireless cases.

The most desired feature introduced by wireless communication technologies is mobility, where users can take their wireless devices across a coverage area and access information or enable voice and media communications. This locomotive feature introduces a few realities that are intrinsic to mobility. Some of these features include the following:

- Mobile elements such as cell phones, PDAs, laptops, and so on are less powerful than static desktops. Given any level of technological progress, it is almost certain that mobile elements need to be smaller and lighter than their static counterparts. This imposes restrictions on processor functionality,

heat dissipation, and overall ergonomics, which results in lower speeds and reduced memory and disk capacities.

- Furthermore, mobile elements face far larger variations in quality and reliability of connectivity. The bandwidths supported by wireless connections are much narrower compared with wired connections. This results in challenges to scale up bandwidth usage.
- Additionally, mobile processing elements have to deal with varying lower bandwidths with possible gaps in network coverage. Protocols need to be in place to ensure the best network connectivity or continuity of a communication session when going through such gaps.
- Mobile elements have limited power and energy resources. Mobile elements are normally powered using onboard batteries, whose constant consumption needs recharging. Many elements from hardware to software need to be tuned to minimize power consumption and last over larger periods.
- Most wireless devices, such as cell phones, need to pay a per-unit time charge to access or send data. The billing requirements to access information need close to real-time service. Delays incurred in downloading media are not acceptable or even possible, for instance it might take a few minutes to download a video on a mobile device, although it plays only for a few seconds.

Just like wired networks are classified into LANs and WANs as described in the preceding chapter, the introduction of wireless networks has resulted in more classifications. These classifications broadly revolve around their use individually or in combination with wired counterparts. The additional categorizations include personal area networks (PANs), wireless LANs (WLANs), and wireless PANs (WPANs). Figure 12-1 illustrates the categorization of these network definitions. A PAN is made up of a collection of secure connections between devices in a very local area. These devices typically include telephones and personal digital assistants in the area around one person. A PAN's reach is limited to a few meters. Examples of PANs are Bluetooth and Infrared Data Association (IrDA). These define physical specifications for intradevice communications and work by using established wireless communication protocols such as those based on IEEE 802.15, IrPHY (Infrared Physical Layer Specification), or IrLAP (Infrared Link Access Protocol) supporting up to 800 Kb/s. A wireless local area network (WLAN) is a wirelessly interconnected local area network that uses wireless radio waves to communicate data between nodes on the network. A wireless LAN has one or more wireless access points connecting the wireless users to the wired network, which is the main connection source to accessing nodes in wider areas. Depending on the number of access points, a wireless LAN's reach might range from a single room or a household to an entire campus. Communication protocols between nodes on a wireless LAN use the IEEE 802.11a, 802.11b, 802.11g, and other standards that support between 2 Mbps to 54 Mbps. These protocols are discussed in Section 4.2.

Apart from the mobility advantages, the infrared frequencies used to communicate indoors or outdoors provide several advantages over long-distance communication. Infrared emitters and receivers capable of high-speed communication are available at a

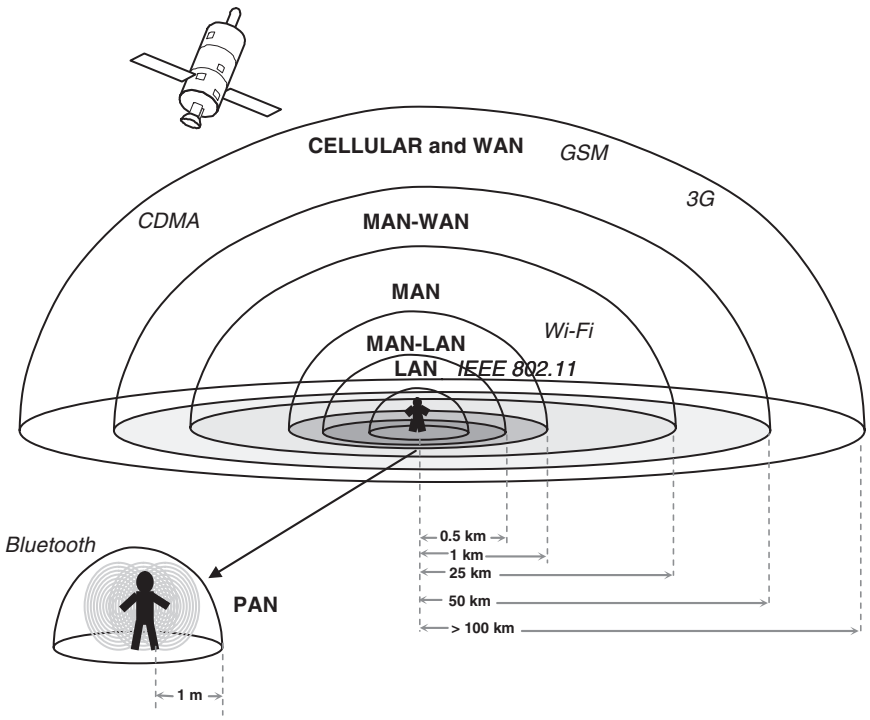


Figure 12-1 Categorization of wireless networks based on distances; personal area networks (PANs) are the smallest. Larger networks are established as the wireless coverage area increases—local area networks (LANs), metropolitan area networks (MANs), wide area networks (WANs), and cell phone networks. Each network has standardized communication protocols.

low cost. The frequency range used in communication is closer to visible light and exhibits qualitatively similar behavior. For instance, both penetrate through glass but not through walls and other opaque barriers. Both signal strengths dissipate in air over longer distances and, hence, remain confined to the local area from which they originate. As a result, several different wireless networks separated by opaque barriers or at sufficient distances apart can work on the same frequency.

2 HISTORY OF WIRELESS DEVELOPMENT

Wireless communication is accomplished by sending information via signals at frequencies that lie within the radio spectrum. Thus, radio communications and wireless communications are one and the same thing. Radio communication first started with the work of Guglielmo Marconi in 1896, when he developed his wireless telegraph apparatus.

Since then, radio communications have been used for radio, television, cordless phones, cellular phones, and numerous remote controls used to control consumer devices. The table in Figure 12-2 summarizes some of these developments.

Time	Description	Category	Signal
1896	Marconi invented the first wireless communication system, called a telegraph. In 1901, he successfully transmitted radio signals across the Atlantic.	Radio signal communication	Analog
1927	The first commercial radio telephone was made operational between the United States and the United Kingdom.	Voice over Radio	Analog
1946	The first car-based mobile telephone was developed in the United States.	Voice over Radio	Analog
1950	The first terrestrial radio telecommunication system was installed to support more than 2000 telephone circuits.	Voice over Radio	Analog
1950s	Numerous commercial mobile systems were deployed for the police, emergency calling, taxis, and so on. Also the first radio-based paging systems were developed.	Mobile telephony	Analog
1960s	The Improved Mobile Telephone Service (IMTS) was developed, which could transmit and receive in duplex mode with more channels.	Mobile telephony	Analog
	Telstar, the first communication satellite, was launched into orbit.		
1964	The International Telecommunications Satellite Consortium (INTELSAT) was established with the first geostationary satellite launched in 1965.	Mobile telephony	Analog
1970s	The ARPANET was developed for digital networked communication using packet switching technology.	Digital network	Digital
1977	The Advanced Mobile Phone Service (AMPS) was invented by Bell Labs. This was installed in different geographic regions, now called cells.	First-generation wireless	Analog
1979	Total Access Communication System (TACS) Nordic Mobile Telephony (NMT), a variant of AMPS, was introduced in Europe. It was later replaced by GSM.	First-generation wireless	Analog
1983	TCP/IP became the protocol of choice for communication.	Digital network	Digital
1990	Motorola launched communication satellites (Iridium System) for mobile communications. network	Mobile telecommunication	Analog and Digital

Figure 12-2 Evolution of wireless communications (Cont.)

1992	Global System for Mobile Communications (GSM) was launched in Europe and Asia.	Second-generation wireless	Analog and Digital
1995	The Personal Communication Service (PCS) spectrum spanning 1.7 to 2.3 GHz was licensed by the FCC for commercialization.	Second-generation wireless	Analog and Digital
1998	The cellular phone industry made advancements; Qualcomm deployed CDMA technology for digital cellular communication.	Second-generation wireless	Analog and Digital
Late 1990s	Personal area networks (PANs) became popular (Bluetooth). IBM, Ericsson, Nokia, Toshiba, and others joined together to develop Bluetooth technology for wireless data exchange among components in a PAN.	Mobile personal area networks	Digital
2000	The 802.11 family of wireless networks gained popularity.	Mobile PC networks	Digital
2002+	3G cell phone networks now support a wide array of multimedia services using W-CDMA.	Third-generation wireless	Digital

Figure 12-2 (Continued)

3 BASICS OF WIRELESS COMMUNICATIONS

This section explains the basics of radio communication using the infrared spectrum. Section 3.1 explains radio waves, propagation, and radio frequency allocation. Section 3.2 defines the various modulation techniques used to communicate analog and digital information via radio signals. Section 3.3 discusses the major multiplexing techniques used to control how multiple users can access the frequency spectrum.

3.1 Radio Frequency Spectrum and Allocation

Wireless communication is achieved by sending information along frequencies that lie in the radio frequency spectrum. The entire electromagnetic spectrum is shown in Figure 12-4, which can be divided into the radio (infrared), visible, and ultraviolet. The radio frequency (RF) refers to that portion of the electromagnetic spectrum below 300 GHz that can be generated and modulated using electromagnetic principles and subsequently transmitted or captured using antennas. The RF spectrum can be large but has been further subdivided to carry different types of signals—radars, AM/FM radio, television, cell phones, GPS receivers, garage door openers, and so on—as shown in the table in Figure 12-3. The governments of countries in any geographic area divide and allocate this spectrum to prevent interference between different devices. For instance, in the United States, the FCC (Federal Communications Commission) decides

Frequency (wavelength)	Name of spectral band	Examples
< 30 Hz (>10,000 km)	Extremely Low Frequency (ELF)	Underwater communications, for example, submarines
30–300 Hz (10,000 km–1000 km)	Super Low Frequency (SLF)	Underwater communications, for example, submarines
300–3000 Hz 1000 km–100 km	Ultra Low Frequency (ULF)	Communications at tunnels (for example, mines, cave explorations)
3–30 KHz 100 km–10 km	Very Low Frequency (VLF)	Very Low Bandwidth communications
30–300 KHz 10 km–1 km	Low Frequency (LF)	AM radio—long wave, K-waves
300–3000 KHz 1 km–100 m	Medium Frequency (MF)	AM radio—medium wave
3–30 MHz 100 m–10 m	High Frequency (HF)	Amateur radio/short wave
30–300 MHz 10 m–1 m	Very High Frequency (VHF)	FM radio and television
300–3000 MHz 1 m–100 mm	Ultra High Frequency (UHF)	Television, mobile phones (FDMA, TDMA), wireless LAN, airplane communications
3–30 GHz 100 mm–10 mm	Super High Frequency (SHF)	Mobile phones (W-CDMA), microwave devices, radars
30–300 GHz 10 mm–1 mm	Extremely High Frequency (EHF)	Astronomy

Figure 12-3 Allocation of the radio frequency spectrum

who—whether private or public groups—can use which frequencies, and also for which purposes. You can see from the table in Figure 12-3 that, although the radio spectrum is large, the part allocated for mobile communications across cell phones, wireless LANs, and so on is merely 1.5–4.0 GHz, which is fairly narrow.

3.2 Radio-Based Communication

Given an allocated radio spectrum, either analog or digital information can be transmitted across it. For example, most radio and TV stations transmit analog information, whereas most cell phone wireless networks are now digital. The information is transmitted on a radio frequency called the carrier frequency or wave. This carrier wave has much higher frequency than the information itself. It is usually a sinusoid wave, whose properties, such as the amplitude, change depending on the encoded

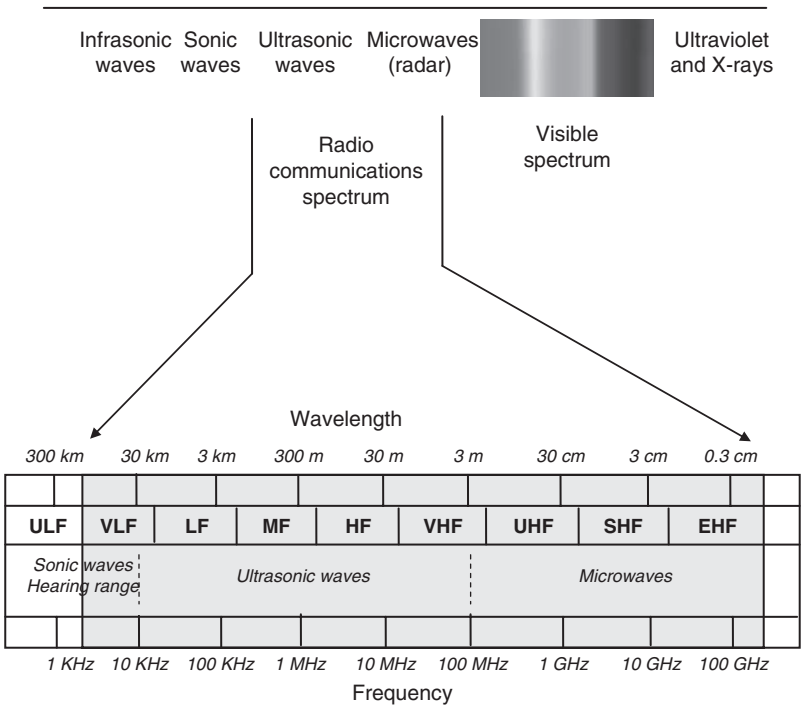


Figure 12-4 Radio frequency spectrum. The frequency range 3 KHz to 300 GHz corresponds to the radio spectrum used in wireless communications.

information. The change is known as modulation and the information signal must be modulated onto the carrier wave. The receiver demodulates the carrier to extract the information signal. Modulation is carried out in different ways depending on whether the information signal is analog or digital. Some of these methods are described in the following list and illustrated in Figures 12-5 and 12-6:

- *Amplitude modulation (AM)*—This is an analog modulation technique where the amplitude of the carrier is modified to encode the signal.
- *Frequency modulation (FM)*—This is an analog modulation technique where the frequency of the carrier is modified to encode the signal.
- *Amplitude-shift keying (ASK)*—This is a digital modulation technique where a "0" bit indicates the absence of the carrier (no signal or flat signal) and a "1" bit indicates the presence of the carrier with some constant amplitude.
- *Frequency-shift keying (FSK)*—This is a digital modulation technique where a "0" bit is the carrier at a certain frequency and a "1" bit is the carrier at a different frequency. Each of the frequencies has the same constant amplitude.
- *Phase-shift keying (PSK)*—This is a more commonly used digital modulation scheme that conveys data by changing the phase of the carrier wave. The frequency or amplitude of the signal does not change, but as the bit changes, the phase is either changed or remains constant.

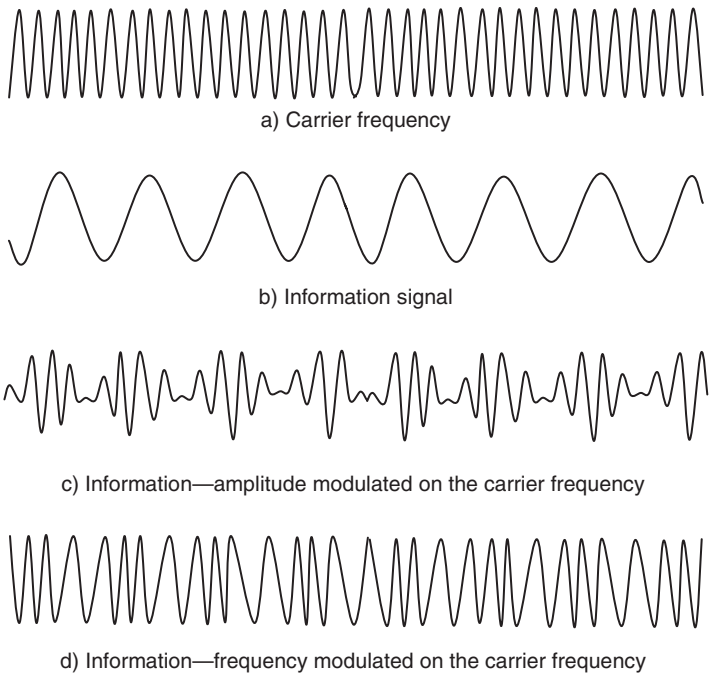


Figure 12-5 Analog modulation techniques. Information (signal b) to be communicated is conveyed by modulating it on a carrier frequency (signal a). The carrier frequency's amplitude can be modulated (signal c) or the frequency can be modulated (d).

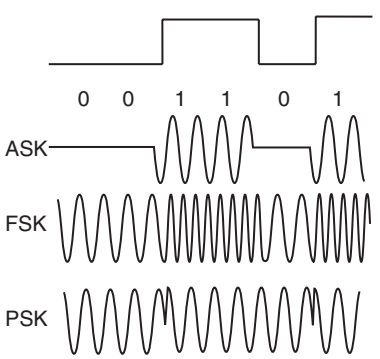


Figure 12-6 Digital modulation techniques. Digital information (top) to be communicated in three ways: Amplitude-shift keying (ASK) works by transmitting a 0 as no signal and transmitting a 1 by some predefined frequency. Frequency-shift keying (FSK) works by using two different frequencies for 0 and 1. Phase-shift keying (PSK) works by using the same predefined frequency but different phases for 0 and 1.

Thus, all modulation schemes convey information by changing some property of the carrier wave—amplitude, frequency, or phase. Other methods of modulation are based on the variants of these basic modulation schemes. Examples include quadrature amplitude modulation (QAM), which uses four different amplitudes, binary phase-shift keying (BPSK), which uses two different phases, and quadrature phase-shift keying (QPSK), which uses four different phases. Now that we know the spectral allocation for wireless communication, and how information is transmitted along a frequency, next we need to discuss how to control user access of this spectrum.

3.3 Medium Access (MAC) Protocols for Wireless

In physically connected networks, data is transmitted along a physical medium, such as a twisted wire, RS232, and so on. MAC protocols allow nodes on the network to have control of access to the physical medium. The token ring and Ethernet with CSMA/CD are examples discussed in the preceding chapter that illustrate MAC protocols. In case of mobile or wireless communications, data exchange is accomplished by sending information on a limited interval of frequency spectrum. For instance, the GSM networks limit their spectrum to 1.85–1.91 GHz for uplink and 1.93–1.99 GHz for downlink. Other protocols such as the IEEE 802.11b use 2.4–2.479 GHz. Although this spectrum is used for local area communications, this is clearly not a lot of spectrum for multiple users. For example, consider hundreds of cell phone users around a local cell area, or many computers in a WLAN. Given that the available communication spectrum is limited, there are three basic approaches used to carve the spectrum and allow multiple users to access it:

- Divide the spectrum by frequency (FDMA).
- Divide the spectrum by time (TDMA), which is often used along with FDMA.
- Divide the spectrum by code or usage pattern (CDMA).

3.3.1 Frequency Division Multiple Access (FDMA)

Frequency Division Multiple Access or FDMA is most commonly used in the first-generation wireless standards. It is an analog-based communication technology where the spectrum or communication band is divided into frequencies, known as carrier frequencies. Only a single subscriber is assigned to a carrier frequency. The frequency channel is, therefore, closed to all other users until the initial call is finished or the call is handed off to a different channel. Each channel is used in a one-way communication. To support a full-duplex mode, FDMA transmission requires two channels—one for sending and one for receiving. This is also known as Frequency Division Duplex (FDD). Figure 12-7 illustrates how the frequency spectrum is divided into multiple channels.

FDMA is, thus, a very simple way of having multiple users communicate on the same spectrum and have no framing or synchronization requirements during continuous transmission. However, the simplicity has drawbacks, for instance, if a channel is not in use, it sits idle. The channel bandwidth is also relatively narrow (25–30 KHz) and tight filtering mechanisms are needed to minimize interchannel interference. For example, the Advanced Mobile Phone Service (AMPS) uses 824–849 MHz for mobile-based cell communication and 869–894 MHz for base-mobile communication. Each 25 MHz band,

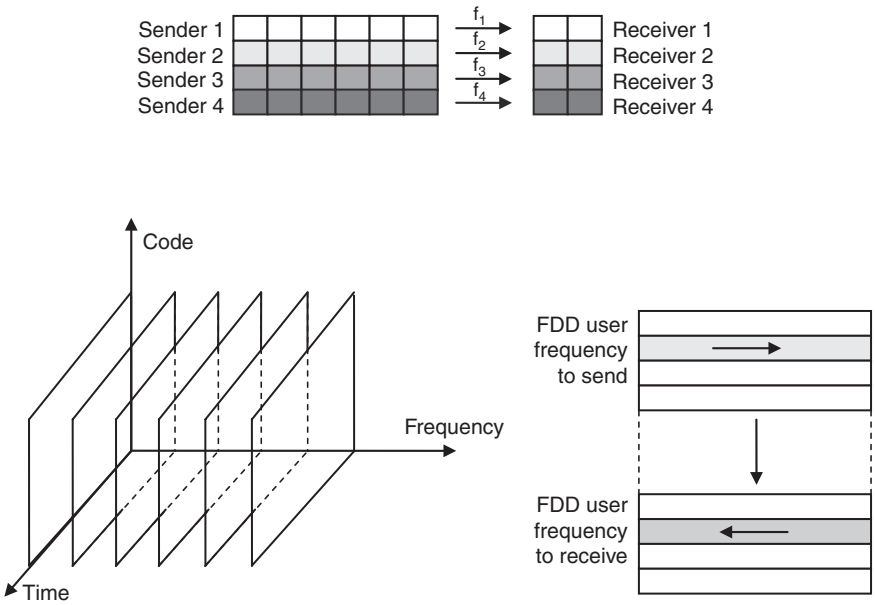


Figure 12-7 FDMA example. Each user sends data to a receiver on a different frequency (top). If a user has to have a two-way communication (FDD), two channels are allocated by the user, one to send and one to receive (lower right). The lower-left figure illustrates how the communication space can be viewed as slots of frequency channels.

therefore, is allocated for single-mode communication. Further, each 25 MHz band is divided into 416 slots or channels resulting in each channel having a 30 KHz band.

Another variation of FDMA, commonly used in wireless local area networks, is orthogonal frequency division multiplexing access (OFDMA). In FDMA, if the channel boundaries are close together, interchannel interference can result in degraded signal quality. OFDMA takes the concept further where users on different channels transmit at orthogonal frequencies where the carrier frequency undergoes a phase shift compared with the neighboring carriers.

One important aspect of frequency-based communication is that carrier frequency fades across distances from a transmitting base station, thus limiting the distance over which the communication is possible. The coverage is limited to a small geographical area, typically called a cell. Many cell units interconnect and reuse the frequency space. The working of cellular networks, its advantages, and problems are explained in Section 4 and 6 respectively.

3.3.2 Time Division Multiple Access (TDMA)

Time Division Multiple Access (TDMA) improves the overall usage of the spectrum compared with FDMA and is the basis for most second-generation mobile communication standards. Unlike in FDMA, where only one user can communicate over a carrier frequency, TDMA works by allowing multiple users to communicate on the same

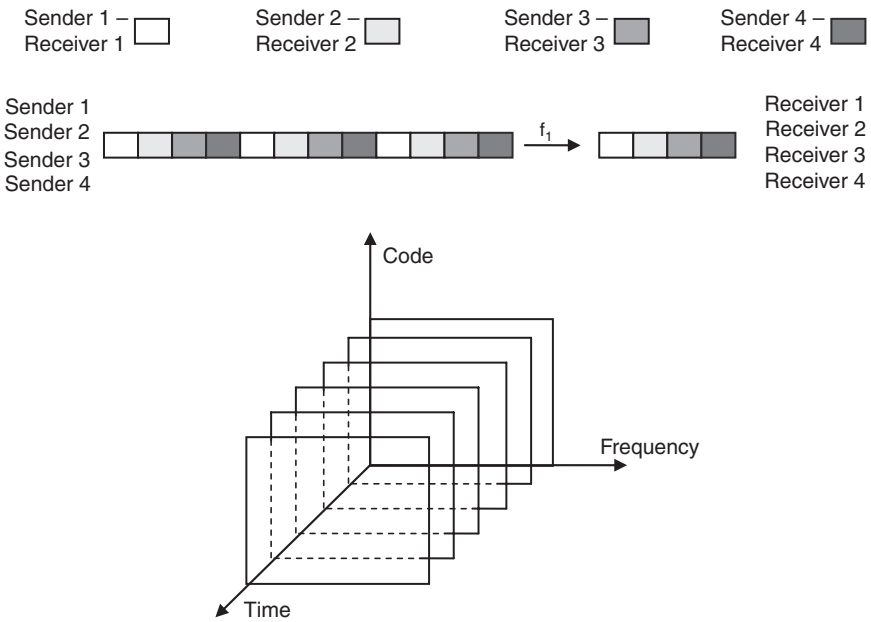


Figure 12-8 TDMA illustration. Senders and receivers communicate on the same frequency but at different times (top). The lower-left figure illustrates how the communication space can be viewed as slots of time channels for a given frequency.

carrier frequency in different time slots. At a given time, only one user is allowed to transmit (or receive) and several users can transmit in their own time slots, thus sharing the same carrier frequency to communicate. Figure 12-8 illustrates TDMA. The base station continuously and cyclically switches from user to user on the channel. Duplex communication to send and receive on a single channel can also be used by a sender sending in one time slot, but listening and receiving on the same channel at another time slot. This is known as Time Division Duplex (TDD).

Although TDMA increases the number of users that can simultaneously communicate across a spectrum, it does have disadvantages. Multiple users on different slots increase the slot allocation complexity and mandate synchronization. Another disadvantage of TDMA is that it creates interference at a frequency, which is directly connected to the time slot length, with other devices. This can result in an irritating buzz, which can be heard when a cell phone is near a radio or speakers. From a functional standpoint, TDMA has many advantages. For example, a mobile phone needs only to listen (and broadcast) during its time slot. This discontinuous transmission allows safe frequency handovers (discussed in Section 6.4). Also, during the nonactive slots, the mobile device can carry out other tasks, such as network sensing, calibration, detection of additional transmitters, and so on. Time slots can be requested and assigned on demand, for instance, no time slot is needed when a mobile device is turned off, or multiple slots can be assigned to the same device if more bandwidth is needed.

TDMA is usually used in conjunction with FDMA. Here, a group of senders and receivers communicate using TDMA on a certain specific frequency f_1. The same might be repeated for a different group of senders and receivers using another frequency f_2. Figure 12-9 illustrates a sample usage of TDMA in conjunction with FDMA for eight senders using four different frequencies.

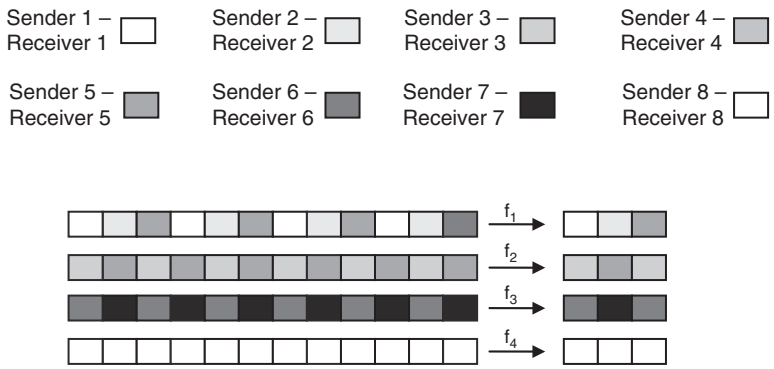


Figure 12-9 TDMA used in conjunction with FDMA. Four different frequencies are shown; each is time multiplexed. f_1 is used by senders 1, 2, and 3. f_2 is used by senders 4 and 5. f_3 is used by senders 6 and 7, whereas f_8 is used solely by sender 8.

3.3.3 Code Division Multiple Access (CDMA)

Code Division Multiple Access (CDMA) is based on the spread-spectrum technology, where the bandwidth of a signal is spread before transmitting. Spread-spectrum techniques are methods where the input signal at a certain frequency known as the base frequency is spread over a wider band of frequencies. This spread signal is then broadcast, making it indistinguishable from background noise and, thereby, allowing greater resistance to mixing with other signals. Hence, by design, this technique has the distinct advantages of being secure and robust to interference. Spread spectrum has been known since the 1950s where it was initially used for analog signals, and, consequently, used in some of the cordless and analog cellular phones. Digital signal voice/data applications using spread spectrum, in particular CDMA, have gained popularity in the wireless communications industry. CDMA increases the spectrum bandwidth by allowing users to occupy all channels at the same time. This is unlike FDMA/TDMA, where only one channel is used either at a single time or in a multiplexed manner.

3.3.3.1 Frequency Hopping

As described earlier, radio signals are transmitted at a particular frequency, known as the carrier frequency using modulation techniques. Frequency hopping is a spread-spectrum method of transmitting the signal by rapidly switching the carrier frequency among various higher frequencies using a pseudorandom sequence known both to the transmitter and receiver. If f_c is the carrier signal typically in KHz, and f_r is the random spread frequency typically in MHz, then the transmitted frequency is $f = f_c + f_r$. The pseudorandom variation causes f to

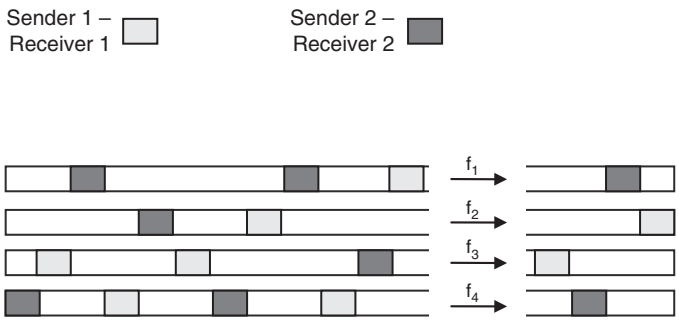


Figure 12-10 Frequency hopping spread spectrum. Two senders are shown. Each sender communicates by spreading its carrier frequency among the different frequencies f_1, f_2, f_3, f_4 randomly.

change depending on f_r. The receiver knows the carrier frequency f_c and the pseudorandom f_r and, hence, knows what frequencies to listen to recover the original. Figure 12-10 illustrates an example where two senders and receivers communicate using the frequency hopping method.

One problem with frequency hopping occurs when multiple users transmit data using frequency hopping. With more users hopping channels in a random/pseudorandom manner, it is possible that two or more users might collide on a channel at the same time, creating undesirable interference. Time multiplexing solutions might solve this problem, but these need transmission management, and quickly impose a limitation on the number of simultaneous users. Another undesirable feature with this frequency hopping technique is the need for sophisticated frequency synthesizer bank circuitry at the sender and frequency filter banks at the receiver to decipher the signal. Both increase the cost of the devices needed to effectively communicate between senders and receivers using spread spectrum. The direct sequence method discussed next helps alleviate all these problems.

3.3.3.2 Direct Sequence

With direct sequence (DS), multiple users can simultaneously make use of the entire shared wideband channel during the period of transmission. It is not as demanding on the electronic equipment as frequency hopping. DS works by giving each user a unique, high-speed spreading code, which gets applied as a secondary modulation. These codes are called chipping codes, and each receiver knows its chipping code. A sender sending to a specific receiver spreads the signal using the receiver's specific chipping code. The receiver is able to retrieve the signal by making use of its specific code. Mathematically, this can be illustrated using the following formulation.

Assume two senders A and B send signals S_A and S_B respectively. They use codes C_A and C_B to spread their spectrum. Each sender will create a spread signal T_A and T_B, which is going to be transmitted. The spread signals T_A and T_B are created by the mathematical operator "$\cdot$". On the receiving side, after T_A and T_B are received, the same operator is used with the receiver's code to reconstruct the signal. During the

transmission process it is natural to expect that the two signals will interfere if they are simultaneously sent. However, using this method the receiver will be able to recover and reconstruct the sent signal at its side as long as the choice of codes is *orthogonal*, that is codes C_A and C_B are such that $C_A \cdot C_B = 0$ while $C_A \cdot C_A = 1$ and $C_B \cdot C_B = 1$.

For sender A, signal being sent $= S_A$
Transmitted signal $T_A = S_A \cdot C_A$
For receiver with chip code C_A *the reconstructed signal*

$$\begin{aligned} RS_A &= T_A \cdot C_A \\ &= (S_A \cdot C_A) \cdot C_A \\ &= S_A \qquad (C_A \cdot C_A = 1) \end{aligned}$$

For sender B, signal being sent $= S_B$
Transmitted signal $T_B = S_B \cdot C_B$
For receiver with chip code C_B *the reconstructed signal*

$$\begin{aligned} RS_B &= T_B \cdot C_B \\ &= (S_B \cdot C_B) \cdot C_B \\ &= S_B \qquad (C_B \cdot C_B = 1) \end{aligned}$$

If the two senders send the signals at the same time, the physical properties of interference say that if two signals at a point are in phase, they will combine constructively or "add up" and if they are out of phase, they will combine destructively or "subtract out," resulting in a difference of amplitudes. Digitally, this phenomenon can be modeled as the addition of transmitted vectors. So if we have two senders sending signals S_A and S_B simultaneously using codes C_A and C_B, respectively, the transmitted vectors T_A and T_B will sum up to form $T_A + T_B$. Now, suppose a receiver receives $(T_A + T_B)$, and has a chip code C_A. The reconstructed signal is recovered by a similar process as $(T_A + T_B) \cdot C_A$. Similarly, at the second receiver, the signal is reconstructed as $(T_A + T_B) \cdot C_B$.
This can be expressed as follows:

$$\begin{aligned} \textit{reconstructed } S_A &= (T_A + T_B) \cdot C_A \\ &= (S_A \cdot C_A + S_B \cdot C_B) \cdot C_A \\ &= S_A \cdot C_A \cdot C_A + S_B \cdot C_B \cdot C_A \\ &= S_A \qquad (C_A \cdot C_A = 1 \;\&\; C_A \cdot C_B = 0) \\ \textit{reconstructed } S_B &= (T_A + T_B) \cdot C_B \\ &= (S_A \cdot C_A + S_B \cdot C_B) \cdot C_B \\ &= S_A \cdot C_A \cdot C_B + S_B \cdot C_B \cdot C_B \\ &= S_B \qquad (C_B \cdot C_B = 1 \;\&\; C_A \cdot C_B = 0) \end{aligned}$$

Figure 12-11 illustrates an example on a real signal. The input signals S_A and S_B are defined to 1011 and 1010, respectively. Each sender uses a different chip code to

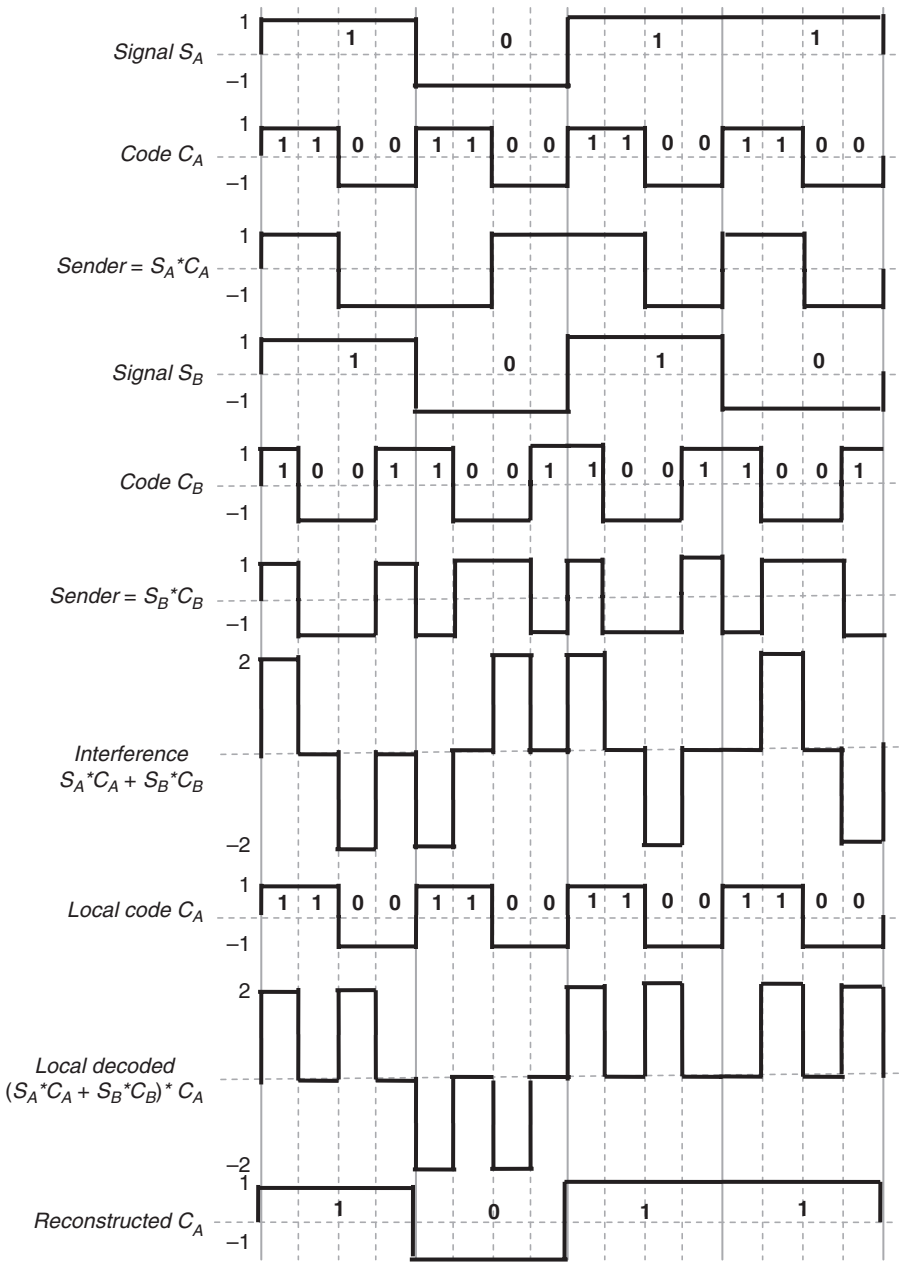


Figure 12-11 Direct-sequence spread spectrum. Signals S_A and S_B are spread using codes C_A and C_B. These signals might interfere when transmitted, but at the receiving end they can be reconstructed as long as C_A and C_B are orthogonal.

modulate its signal. For example, S_A is modulated by the code $C_A = (1,1,-1,-1)$, where a 1 bit in S_A is represented by C_A and a 0 bit in S_A is represented by a $-C_A$. This would make the transmitted signal T_A transform to $(C_A, -C_A, C_A, C_A)$, which is the same as (1, 1, −1, −1, −1, −1, 1, 1, 1, 1, −1, −1, 1, 1, −1, −1), also called the transmitted vector. Computing the transmitted vector can be formulated as a two-step process where S_A is first transformed to (1, −1, 1, 1) by keeping a 1 and replacing a 0 by a −1. (1, −1, 1, 1) is then transformed to the transmitted vector T_A by a time-based dot product $S_A * C_A$. The second sender in Figure 12-11 needs to transmit $S_B = 1010$ and uses a chip code $C_B = (1, -1, -1, 1)$. Note that the two codes C_A and C_B are not arbitrary but are chosen such that they are orthogonal because their dot product $C_A \cdot C_B = (1 \times 1) + (1 \times -1) + (-1 \times -1) + (-1 \times 1) = 0$. As shown, S_B can be modulated by C_B to create the transmitted vector T_B as (1, −1, −1, 1, −1, 1, 1, −1, 1, −1, −1, 1, −1, 1, 1, −1).

When a receiver receives T_A, it can re-create S_A at its end if it knows C_A. This can be done by taking the dot product of T_A and C_A for each coded section, to produce the received vector $R_A = T_A \cdot C_A$. Because T_A is represented as $(C_A, -C_A, C_A, C_A)$, $T_A \cdot C_A = (C_A \cdot C_A, -C_A \cdot C_A, C_A \cdot C_A, C_A \cdot C_A) = (1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1)$. Because $C_A \cdot C_A = 1$ and $-C_A \cdot C_A = -1$, the reconstructed signal RS_A can be written as (1, 0, 1, 1) the same as S_A. Similarly, $RS_B = (1, 0, 1, 1)$ can be reconstructed from recovered vector $R_B = T_B * C_B$. The reconstruction process can alternatively be visualized as a part wise averaging process of the recovered vectors R_A and R_B. The first bit of RS_A is reconstructed by averaging the first 4 bits (or n bits, where n is the length of C_A). The next bit is reconstructed by averaging the next 4 bits. The various signal values are shown worked out in the following illustration, while Figure 12-11 shows the plotted values.

$$S_A = (1\ -1\ 1\ 1) \qquad S_B = (1\ -1\ 1\ -1)$$
$$C_A = (1\ 1\ -1\ -1) \quad C_B = (1\ -1\ -1\ 1)$$

$$T_A = S_A \cdot C_A = (1, 1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, 1, 1, -1, -1)$$
$$T_B = S_B \cdot C_B = (1, -1, -1, 1, -1, 1, 1, -1, 1, -1, -1, 1, -1, 1, 1, -1)$$

At Receiver1:

$$R_A = T_A \cdot C_A = (1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1)$$
$$RS_A = (avg(1,1,1,1), avg(-1,-1,-1,-1), avg(1,1,1,1), avg(1,1,1,1))$$
$$RS_A = (1, -1, 1, 1)$$

At Receiver2:

$$R_B = T_B \cdot C_B = (1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1)$$
$$RS_B = (avg(1,1,1,1), avg(-1,-1,-1,-1), avg(1,1,1,1), avg(-1,-1,-1,-1))$$
$$RS_B = (1, -1, 1, -1)$$

With Interference–

$$T_A + T_B = (2, 0, -2, 0, -2, 0, 2, 0, 2, 0, -2, 0, 0, 2, 0, -2)$$

At Receiver1:

$$R_A = (T_A + T_B) \cdot C_A = (2, 0, 2, 0, -2, 0, -2, 0, 2, 0, 2, 0, 0, 2, 0, 2)$$
$$RS_A = (avg(2, 0, 2, 0), avg(-2,0,-2,0), avg(2, 0, 2, 0), avg(0, 2, 0, 2))$$
$$RS_A = (1, -1, 1, 1)$$

At Receiver2:

$$R_B = (T_A + T_B) \cdot C_B = (2, 0, 2, 0, -2, 0, -2, 0, 2, 0, 2, 0, 0, -2, 0, -2)$$
$$RS_B = (avg(2,0,2,0), avg(-2,0,-2,0), avg(2,0,2,0), avg(0, -2, 0, -2))$$
$$RS_B = (1, -1, 1, -1)$$

In the example shown in Figure 12-11, we made use of two orthogonal chip codes C_A and C_B. Because each chip code is made up of 4 bits, there can be more chip codes that follow the same orthogonal property. For instance, along with codes (1,1,−1,−1) and (1,−1,−1,1), the chip codes (1,1,1,1) and (1,−1,1,−1) form an equivalence class where each code is orthogonal to every other code. Such a class of codes is constructed from the columns (or rows) of Walsh matrices. Walsh matrices are square matrices that are recursively constructed as shown here:

$$W(1) = 1$$

$$W(2) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$W(4) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

$$W(2^k) = W(2) \otimes W(2^{k-1}) = \begin{bmatrix} W(2^{k-1}) & W(2^{k-1}) \\ W(2^{k-1}) & -W(2^{k-1}) \end{bmatrix}$$

The columns (or rows) of the matrix form the orthogonal chip codes. The 4×4 matrix generated for $k = 2$ suffices for this example, which can have four unique users. You can see that with higher orders of k, many more codes can by obtained such as in the case needed to support multiple users in a mobile network. However, as the length increases, so does the bandwidth needed to communicate a signal because each bit in the signal is represented by as many bits as present in the chip code. Moreover, to use the chip code in a synchronous manner as shown previously, a mobile base station needs to coordinate so that each sender transmits their signals with the assigned chip code *at the same time*, which is hard to do with multiple users. Because in a practical transmission, all users cannot be precisely coordinated, especially because mobile senders are moving from one cell to another, a somewhat different approach is followed. Because it is not mathematically possible to create codes that are orthogonal for different starting times, a unique *pseudorandom* sequence, popularly also called a *pseudonoise* (PN) sequence, is used. The PN sequences are not correlated statistically, so the resulting interference can then be modeled as Gaussian noise that allows

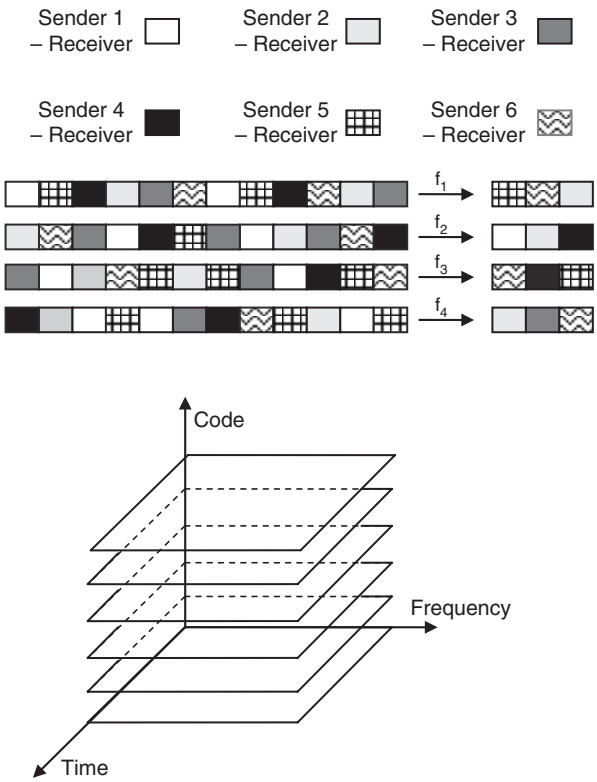


Figure 12-12 CDMA illustration. Senders send data at all frequencies and at all times by spread-spectrum techniques, where each sender uses a specific code or identifier. All receivers can hear all the senders. The code must be known to the receiver to decode the signal. The bottom figure shows how each communication is achieved by giving each sender a different code slot.

extraction of the signal using the PN codes. Each sender-receiver pair thus communicates using a unique PN code. Figure 12-12 shows how CDMA using unique codes compares with the FDMA and TDMA illustrations on the previous pages. Bandwidth utilization is maximized in this case compared with the other two cases.

3.3.4 Space Division Multiple Access (SDMA)

This method has recently become popular to make power-efficient distribution of wireless communications. In wireless networks, if TDMA, FDMA, or CDMA is used by itself, the base station has no information about the location of the receiver mobile client. Hence, the base station needs to radiate the signal in all directions to ensure communication, resulting in wasted power and transmission in directions where no mobile client can access the signal. The omnidirectional radiation also interferes with signals in adjacent cells transmitting on similar frequencies and the signal reception at

the mobile client contains more interference and noise. Space Division Multiple Access methods aim to reduce this problem by exploiting the spatial information of the location of the mobile client with respect to the base station. Assuming that the mobile client location is not going to suddenly change, the base station can efficiently send a signal in the mobile unit's direction, making the ultimate signal reception clear of noise and interference, as well as conserve power or energy by not blindly radiating the same signal in all directions. Smarter antenna technologies are employed at the base station to achieve SDMA, where a system of antenna arrays are used to decipher the direction of arrival (DOA) of the signal from the mobile client and use this to track, locate, and choose which antenna should broadcast the signal. The actual broadcast in a specific direction can be done via TDMA, CDMA, or FDMA. Figure 12-13 shows an example of this.

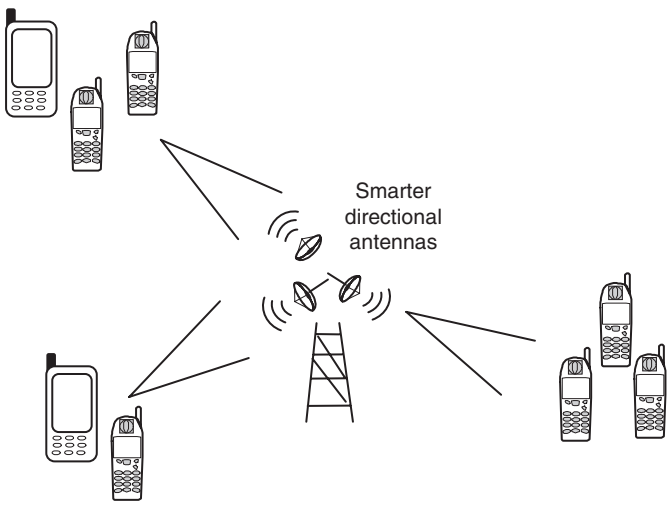


Figure 12-13 SDMA illustration. Smarter antennas, such as spot beam antennas, are used to send signals in a directional manner. The antennas dynamically adapt to the number of users.

4 WIRELESS GENERATIONS AND STANDARDS

The recent evolution of wireless communications has been categorized into different groups. There are those that relate to voice and wireless cellular networks and those that relate to wireless LAN internetworking (the IEEE 802.11 family) computers.

4.1 Cellular Network Standards

A cellular network is a radio-based communication network made up of a number of transmitters called radio cells or base stations. The base stations are geographically distributed to provide a coverage area. The development of cellular networks began in the 1980s and has given birth to three generations.

4.1.1 First Generation (1G)

The 1G wireless networking was used almost exclusively for voice communications such as telephone conversations. The communications transmitted data using analog technology such as FDMA or Frequency Division Multiple Access. In FDMA, each user is assigned a distinct and separate frequency channel during the length of the communication. The table in Figure 12-14 describes the essential features of the first-generation systems. The Advanced Mobile Phone Service (AMPS) in North America is an example of a 1G standard that made use of FDMA. FDMA has also been used in the Total Access Communication System (TACS) and Nordic Mobile Telephony (NMT) in Europe. AMPS operates in the 800–900 MHz range. The downlink from a base station to a mobile station operates in the band of 869–894 MHz, while the uplink from mobile station to base station operates at 824–849 MHz. Each 25 MHz frequency band, either uplink or downlink, is then further divided using FDMA so multiple users can access it. TACS operates similarly in the 900 MHz range. Other 1G wireless standards based on FDMA are HICAP developed by NTT (Nippon Telegraph and Telephone) in Japan and DataTac developed by Motorola and deployed as an ARDIS network originally used for pagers. The C-450 was installed in South Africa during the 1980s and operates in an uplink range of 915–925 MHz and a downlink range of 960 MHz. It is now also known as Motorphone System 512 and run by Vodacom South Africa. RC-200 or Radiocom 2000 is a French system that was launched in November 1985.

	AMPS	TACS	NMT	NTT	C-450	RC2000
Medium access	FDMA	FDMA	FDMA	FDMA	FDMA	FDMA
Uplink (MHz)	824–849	890–915	453–458 / 890–915	925–940	450–455	414–418
Downlink (MHz)	869–894	935–960	463–468 / 935–960	870–885	460–465	424–428
Modulation	FM	FM	FM	FM	FM	FM
Number of channels	832	1000	180/1999	600	573	256
Channel spacing (KHz)	30	30 25	25/12.5	25	10	12.5

Figure 12-14 1G analog cell phone standards

One important aspect of 1G FDMA technology is that because the frequency space is shared among users, frequency bands can be reused. Figure 12-15 shows an illustration where a geographic area is divided into cells. Each cell uses a certain unique frequency channel for communication that does not correspond to the immediate neighboring cells. However, other cells can reuse the same frequency channels for communication. This cell division gives rise to the *cell* in cell phones and cellular technology.

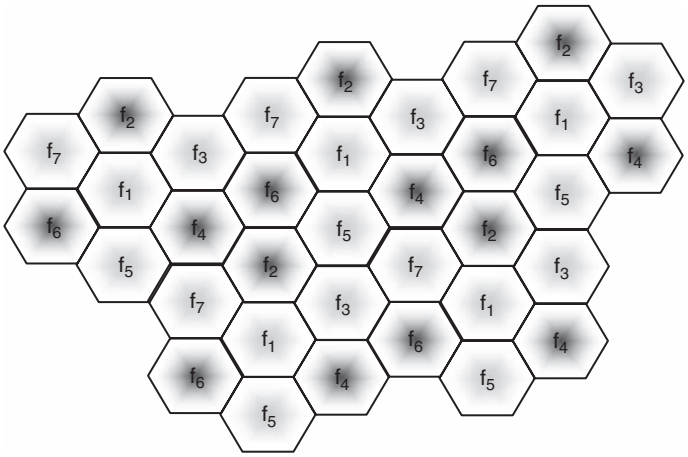


Figure 12-15 FDMA-based cellular breakdown. Frequency bands are reused in different areas as long as no two neighboring areas have the same frequency. A client will switch channels as it moves from one region to another.

4.1.2 Second Generation (2G)

The 2G wireless communication standards predominantly use digital technology and provide other services, such as text messaging, streaming audio, and so on, in addition to telephone voice communications. Communication in the 2G is accomplished by Time Division Multiple Access (TDMA) or Code Division Multiple Access (CDMA). As the name suggests, TDMA creates multiple channels by multiplexing time slots on the same carrier frequency so that multiple users can communicate on the same frequency. TDMA is explained in Section 3.3.2. An example of a 2G standard is the Global System for Mobile Communications (GSM), which was established in 1982 by the European Conference of Postal and Telecommunications. GSM is now one of the most popular wireless communication standards used worldwide. In Europe, GSM originally operated in the 900 MHz range (GSM 900) but now also supports frequencies around 1.8 GHz (GSM 1800). In North America, GSM uses the 1.9 GHz frequency range.

The technical details of the GSM standard were finalized around 1990 with the first commercial applications as early as 1993. As with 1G technology, GSM uses different bands for uplink and downlink. GSM 900 uses 935–960 MHz for uplink and 890–915 MHz for downlink. Each uplink (or downlink) band uses FDMA to have 124 carrier frequencies. Each carrier frequency is then further divided into time-based frames using TDMA to support an additional 26 channels per carrier frequency.

One key feature of GSM is the Subscriber Identity Module (SIM). This is commonly implemented as a SIM card containing a unique personalized identity that can be inserted into any mobile client, allowing each user to retain identity information across different handheld devices and also across different operators on a GSM network. A GSM network that provides all the communication services can be divided into different modules, as shown in Figure 12-16. It consists of mobile clients that communicate to base stations forming the wireless network. The base stations in turn

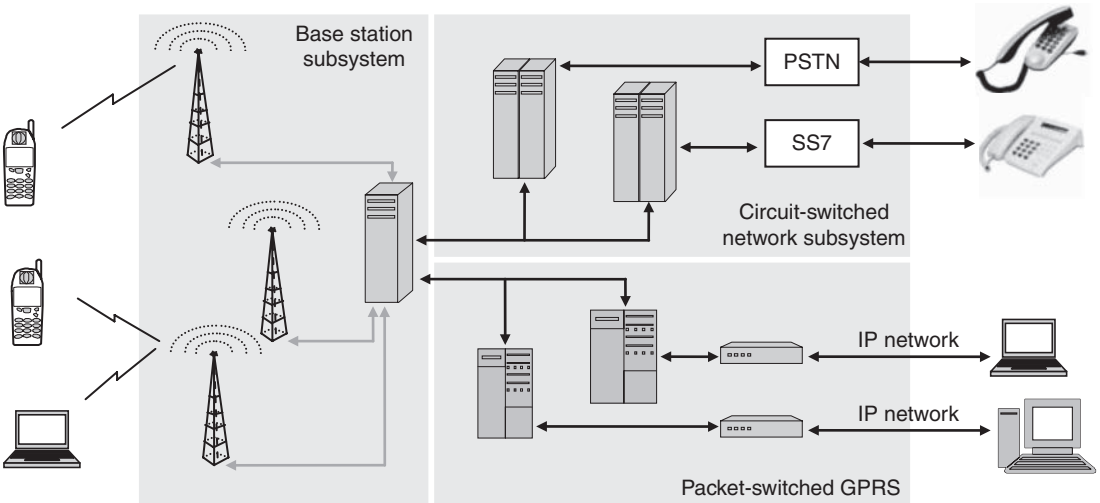


Figure 12-16 Structure of a GSM network

connect to both circuit-switched networks such as the Public Switched Telephone Network (PSTN), signaling system 7 (SS7) telephonic networks, and packet-switched IP networks such as the Internet using the General Packet Radio Service (GRPS). The GSM network started as a circuit-switched network limiting data rates to 9.6 Kbps. The General Packet Radio Service (GPRS) developed in the late 1990s supports packet-switched networks, thus allowing users to be always connected, with a 56 Kbps bandwidth support. The GRPS is also referred to as 2.5G (before 3G).

Other 2G wireless standards are the IS-54 and IS-136 mobile phone systems, also known as Digital AMPS (D-AMPS) used throughout North America. D-AMPS is now being replaced by GSM/GRPS and more advanced 3G technologies. The IS-95 is another 2G mobile standard based on CDMA and developed by Qualcomm. The Personal Digital Cellular (PDC) network phone standard was developed and exploited mainly in Japan by NTT DoCoMo. It provides services such as voice, conference calling, and other data services up to 9.6 Kbps and also packet-switched wireless data (PDC-P) up to 28.8 Kbps. iDEN, developed by Motorola, is another 2G wireless standard supported using TDMA. iDEN was later upgraded to support higher 100 Kbps bandwidths in the form of WiDEN. Figure 12-17 relates a few qualitative features of these second-generation standards.

4.1.3 Second and Half Generation (2.5G)

As the numbering suggests, the 2.5G pertains to technology between 2G and 3G cellular wireless generations. Once digital cellular voice became available with 2G systems, there was a need to incorporate additional data services, such as e-mail, text messaging, multimedia messaging, and so on. The 2G systems with such enhancements are unofficially termed as 2.5G systems, mostly for marketing reasons. GSM systems followed different upgrade paths to provide data services, the simplest being the High Speed Circuit Switched Data (HSCSD), which allows up to four consecutive

	GSM	IS-136	IS-95	PDC
Medium access	TDMA with FH	TDMA	CDMA	TDMA
Uplink (MHz)	890–915	824–849	824–849	810–830, 1429–1453
Downlink (MHz)	935–960	869–894	869–894	940–960, 1477–1501
Modulation	Gaussian Minimum Shift Keying (GMSK)	Phase Shift Keying PSK	Phase Shift Keying PSK	Phase Shift Keying PSK
Number of channels	1000	2500	2500	3000
Channel spacing (KHz)	200	30	1250	25
Compressed speech rate (Kbps)	13	8	1.2–9.6	6–7

Figure 12-17 Second-generation digital cellular phone standards

time slots to be assigned to a single user providing up to 55 Kbps. The circuit-switched HSCSD systems were later enhanced to the packet-switched General Packet Radio Service (GPRS) providing data rates up to 171.2 Kbps. These data rates of GSM were further improved through variable-rate modulation and coding, referred to as Enhanced Data Rates for GSM Evolution (EDGE), which provided data rates up to 384 Kbps. GPRS and EDGE are compatible not only with GSM but also with IS-136.

4.1.4 Third Generation (3G)

The third-generation wireless standardization process started in 1998 with proposals for International Mobile Telecommunications (IMT-2000). The proposal became necessary to address issues with the many frequency bands associated with the 2G standards and also for the need to provided additional standardized data services. The 3G services provide voice communications (like 1G), narrowband data communication services (like text messaging in 2G), and also broadband data communications, such as video streaming, Web surfing, continuous media on demand, mobile multimedia, interactive gaming, and so on. In other words, the 3G family of standards provides the same bandwidth capabilities of WLAN but at a more global and public level. With the data rates that 3G supports, it will enable networked services and wireless services to converge better.

Although designed to be a unified standard, a common agreement was not reached among members, resulting in two substandards within the 3G family. Most countries supported the CDMA2000 standard, which was backward compatible with the 2G cdmaOne. The other part is the wideband CDMA (W-CDMA), which is backward compatible with GSM and IS-136 and also supported by the Third Generation Partnership Project (3GPP). Figure 12-18 summarizes the main characteristics of these two standards. Both use CDMA as a multiplexing mechanism, but have different chip rates.

Characteristics	CDMA-2000	W-CDMA
Modulation	PSK	PSK
Peak data rates (Mbps)	2.4	2.4 (8–10 with HSDPA)
Channel bandwidth (MHz)	1.25	5
Chip rate (Mbps)	1.25	3.84

Figure 12-18 Third-generation digital cellular phone standards

The CDMA2000 standard builds on cdmaOne. Central to the CDMA2000 standard is its 1xRTT indicating that the radio transmission technology (RTT) operates in one pair of 1.25 MHz radio channels, and is, thus, backward compatible with dmaOne systems. The CDMA2000 1X system doubles the voice capacity of cdmaOne systems, providing high-speed data services peaking at 300 Kbps, and averaging 144 Kbps. W-CDMA is the other competing 3G standard to CDMA2000 and is an evolutionary successor to GSM. The W-CDMA development in the 3G group is addressed by the Universal Mobile Telecommunications System (UMTS). There are three main channel capabilities proposed for UMTS—a mobile rate of 144 Kbps, a portable rate of 384 Kbps, and an in-building rate of 2 Mbps. UMTS will have the capacity to provide wireless static and on-demand services requiring less than 2 Mbps. It is slated to combine a range of applications that include cellular phones, cordless phones, and mobile data networking for personal, business, and residential use. W-CDMA supports peak rates of up to 2.4 Mbps, with typical average rates at 384 Kbps. W-CDMA uses 5 MHz channels, in contrast to the 1.25 MHz channels of CDMA2000. An enhancement to W-CDMA called High Speed Data Packet Access (HSDPA) provides data rates of around 8–10 Mbps. With the 3G standard already well established, the HSDPA is also thought of as the precursor to fourth-generation systems.

4.2 Wireless LAN Standards

The need for wireless communication among computers was driven mostly by the increasing availability of laptop computers and the emergence of pervasive or ubiquitous computing. The IEEE 802.11 is a family of protocols defined for wireless local area networks. It currently consists of wireless modulation protocols illustrated in the table shown in Figure 12-19.

4.2.1 802.11 Original

The original version of the 802.11 standard was released in 1997. It was designed to support a maximum of 2 megabits per second to be transmitted via IR signals in the ISM frequency band (Industrial Scientific Medical) at 2.4 GHz. For medium access, it used Carrier Sense Multiple Access with Collision Detection (CSMA/CD), which traded bandwidth for reliability of data transmissions. Consequently, during actual transmission on a wireless network, the average throughput yield was 1 Mbps.

Type of 802.11 protocol	Radio communication frequency	Maximum data rate designed	Average data rate	Signal range (indoor)	Signal range (outdoor)
802.11 legacy or 802.11 original	2.4 GHz	2 Mbps	1 Mbps	up to 20 meters	100 meters
802.11 b	2.4 GHz	11 Mbps	6 Mbps	up to 30 meters	140 meters
802.11 a	5 GHz	54 Mbps	25 Mbps	up to 30 meters	120 meters
802.11 g	2.4 GHz	54 Mbps	25 Mbps	up to 40 meters	140 meters
802.11 n	2.4 GHz	100–200 Mbps	54 Mbps	up to 70 meters	250 meters

Figure 12-19 IEEE 802.11 family

Although the original specification was for wireless, it offered many choices and services that made interoperability testing from different manufacturers a challenge. Ultimately, it never materialized into consumer-level products, such as wireless routers and receivers, and remained in a developmental and testing status until the 802.11b specification was ratified.

4.2.2 802.11b

The original 802.11 specification was amended in 1999 to form the 802.11b specification supporting a maximum bit rate of 11 Mbps. It is sometimes also referred to as 802.11 High Rate. The 802.11b uses direct-sequence spread spectrum (DSSS) discussed in Section 3.3.3.2 to modulate information at 2.4 GHz. Specifically, it uses Complementary Code Keying (CCK) where a complementary code containing a pair of finite bit sequences of equal length is used for the spectral spread. 802.11b allows wireless transmission capabilities to be comparable to the popularly wired Ethernet. The 802.11b is normally used in a point-to-multipoint manner, where a central station communicates with mobile clients using an omnidirectional antenna. The mobile client range is around 30 m from the central station. The 802.11b has also been proprietarily extended to 802.11b+ to include channel bonding, better handling of bursty transmission, and so on, yielding an increased speed up to 40 Mbps.

4.2.3 802.11a

The 802.11a was also ratified in 1999 and uses the same core architecture of the original 802.11, but operates in the 5 GHz bandwidth. The 2.4 GHz band is heavily used by other instruments, such as cordless phones, Bluetooth-enabled devices, microwaves, and so on, which is not the case for the 5 GHz band. Communicating at 5 GHz has the added advantage of less interference and, hence, is capable of delivering better throughputs. The 802.11a uses orthogonal frequency division multiplexing (OFDM; see Section 3.3.1), which allows a maximum data rate of 54 Mbps. However, the 5 GHz higher band has other problems, such as it is more readily absorbed (and not easily reflected) than the 2.4 GHz band, yielding accessibility problems. Line of sight is the best way to get good signal throughput and the access distance is in the

region of 10 m, depending on the environment, less than that of 802.11b. Overall, 802.11a did not become widely popular mainly because 802.11b was already adopted and the distance restrictions imposed compared with 802.11b. However, 802.11a is used in wireless ATM systems where it is used in access hubs.

4.2.4 802.11g

The 802.11g standard was established in 2003 and operates at a maximum of 54 Mbps like 802.11a. It uses OFDM as its modulation scheme for higher data rates but at lower rates can also switch to CCK (5–10 Mbps) and DSSS (1–2 Mbps), which is like its predecessor formats. The 802.11g operates at 2.4 GHz, similar to 802.11b, but it can achieve higher data rates similar to 802.11a. However, the range in which wireless clients can achieve the maximum 54 Mbps is less than that of 802.11a. The 802.11g has by far become the most popular wireless standard in WLANs, not only because of its efficiency, but also because manufacturers of 802.11g components also support the 802.11 a/b standards. However, it does suffer from the same interference problems as 802.11b because of the many devices using the 2.4 GHz band.

4.2.5 802.11n

The 802.11n standard is currently in the making and aims to deliver a maximal through put of 540 Mbps, with an average of 200 Mbps. It will operate at 2.4 GHz and will build on its predecessor by using multiple-input multiple-output (MIMO) technology. MIMO uses multiple antennas at the sender and receiver yielding significant increases in throughput through spatial multiplexing that creates increased access range because of multiple directions to send/receive.

4.3 Bluetooth (IEEE 802.15)

Bluetooth is a new standard developed for very short-range wireless communications in wireless personal area networks. In particular, it is intended to use in a one- to two-meter range, such as to replace cables connecting mobile cell phone pieces, or cables connecting various components of a desktop (keyboard, mouse, printers), or connecting devices in a car. It uses the 2.402–2.48 GHz band to communicate and was originally developed by Ericsson and later formalized by the Bluetooth Special Interest Group (SIG) established in 1998 by SONY, Ericsson, IBM, Intel, Toshiba, Nokia, and others. There are many flavors of its implementation. Consumers got drawn to it because of its wireless, inexpensive, and automatic working qualities. Bluetooth devices avoid interfering with other systems at the same 2.4 GHz range by sending out very weak signals (at 1 milliwatt power) that fade off in 1–10 meter ranges. Compare this with cell phones that send 3 watt signals. Bluetooth is designed to connect to multiple devices simultaneously. Multiple devices do not interfere because of the frequency hopping spread spectrum technology to control access. Bluetooth uses 79 randomly chosen frequencies in its operating range to spread the information signal.

The original versions 1.0 and 1.0B had numerous problems, resulting in interoperability checking, and were subsequently replaced by the more popular standards.

- *Bluetooth* 1.1—This was the first commercialized working version and added support for nonencrypted channels.
- *Bluetooth* 1.2—This uses adaptive frequency hopping spread spectrum (AFH), which improves signal quality whenever interference is present. It has higher transmission speeds of up to 700 Kbps.
- *Bluetooth* 1.3—This version has faster transmission speeds of up to 2.0 Mbps and higher, lower power consumption, and lower bit error rate.

5 WIRELESS APPLICATION PROTOCOL (WAP)

Mobile devices such as PDAs and cell phones have become carry-on devices and are used primarily for voice communications. Along with the mobile voice services, there is also a need to access information services, for instance, browsing the Web on your PDA or using your cell phone to complete a transaction. Such applications require a standardized way to communicate information on mobile devices and support information applications. The HTML language and the HTTP protocol suite were designed to accomplish this need for the Internet on larger computers. The Wireless Access Protocol (WAP) does the same for mobile devices by providing a standard to communicate, process, and distribute information. It is, thus, very similar to the combination of HTML and HTTP, except that it has been optimized for low-bandwidth, low-memory devices that have smaller display capabilities—such as PDAs, wireless phones, pagers, and other mobile devices. It was originally developed by the WAP forum (now a part of the Open Mobile Alliance [OMA]), a group founded by Ericsson, Nokia, Motorola, Microsoft, IBM, and others. WAP can be thought of as a standardized communication protocol as well as an application environment standard that is independent of specific cell phone device hardware or the supporting communications network. WAP was developed to be accessible to the GSM family (GSM-9000, GSM-1800, GSM-1900), CDMA systems (IS-95), TDMA (IS-136) systems, and 3G systems (IMT-2000, UMTS, W-CDMA), and extensible to future cell networks.

WAP protocols work similarly to HTTP requests. A WAP request is routed through a WAP gateway, which acts as an intermediary processing unit between the client cell phone, which might be on a GSM or IS-95 network, and the computing network mostly supported by TCP/IP. The WAP gateway, thus, resides on the computing network, processes requests from a mobile client, gets information, retrieves content using TCP/IP and other dynamic means, and, ultimately, formats the data for return to the client. The data is formatted using the Wireless Markup Language (WML), which is similar to HTML and based on XML. Once the WML that contains the content is created, the gateway sends the request back to the client. The client is equipped with a WML parser that describes how to display the data on the display terminal.

The WAP protocol suite that is made to be interoperable on different devices follows a layered approach, similar to the OSI for networks. These layers consist of the following:

- *Wireless Application Environment (WAE)*—This is the environment that contains high-level API descriptions used by applications to initiate requests or send information.

- *Wireless Session Protocol (WSP)*—The WSP is best thought of as a modest implementation of HTTP. It provides HTTP functionality to exchange data between a WAP gateway server and a WAP mobile client. It can interrupt information transactions and manage content delivery from server to client in a synchronized and asynchronized manner.
- *Wireless Transaction Protocol (WTP)*—WTP provides the transaction support for a transaction initiated by WSP. This is similar to the TCP or UDP delivery across standard networks, but is designed more for packet-loss problems common in 2G and 3G wireless networks. WTP manages the overall transaction between the WAP server and WAP client.
- *Wireless Transport Layer Security (WTLS)*—This is an optional layer, which if invoked makes use of public-key cryptography to deliver data in a secure manner.
- *Wireless Datagram Protocol (WDP)*—This is the final layer that operates above the device and routes data along the wireless network, similar to the Network-Transport layer combination in OSI. Although it uses IP as its routing protocol, it does not use TCP but UDP due to the unreliable nature of wireless communications.

Initial WAP deployments were harder because of business models for charging air-time, lack of good authoring tools, and lack of good application creators and content providers. Nevertheless, it is now successfully used in all markets with the introduction of wireless services from providers, for example, T-Mobile T-Zones, Vodafone Live, and so on. WAP has been successfully used to create many applications, such as SMS Text, Web browsing and e-mail on your mobile device, messaging, and MMS (Multimedia Messaging Service) on different devices that communicate across different networks.

6 PROBLEMS WITH WIRELESS COMMUNICATION

Compared with wired communication, wireless communication presents a variety of problems caused by the absence of a hard communication medium. In this section, we present some of the most common problems that cause radio signals to degrade over distance.

Most basic radio communication models assume that radio waves emanate from a sender that functions as a point source and travel in any direction in an uninterrupted straight line, and, thus, fill a spherical volume centered at the source. A receiver anywhere in this volume can receive the signal. However, this simplistic model does not accurately reflect the actual way senders and receivers communicate indoors or outdoors. Although the effects are more drastic indoors than outdoors, the basic mechanism of the radio wave propagation model is affected by the following:

- *Reflection*—This occurs when a traveling radio wave bounces off another object that has larger dimensions than its wavelength. Reflections are normally caused

by large solid objects, such as walls, buildings, and even the surface of the Earth or ionosphere.

- *Diffraction*—This occurs when the pathway between a sender and receiver is obstructed by a surface with sharp edges and other irregularities. The sharp edges cause the wave to diffract, resulting in many waves and directional changes.
- *Scattering*—This occurs when there are objects in a radio wave's pathway that are smaller in size compared with the wavelength of the signal. In case of radio waves, common indoor/outdoor objects that cause scattering are leaves, street signs, lamps, pipes, and so on.

All these phenomena add noise to the signal, but they also cause the well-known radio wave communication problems described in the following sections.

6.1 Multipath Effects

Multipath is a term used to describe the multiple pathways that a radio signal might follow instead of, or in addition to, a straight line between source and destination. The signals arriving on different paths have their own attenuations, time, and/or phase delays resulting in constructive or destructive interference and overall signal degradation. Figure 12-20 illustrates the effect of multipath interference on a signal at the receiving end.

In case of outdoor signals, such as television transmission, the multipath effect causes jitter and ghosting, manifesting as a faded duplicate image right on top of the main image. In radar communications, multipath causes ghosting where ghost targets are detected. In cellular phone communications, multipath causes errors and affects the ultimate signal quality. Multipath signals are even more common in indoor

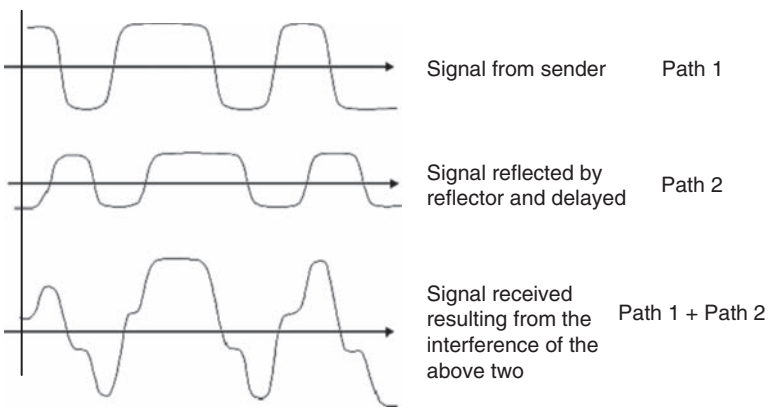


Figure 12-20 Multipath effects on signal. The top signal is sent by the sender, the middle signal arrives at the sender because of reflections, and the bottom signal is what the receiver potentially receives because of interference caused by two same, though out-of-phase signals.

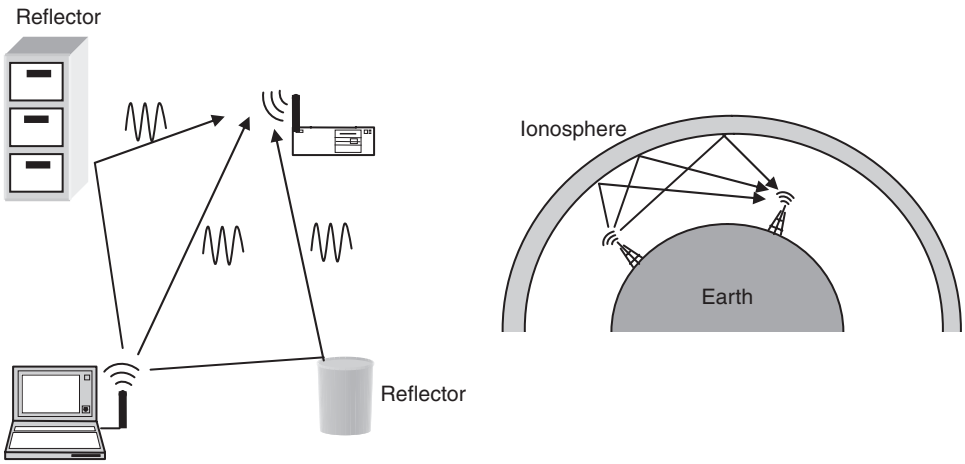


Figure 12-21 Multipath reflections. The left figure shows how a signal from a sender arrives at a receiver via multiple paths in an indoor environment. The right figure shows the same for outdoor environments. The radio signal comes to the receiver's antenna by bouncing off the Earth's ionosphere in multiple ways.

communications, such as in wireless LANs and cordless phones. Here, there is a greater chance of a signal bouncing about stationary and moving objects before it reaches its final destination. Figure 12-21 illustrates simple causes of multipath effects in indoor and outdoor settings. There are mathematical models used to approximate multipath effects, such as the Rayleigh fading model, which best fits reflections from hills, buildings, vehicles, and so on.

Effective techniques used to alleviate problems caused by multipath effects are to design receivers appropriately or design waveforms in such a manner as to cause less multipath effects. An example of the former is using *rake receivers*, whereas orthogonal frequency division multiplexing (OFDM) is an example of the latter. Rake receivers incorporate antennas designed to get over the multipath fading problems. It does this by using more than one, typically several, antennas. Each antenna is tuned to receive a signal delayed slightly to simulate receiving multiple receptions. The signal at each is decoded independently, but integrated appropriately at a later stage to make use of different transmission characteristics of each transmission path. In OFDM, the signal is multiplexed among different frequencies in a manner such that their relative phase is independent of each other, which eliminates interference.

6.2 Attenuation

Attenuation is the drop in the signal power when transmitting from one point to another. This phenomenon is also termed as path loss and is predominantly caused by the transmission path length as the signal weakens due to effects of refraction, diffraction, and free space loss with distance traveled. Additionally, obstructions in the

signal path can also cause attenuation and signal energy loss, for instance when a signal bounces off of multiple objects and is reflected, energy is lost due to absorption and dissipation. Multipath effects can also cause signals to attenuate when signals from two paths destructively interfere. The Friis radiation equation is one way to model attenuation, where the signal power decreases inversely to the distance traveled.

$$P_r = P_s \frac{G_s G_r \lambda^2}{(4\pi^2) d^n L}$$

P_s and P_r are the signal power at the sender and receiver.
G_s and G_r are the antenna gain at the sender and receiver.
λ is the communication wavelength, L is the loss at the receiver, and n is the path loss factor.

The path loss factor n is normally in the range of 2 to 4. $n = 2$ for free space propagation where the sender and the receiver are in direct line of sight. $n = 4$ for lossy environments where the signal bounces off one or more reflectors before arriving at the receiver. Predicting the exact path loss values is only accurate with line-of-sight communication, for other interreflecting scenarios, only an approximation can be obtained.

6.3 Doppler Shift

Doppler shifts are frequency variations caused by the relative motion between sender and receiver. When a sender and receiver are moving relative to one another, the frequency of the received signal is not the same as that sent by the source. When moving toward each other, the received signal's frequency is higher than the sent frequency and vice versa. This phenomenon, called the Doppler effect, is commonly observed for all frequencies. For instance, the pitch of a fire engine's siren appears to change as the fire engine approaches the observer and drives past. The Doppler shift in frequency is computed as follows:

$$\Delta f = f \frac{v}{c}$$

f is the frequency of the sender.
v is the relative speed difference (can be + ve or − ve)
c is the speed of light.

To understand the effect the Doppler shift can have on wireless communication when the relative motion is high, let us take the example of a car traveling in a car at 60 km/hr. If we let $f = 1$ GHz, which is roughly the band of wireless cellular communications and $v = 60$ km/hr and $c = 3 \times 10^8$ m/s, then

$$\Delta f = 10^9 \times \frac{16.67}{3 \times 10^8} \qquad 60\ km/hr = 16.66\ m/s$$
$$= 55.5\ Hz$$

The shift of 55 Hz is not high, but can cause problems depending on the transmission modes used. For example, with FDMA, it can cause interfrequency interference. If v is high, for instance in the case of low earth orbiting communication satellites, the shift can cause an impact on the quality.

6.4 Handovers

Current geographically distributed wireless deployments, whether over a small area or large area, make use of multiple receivers. This is because of a variety of problems inherent to radio communications, such as signal attenuation over distance, larger coverage area needed on a single network, communication across different networks, and so on. In such instances, a mobile client invariably needs to communicate with multiple base stations as the mobile client changes positions. For example, when traveling while talking on a cell phone, the cell phone might need to change the receiving base stations (cells) it is communicating with. If a laptop is traveling in a large building with wireless LAN support, or Wi-Fi support, it will need to switch between wireless routers. This switching from one base station to another is technically termed as a *handover* or *handoff*. Depending on the mode of communication and the circumstances when the handover is happening, there is a likelihood that during the transition phase, data packets are garbled, lost, dropped, or the connection is broken and needs to be reestablished. Handovers are classified as being *hard* handovers or *soft* handovers.

In FDMA systems, each base station (or cell) communicates at a different frequency with a mobile unit. Each mobile unit is connected to only one base station at a given time. Hence, when a mobile unit switches from one base station to another, the first connection has to be broken for a short duration and then reestablished with a different base station. This is termed as a *hard* handover. The same is true for TDMA systems, for instance the GSM and GPRS network systems support hard handovers. A soft handover, on the other hand, allows a mobile unit to be simultaneously connected to two or more base stations. In coverage areas that overlap, the mobile unit can listen to signals from more than one base station. The decoded signals from all these stations are combined to create the overall received signal. If the received signal power of one base station begins to fade, the other neighboring base stations signals are still received at the same frequency. Hence, during a transition, there is a higher likelihood of a mobile unit's call connection and quality being maintained. This is termed as a *soft* handover.

Additionally, a handover can also be characterized based on the number of network interfaces involved in the handover. A vertical handover is a handover between two network access points, which are usually in different network domains or technologies. For example, when a mobile device moves out of a 802.11b covered network area into a GPRS network, the handover would be considered a vertical handover. A horizontal handover, on the other hand, is a handover between two network access points that use the same network technology and interface. For example, when a mobile device moves in and out of various 802.11b network domains, the handover activities would be considered as a horizontal handover because the connection is disrupted solely by device mobility. Vertical and horizontal handoffs are becoming increasingly important with the different urban networks being deployed today to ensure seamless and continuous connectivity for mobile users. For example, a mobile device might be on a CDMA-based cell network but need to ultimately go via a Wi-Fi network to the Internet to access information or another mobile unit on a different network.

7 QUALITY OF SERVICE (QoS) OVER WIRELESS NETWORKS

It is expected that multimedia services offered on wireless devices will continue to grow in content complexity, interactivity, and real-time delivery. One obvious conclusion that we can draw is that higher bandwidths are required to support multimedia content delivery over wireless networks. The bandwidths supported by wireless LANs do make this possible; however, cell phone network bandwidth still needs to rise. For example, Web browsing and streaming audio might be a consumer reality using 2G networks, but videoconferencing, video streaming, and collaborative work are not entirely supported in 2G networks but are currently well supported by the high-bandwidth 3G networks. In such cases, real-time delivery of content can pose interesting challenges when compared with wired networks because wireless data transmission occurs over narrower bandwidths and incurs a larger comparative data loss and distortions. This introduces more stringent requirements on QoS issues when compared with wired networks. The factors that define QoS—error resiliency, error correction, synchronization, delay, and jitter—were introduced in Chapter 11. Also explained there are classifications of QoS services and packet scheduling at intermediary nodes, which allows realizing the necessary QoS for an application on a wired network. For the most part, all these definitions stay the same for wireless networks. However, protocols to implement QoS on a wireless network and their architectures do differ.

QoS architectures in wireless networks can be considered in two ways. The first case deals with infrastructure networks, where there are two types of stations: end stations (hosts) and a central station (also known as an access point or base station). The central station regulates all the communication in the network—that is, any two host terminals that need to communicate do so via the central station and there is no peer-to-peer communication that occurs directly between the hosts. The traffic from a source host is sent to the central station and then the central station forwards the traffic to the destination host. Traffic handling such as classification, traffic policing, packet scheduling, and channel access as well as resource reservation mechanisms reside in all stations (end hosts and central station). In addition, the central station also includes an admission control mechanism. Figure 12-22 shows the overall architecture in such a network where two hosts are communicating via a base station. Each host and the base station allocate queues depending on the differentiating services requested and packets are scheduled according to priority queuing (*PQ*), custom queuing (*CQ*), and weighted fair queuing (*WFQ*).

In a wired connection, the sender or receiver is always connected to a network. Under normal circumstances, it can be assumed that once a session is set up between them with optional reservations, the communication session will remain in effect until either the sender or receiver terminates it gracefully using any communication protocols used. Therefore, it is easier in wired networks to accommodate both inelastic and elastic traffic. An elastic scenario is achieved by an appropriate reliable protocol such as TCP and inelastic traffic is achieved by guaranteeing availability of network resources using the RSVP, SIP, and other session management protocols. However, these solutions cannot be naively extended to work for a wireless setting because the channel is shared and users have mobility. In a wireless medium, both the user and receiver are freely roaming, which can cause loss of connection between them and the

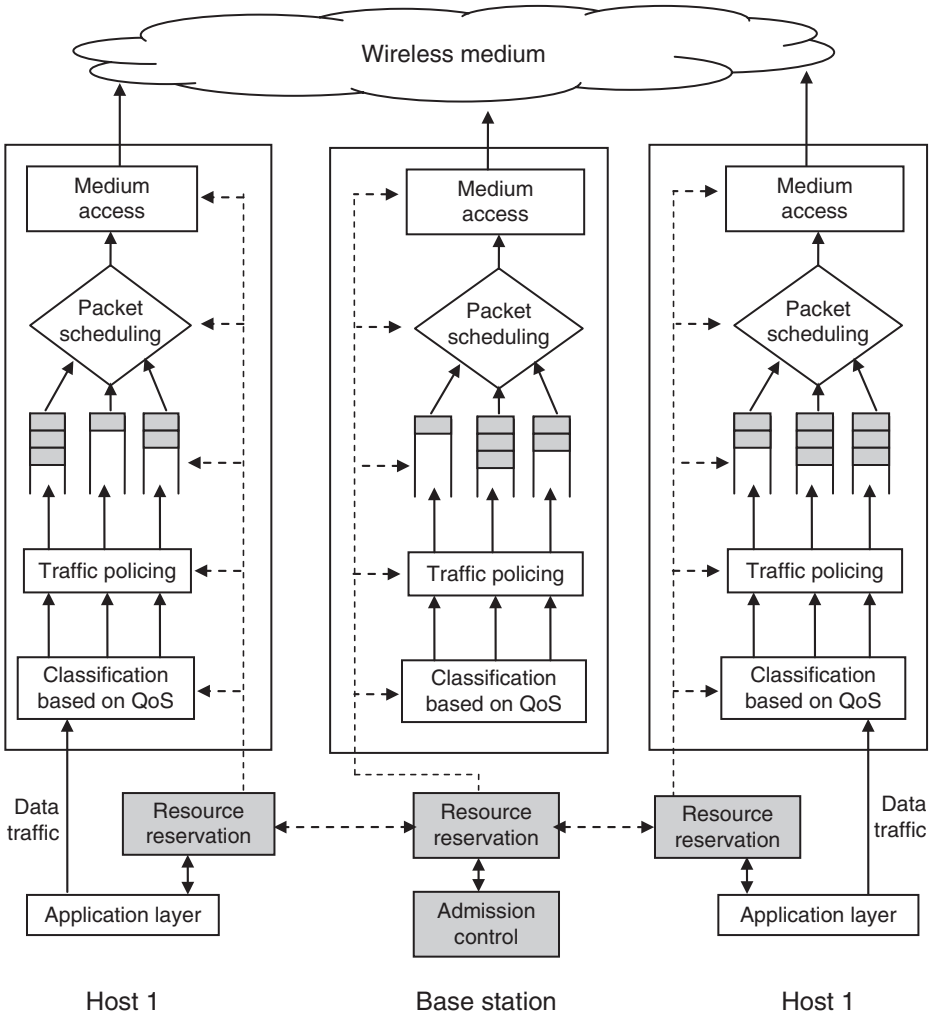


Figure 12-22 Architecture for QoS in a wireless network

base station. This might arise when the sender or receiver has to change base stations as they move from one area to the next. There might also be a loss of connection with the same base station owing to obstruction in line of sight while moving, for example the sender momentarily goes behind a large building and temporarily loses connection. These ill-posed and varying factors make it harder to accommodate inelastic traffic situations.

Advancements to the normal wired protocols are normally made to deal with QoS in wireless domains. In the following sections, we discuss such advancements in the different OSI layers. In addition, improvements can be made by adapting the content streams and managing bit rates more amenably. Some of these solutions for wireless QoS are discussed next.

7.1 Extending Application-Layer Protocols

Among Application-layer protocols, amendments made to SIP provide mobility management for media data over wireless links. SIP is not a QoS management protocol per se, but SIP can be used as a protocol that is responsible for mobility management at the Application layer, with an additional objective to improve QoS for real-time and non-real-time multimedia applications. As discussed in Chapter 11, SIP supports name mapping and redirection, which increases personal mobility so users can maintain an external visible identifier regardless of their network location. To achieve terminal mobility, a SIP mobility implementation could poll the device to find out if a handoff took place. This enables SIP to log a new SIP registration after handoff. SIP-based mobility, thus, offers attractive benefits when used in mobile multimedia applications. However, some inherent problems with this approach make the adoption of this scheme difficult. For example, it cannot handle mid-call subnet changes because it is an Application-layer solution. This is where it requires the support of a lower-level mobility protocol, for example, mobile IP.

7.2 Extending Network-Layer Protocols

The Network layer is the primary OSI layer for enforcing QoS policies in computer networks. This is because the main network components (routers) operate at the network level and play a prominent role in network performance. A number of Network-layer approaches exist that intend to improve mobility management at the IP layer and, thus, the QoS indirectly. Mobile IP is the most well-known mobility management solution. However, handoff delay and overheads of mobile IP's triangular routing, triangular registration, and IP encapsulation are major issues that present a bottleneck for mobile IP to become a widespread acceptable solution for real-time interactive multimedia communications over the wired or wireless IP network. An interesting approach in this area attempts to improve the QoS by providing a more scalable reservation system for wireless applications through localized RSVP messages. In this approach, RSVP is used in such a way that when a mobile node moves to a new point of attachment, new PATH and RESV messages are sent only locally between the mobile node and the base station, creating a hierarchy in the QoS reservation system. This approach also significantly improves the resource reservation time.

7.3 Content Adaptation for Wireless Multimedia Traffic

Mobility issues and noise characteristics of paths between various communications endpoints are expected to cause bandwidth variations over time in wireless networks. In this solution, the content streams are designed such that they can be adapted to the changing bandwidth to significantly improve the quality of service. Chapter 8 mentioned the MPEG-2 stream where the compressed video stream can be divided into a base layer stream and an enhanced layer stream. This paradigm has been further extended to define multiple enhancement media streams in MPEG-4 as we shall see in Chapter 14. Under normal throughput circumstances, both base layer and enhancement layer streams are transmitted. This can be utilized effectively in wireless

networks when the required transmission rate of the data stream exceeds the limit imposed by the effective forwarding rate from the base station to the host. Normally, this would cause a loss of packets and dramatic lowering of the QoS. However, if this packet loss and/or delay time fluctuations can be detected and the loss in effective throughput can be estimated, the session management protocol can switch to communicating only a base layer stream with no enhancements. Figure 12-23 illustrates this bit rate control mechanism.

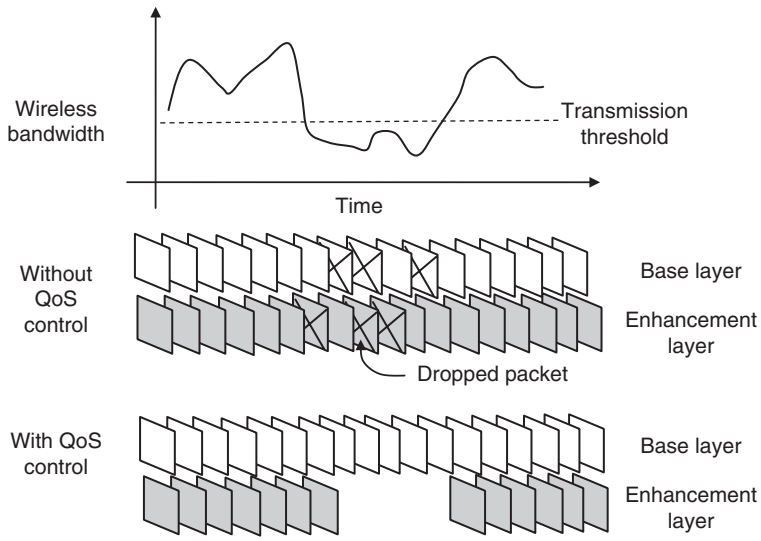


Figure 12-23 QoS obtained by bit rate control in wireless networks. The top figure shows the wireless bandwidth over time in a communication session. The middle figure shows that both base and enhancement layer packets are lost when the bandwidth falls below a threshold if no QoS control is imposed. The bottom figure shows how QoS control adjusts to the bandwidth by allowing only the base layer stream to be transmitted.

8 2G, 3G, AND BEYOND 3G

The 2G standards do deal with images, real-time audio, and Web browsing, but are not capable of dealing with video. The 3G wireless networks support on an average of up to 384 Kbps and have defined video- and audio-compliant standards that will enable videoconferencing, video streaming, and so on. Although the 3G services are still being deployed into the hands of the consumers, researchers and vendors are already expressing a growing interest in Beyond 3G (or 4G) wireless networks. The international wireless standardization bodies are working for commercial deployment of 4G networks before 2015. Like previous generation upgrades, the 4G network is also not meant to be an incremental improvement but a large one that will require network companies to replace their hardware and for consumers to replace their wireless handsets.

The 4G standard is designed to support global roaming across multiple wireless and mobile networks, for instance communication from a handheld in a cellular

network, to a satellite-based network, to a high-bandwidth wireless LAN. This will enable increased coverage, further the convenience of using a single device across more networks, and provide a wider array of multimedia services. 4G is also being poised to deliver seamless, global, high-speed communication at up to 20 Mbps. With this rate, 4G has been destined to achieve wireless broadband. A few salient features of 4G when compared with 3G are shown in the table in Figure 12-24.

Property	3G	4G
Data rate	384 Kbps (up to 5 Mbps when not moving)	Up to 20 Mbps
Communication frequency	1.8–2.5 GHz	2–8 GHz
Medium access	Wideband CDMA	Multicarrier CDMA and OFDMA
Switching	Circuit switched and packet switched	Packet switched

Figure 12-24 Table showing salient comparisons between 3G and 4G wireless standards

Regarding quality of service, 4G systems are expected to provide real-time and Internet-like services, including those by the now-popular wired applications, such as Multimedia Messaging Service (MMS), video chat, HDTV, and digital video broadcasting (DVB). Although guaranteed services are bound to be more effective with the higher bandwidth, it is not clear how an increase in user consumption will be achieved on handheld devices. Apart from guaranteed services, 4G is poised to deliver better-than-best-effort services, which includes the following:

- *Predictive services*—The service needs an upper bound on end-to-end delay.
- *Controlled delay*—The service might allow dynamically variable delay.
- *Controlled load*—The service needs resources.

The 4G network is very ambitious and bound to have a few challenges. Some of these can be enumerated as follows:

- The 4G standard communicates using a higher communication frequency. This might lead to smaller cells that might cause intracell interference or higher noise figures due to reduced power levels.
- Because the 4G standard touts multiple network access, defining interfaces across networks having varying bandwidths and latencies can be problematic.
- Seamless roaming and seamless transfer of services across various networks cannot be supported with the current protocols and new protocols will need to be developed.

- A new IP protocol might be needed because of the variable QoS services and the network should do "better-than-best" effort.
- The digital-to-analog conversions at high data rates, multiuser detection, estimation (at base stations), smart antennas, and complex error control techniques as well dynamic routing will need sophisticated signal processing.

9 EXERCISES

1. [**R02**] Broadcasting is a method to reach all users on a network. In the previous chapter, we talked about different algorithms to implement broadcasting on a connected or wired network. How would you broadcast on a wireless network? Is data broadcasting easier on wired or wireless networks?
2. [**R02**] MAC protocols are used to access the medium.
   - What are the different medium access control protocols for wireless media? Mention the major advantages and disadvantages over each other.
   - Compare and contrast these with some of the MAC protocols for hardwired networks.
   - What are FDD and TDD? What are the advantages and trade-offs with each?
3. [**R03**] TCP is the most widely used protocol over wired networks for providing reliable and in-sequence transport service between hosts over a wired connection. This question relates to the TCP protocol over wireless.
   - Explain why TCP performance worsens in wireless networks.
   - What do you think should be done to improve this performance to make it similar to wired networks?
4. [**R04**] A receiver is 1 m away from the base station and receives a signal whose strength is 100 units. Assuming a reflective environment (not direct line of sight), what is the signal strength if the receiver moves an additional meter away from the base station?
5. [**R04**] Although we have discussed various wireless standards, one related technology, which was not discussed but needs to be mentioned, is paging systems. Paging systems are considered a precursor to modern cellular technology. In this questions you are asked to read up on paging systems and compare them with cell phone systems
   - How do paging systems work and what networks do they make use of?
   - How is the signal different from what is used with current cell phone communication?
6. [**R04**] In CDMA-based communication systems, each receiver is given a unique code.
   - Suppose each code is n bits long. How many possible codes can there be?
   - What condition should any two codes satisfy apart from being unique? Why?

- Write down all possible codes are $n = 4$ bits long. The example in Figure 12-11 shows a signal worked out with two of these codes. Go through the exercise of taking two different codes with the same signal and work out the transmitted signal, the interfered signal, and the received reconstruction.

7. [**R04**] When a mobile unit moves from area to area, it might need to transition between mobile stations. During such transitions, or handovers, it is necessary that the continuity and quality of the ongoing communication transaction be maintained.
   - How do GSM systems deal with handovers?
   - What are the desirable qualities?
   - Which MAC protocol among FDMA, TDMA, and CDMA is more amenable to handovers?
   - Why is the handover using CDMA termed as a *soft* handover? Describe how it works.
8. [**R04**] A plane is traveling at 500 km/hr and is receiving a signal from a control tower that is sent on a 400 MHz band.
   - What frequency should the plane's communication system be tuned to if it has to receive the best possible signal?
   - Does it matter if the plane is approaching the airport at that speed or is flying away at that speed after taking off?
9. [**R04**] Rather than a plane, consider a satellite orbiting at a distance of 40,000 miles above the surface of the Earth. A ground station attempts to communicate with the satellite at 30 GHz. Assume the Earth's radius is 4000 miles.
   - If the satellite needs to tune 500 Hz below the transmitting frequency to receive the signal, what is the relative velocity of the satellite with respect to the Earth's surface?
   - What speed should the satellite be traveling at so as to communicate exactly at 30 GHz?
10. [**R07**] The geometric layout for cellular communication is shown to be hexagonal in Figure 12-15. This question is meant to increase your understanding about such regular geometric layouts and find the most efficient one for communications. For the purpose of this question, assume that the coverage area is flat and supports an ideal line-of-sight path loss.
    - The main idea behind the geometric layout is the regularity where each base station is placed so as to be equidistant from all its neighbors. Is the hexagonal layout the only possibility? What other geometric layouts can there be?
    - Can you analytically prove that there are only three possible layouts that satisfy the condition that each base station is equidistant from all its neighbors?
    - Reason out that the hexagonal one is the most efficient.

11. [**R06**] This question describes some issues regarding the number of reusable frequencies in a geometric cell layout, such as in AMPS or GSM.
    - In these protocols, each base cell communicates with all the mobile clients in a given coverage area at a certain frequency. Other cell base stations use different frequencies to communicate in their coverage area. This introduces complicated call-management issues during handovers. Why is frequency allocation geographically divided by cell area? In other words, why can't all neighboring cells use the same frequency? That way, when roaming, there will be no handovers.
    - The number of frequencies being used is called the reuse factor. Depending on the cell sizes and geometric layouts, the reuse factor might need to vary. Can you suggest reuse factors for the layouts you arrived at in the previous question?
    - You to want minimize the spectrum that you use for communications. Under ideal conditions, do you see any theoretical minimum reuse factor that you can set for an FDMA communication model?
12. [**R06**] A transmitter communicates with mobile receivers on a 900 MHz frequency band. The signal power generated by the receiver is 50 W with unity gains produced by the transmitter-receiver antennas.
    - What is the signal power at the mobile receiver 100 m away assuming line-of-sight communication?
    - What is the signal power at the mobile receiver 5 km away assuming line-of-sight communication?
    - What if communication in both the cases was not in free space, but rather bounced off reflectors?

This page intentionally left blank

CHAPTER 13

Digital Rights Management

Digital multimedia is an advancing technology that fundamentally alters our everyday life. Better devices continue to emerge to capture digital images, video, and audio; better software is continuously appearing to create convincing and compelling multimedia content, which is then distributed through wired and wireless networks. By now, it should be well understood that digital media offer several distinct advantages over their analog counterparts. First, the quality of digital audio, image, and video signals is higher and editing or combining different media types is easy. Second, storage and distribution is very easily accomplished via low-cost memory devices and digital networks, regardless of content type. For precisely these qualities, international standards bodies, such as the International Organization for Standardization (ISO) and the International Telecommunication Union (ITU) have established standards, such as JPEG, MPEG-1, MPEG-2, MPEG-4, and so on, to enable the industry to conform to a universal platform to exchange digital multimedia data. As a result, major media corporations, which include the consumer electronics industry, the digital distribution industry, and content creation houses, such as movie studios, the cable television industry, and the game industry, are making great strides toward the creation and distribution of standards-based media information. Although this media revolution is being welcomed by the digital industry as a whole, the one critical obstacle is the ease with which digital media content can be accessed, copied, or altered, legally or otherwise.

Digital media content can easily be copied and transmitted over networks without any loss of quality and fidelity. Tracking such duplication is virtually undetectable, and can result in serious loss of revenue for the rightful content owners. Although government bodies are setting forth legal guidelines and statutory laws to

punish the perpetrators, it is both difficult and expensive to engage in legal actions. It is, therefore, critical for content owners to take steps to safeguard their content from piracy and illegal duplication. Techniques to provide copyright protection and *digital rights management* (DRM) fall into two complementary classes—*watermarking* and *encryption*.

- Watermarking provides ways to embed into, and retrieve from, the image, video, or audio data, a secondary signal that is imperceptible to the eye/ear and is well bonded to the original data. This "hidden" information can encode the intended recipient or the lawful owner of the content.
- Encryption techniques can be used to protect digital data during the transmission from the sender to the receiver. These techniques ensure that the data is protected during transit. The received data then needs to be decrypted by the receiver. The data can then be in the clear and no longer protected.

This chapter starts by providing relevant historical background on watermarking and encryption, and describing the evolution of methods to perform digital watermarking and encryption. Each class is then described in detail separately. Section 2 deals with watermarking, explaining desirable qualities of digital watermarks, common ways to attack watermarks, and the state of the art in watermarking algorithms for text, images, video, audio, and graphics. We try to highlight both the benefits and drawbacks in each case. Section 3 describes encryption in a similar way by explaining the requirements of media encryption techniques and standard ways of encrypting images, text, audio, and video data. Finally, Section 4 explains digital rights management as it pertains to the media industry and explains a few solutions currently used for music, motion pictures, consumer electronics, and the information technology sectors.

1 HISTORY OF WATERMARKING AND ENCRYPTION

The concepts of watermarking and encryption predate the emergence of digital media, and have been used in many forms as long as open communication channels have existed. Watermarking techniques are a particular type of steganography, which literally means covered or hidden (steganos) writing (graph). In contrast, encryption deals with rendering messages unintelligible to any unauthorized person who might intercept them. Marking documents with secret messages has evolved with human writing from around 4000 years ago. Although the meaning and purpose of the earlier watermarks is uncertain, they might be attributed to practical functions such as identification or authenticity of paper-related documents. The term watermark was coined near the end of the eighteenth century, when paper currency was popularized in commerce, to authenticate various erudite writing, which was often restricted to a privileged set of people and due to the need to conceal messages from enemies. Counterfeiting prompted advances in watermarking technology by the use of color dyes.

Although digital text and signals started being used in the 1960s, it is difficult to determine when the term *digital watermarking* started to get attention. In 1979,

Szepanski illustrated watermarking technology that used machine detectable patterns on documents as a way to control piracy. Later, the 1990s saw the production of much research on techniques to insert and detect digital watermarks in audio, images, and video. After successful digital media distribution using MPEG-1, MPEG-2, and DVDs, several organizations began considering watermarking technology for inclusion in various standards. Today, digital watermarking techniques have many applications, such as copyright protection, tracking and tracing of media objects to identify and authenticate intended recipients, and so on.

Encryption, where the message is rendered unintelligible, has been used for probably as long as people have wanted to keep their communications private. One of the earliest documented reports is that of Spartan generals who wrote their message on a narrow strip of parchment wrapped around a thin cylinder. When the paper was observed in a flat unwound state, the letters made no sense and could only be read by wrapping the paper around a cylinder of the same size. Similar simple, though effective techniques have been used throughout history. For instance, the Greeks have been known to deliver messages tattooed on the scalp of the messenger, making the writing invisible once hair grows. The Greeks were also among the first to use numerical ciphers by having different ways to substitute numbers for letters. A cipher is a mapping, or an algorithm, that performs an encryption. Figure 13-1 shows an example of a simple cipher.

The use of ciphers has been predominant for military communication, where simple substitution-based ciphers were used in the eighteenth century, to modern times where more complicated transpositions, different mediums (such as music), and automated systems have been developed to cipher and decipher information. Automated mechanical encryption methodologies were first developed during World War II. The German Enigma machine used mechanical rotors to perform the encryption, and the encrypted content needed to then be delivered by a messenger, or using radio communications. With the digital age, automated encryption and decryption methods have become commonplace in most sensitive communication transactions, such as e-mail or digital commerce.

Standards around encryption have also been introduced, such as the Data Encryption Standard (DES) developed in the 1970s and the Advanced Encryption Standard (AES) developed in 2002. With the proliferation of commercial digital media, there is now also a need to set up an effective encryption infrastructure that secures the digital distribution of audio, video, graphics, and other related media types.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

This is a book about multimedia

44 32 42 34 42 34 11 21 43 43 52
11 21 43 54 44 23 54 13 44 42 23
51 41 42 11

Figure 13-1 Example of a numeric cipher. The table on the left shows one possible way to code an alphabet. Any sentence making use of that alphabet then maps to an encoded message.

2 WATERMARKING TECHNIQUES

Digital watermarking is the process of embedding/retrieving meaningful secondary information into the multimedia data. The basic problem here stems from the requirements to embed a mark in the digital signal in such a way that it will not reduce the perceived value of the digital object, and at the same time to make it difficult for an unauthorized person to remove. Proper watermarking techniques demand a good understanding of multimedia signal processing, communication theory, and the human visual/audio system (HVS/HAS). To embed watermark information into the original data, watermarking applies minor modifications to it in a perceptually invisible manner, where the modifications are related to the watermark information, as illustrated in Figure 13-2. Perceptual analysis makes use of visual and audio masking processing where faint visible or audible signals become invisible or inaudible in the presence of other dominating frequencies. The watermark information can be retrieved afterward from the watermarked data by detecting the presence of these modifications.

2.1 Desirable Qualities of Watermarks

There are many ways to embed watermarks; some require explicit visibility and some require the watermark to be imperceptible. Although different media types embed watermarks using different means, the common desirable qualities for creating watermarks in digital data are as follows:

- *Perceptual transparency*—In most applications, the watermarking algorithm must embed the watermark in such a way that it does not affect the quality of the underlying host data. A watermark-embedding procedure is truly imperceptible if a human observer cannot distinguish the original data from the data with the inserted watermark.

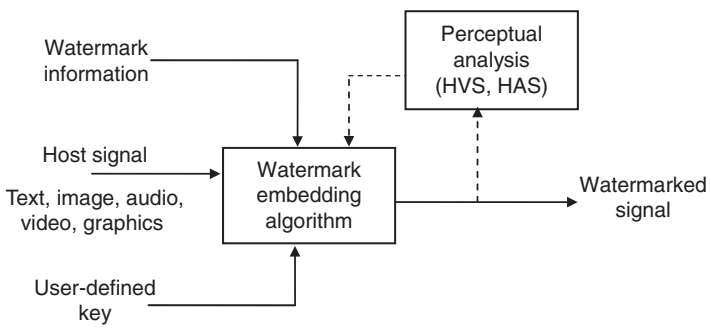


Figure 13-2 Generic watermarking process. Information is embedded in a host signal with a user-defined key. The key is used in a variety of ways to spread or modulate the watermark. Perceptual analysis is often used to evaluate the embedding quality.

- *Payload of the watermark*—The payload measures the amount of secondary information hidden in the watermark. For the protection of intellectual property rights, it seems reasonable to embed a small amount of information, similar to that used for International Standard Book Numbers (ISBNs; roughly 10 digits). On top of this, you could also add the year of copyright, the permissions granted on the work, and the rating for it, or even an image to signify specific properties. However, embedding the watermark should not significantly change the bit rate of the original data.
- *Security*—Most important, the watermark should withstand "attacks" on the data. These attacks are normally manipulations, which might be unintentional, such as noise due to compression and transmission, or intentional, such as cropping, filtering, or even just plain tampering. A watermarking technique is truly secure if knowing the exact algorithms used for embedding and extracting the watermark does not help an unauthorized party to detect the presence of the watermark.
- *Recovery*—The watermark should be recoverable using methods that do not perceivably degrade the original data quality. In some applications, like copyright protection and data monitoring, watermark extraction algorithms can use the original unwatermarked data to find the watermark. This is called nonoblivious watermarking. In most other applications, such as copy protection and indexing, the watermark extraction algorithms do not have access to the original unwatermarked data. This renders the watermark extraction more difficult. Watermarking algorithms of this kind are referred to as public, blind, or oblivious watermarking algorithms.

Note that these criteria are mutually contradictory, for example increasing security would generally need increasing the payload of the watermark. For each application, the proper balance of these criteria needs to be defined for a watermark to be imperceptible yet secure to attacks and still be effectively retrievable during and after communication.

2.2 Attacks on Watermarks

When a digital object is watermarked, any process, whether automatic or manual, that attempts to remove it or alter it in any fashion, intentionally or otherwise, is called an attack. Digital media data is susceptible to various modifications. These modifications almost always change the data stream of the media object. For instance, images are cropped, filtered, and rotated; video data is frequently converted to different formats; and audio data is filtered. All forms of media are compressed for storage and distribution reasons. These modifications, although legitimate, can have adverse effects on the embedded watermarks. Along with legitimate alterations, the introduction of digital watermarks for copyright protection and digital rights management has given rise to intentional attacks where the perpetrator intends to remove and even add watermarks so as to circumvent copyright laws. Most digital watermarking systems aim to provide robust watermarks, which can still be detected after severe legitimate processing.

The goal of intentional attackers here varies, depending on the type of watermark and watermarking scheme used. For instance, in the case of robust watermarks, the attacker's goal here is to remove the watermark, or make the watermark undetectable to digital forensic systems, while still keeping the perceptual quality of the digital media object. Once a watermark has been removed, rightful ownership cannot be established. In other instances of fragile watermarks, where changing or altering the data changes the watermark, the attacker's goal is to maintain the validity of the watermark after data alteration. We are not providing an exhaustive list, but give some common classes of attacks with pertinent examples.

Attacking with uncorrelated noise is a common attempt to destroy watermarks. Here, the noise is distributed among the data samples in a bit plane and tends to weaken the watermark. *Overmarking* is another example of an intentional attack. In doing so, the attacker hopes to damage the initial watermark by embedding a second, additional watermark. For this, the attacker has to know where the watermark is (visible watermark) and perhaps how it has been embedded. Inversion attacks occur when the attacker first attempts to detect the presence of a watermark and find out which bits of the data samples contain the watermark, then removes it. Another class of attacks called iterative attacks are carried out effectively on visible watermarks by altering the data to make it easier to see the watermark, then extracting it, and iterating through an algorithm until it changes, thus creating a version of the original watermarked data having a different watermark. This can be used to claim ownership.

2.3 Watermarking in Text

Using watermarks to preserve the authenticity of the document is not new and dates back at least 2000 years. One factor making text difficult to watermark is that, compared with a photographic image, a text document has very few places to hide watermarks. Many methods have been proposed to watermark electronic text documents. Some alter the physical spacing, syntax, or text character appearances in a way that it is not easily detectable; others change the grammar and text content without changing the meaning. The most common ones used are described in the following sections.

2.3.1 Line Shift Coding

In line shift coding, each even line is slightly shifted by a small predetermined amount either up or down according to the value of a specific bit in the watermark. If a one is to be embedded, the corresponding line is shifted up, and shifted down if a zero has to be embedded. The odd lines are considered to be control lines; hence, they remain untouched and function as references for measuring and comparing the distances. These distances are normally compared between the bases of the lines, or the distances between the centers of the lines. Because the bases of the lines in the original document are uniformly spaced, the distances between the control lines are also uniform (twice that in the original document). It is easy to compare and see whether a one or zero bit is embedded by comparing the distance between the control line and the watermark embedded line. The original document is not needed in this case to detect a watermark.

2.3.2 Word Shift Coding

Word shift coding works in a similar way to line shift coding, by adjusting distances not between lines, but between words on a line. In word shift coding, words appearing in each line are first organized into groups with each group having a sufficient number of characters. The groups can be numbered sequentially. As with line shift coding, the odd groups serve as references for measuring and comparing distances between groups, while each even group is shifted depending on the watermark bit to be embedded. For instance, if a one is to be embedded, the even-numbered group is shifted to the left. If a zero is to be embedded, the even-numbered group is shifted to the right. To extract the watermark, the shift in the groups can be detected. Unlike line shift coding, where the distance between the control lines is fixed, the distance between the control groups is variable in word shift coding and, hence, the original document is required for comparison.

2.3.3 Feature Coding

Feature coding is another common method used to embed watermarks. In this method, prominent features of individual characters used in the text are altered in a specific way to embed a watermark bit. For instance, the horizontal segment in the letter *t* might be extended or reduced, depending on whether you want to embed a one or a zero. Another example includes increasing or decreasing the size of the dot in letters *i* and *j*. Watermark detection can be carried out by comparing these features in the original document with the corresponding features in the watermarked document. Because individual character features are modified to embed watermarks, this method is sometimes also known as *character coding*.

2.4 Watermarking in Images and Video

Media-based watermarking techniques proposed thus far in research and industry exploit the redundancies in the coded image to embed the information bits, and can be broadly divided into spatial domain and transform domain methods. Also, the algorithms can be classified based on whether the original (without watermark) image is used in the watermark extraction process (*oblivious* versus *nonoblivious*). Using the original image in the extraction process provides greater strength to the embedded bits but restricts the uses of such a watermark. An embedding technique that ensures retrieval without the original is, therefore, preferred for wider applications such as embedding captions, annotations, and so on. There is a significant amount of literature on methods used in watermarking. Here, we limit our description to a few representative techniques that are commonly used for their robustness and/or their relationship with standards-related work (such as JPEG/MPEG).

2.4.1 Spatial—Least Significant Bit Modification

This is one of the simplest examples of a spatial domain watermarking technique. If each pixel in a gray-level image (or for each color component in a color image) is represented by an 8-bit value, the image can be sliced up in eight bit planes. Figure 13-3 illustrates these eight bit planes for a sample image. All the eight planes are shown together in the upper left and form the original image. The remaining images show all

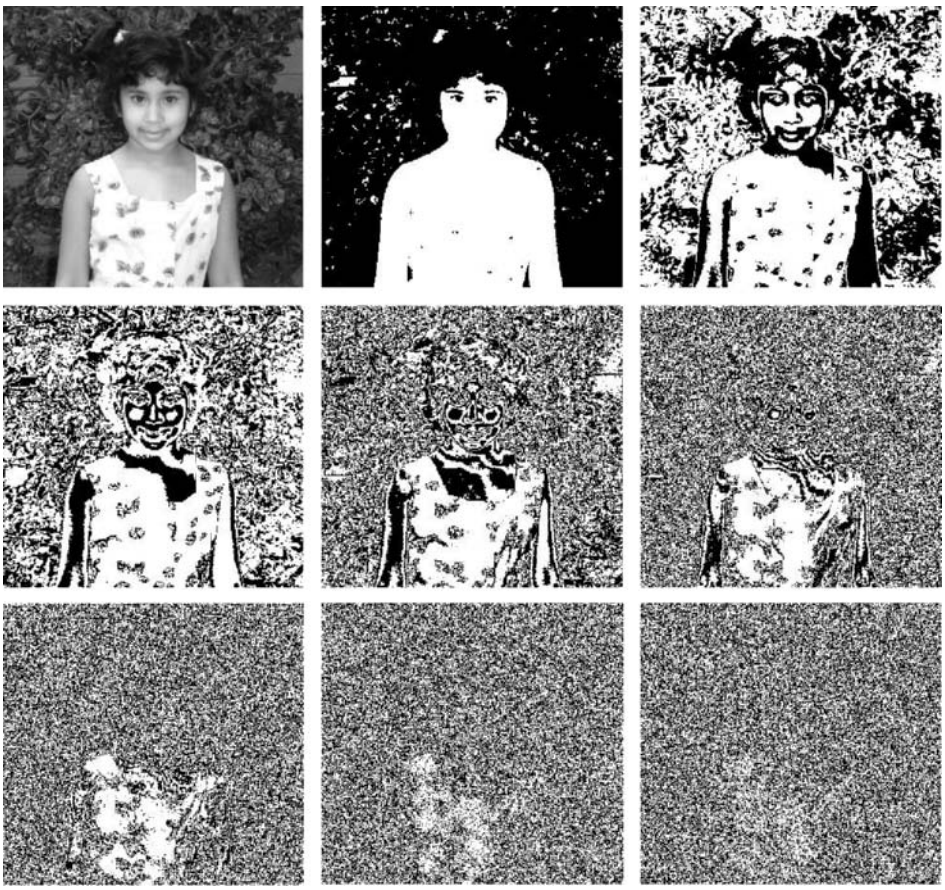


Figure 13-3 Bit planes showing images using each bit. The original 8-bit image is shown in the upper-left corner. Each significant bit image is shown from the upper-middle to the lower-left image. There are eight images corresponding to each bit at every pixel. As the significance of the bit is lowered, the information is less correlated and random.

the individual bit planes with the upper middle image showing the most significant bit plane and the lower left image showing the least significant bit plane.

Because the information in the least significant bit plane is not correlated to the content, it can easily be replaced by an enormous amount of watermark bits without affecting the image perception. One way of inserting a string of bits is to modify every n^{th} bit in row order. A smarter approach is to visually perturb regions in the image bit plane so as to "perceive" a message when looking at the bit plane only. Alternatively, one of the correlation-based mechanisms explained in the next sections can be used to embed information in the bit plane. Although watermarking techniques utilizing bit planes are simple, they are neither very secure nor robust to attacks, as the least significant bit plane is easily affected by random noise, effectively removing the watermark bits.

2.4.2 Spatial—Techniques Using Correlation in the Spatial Domain

Most commonly used techniques in this class work by adding noise to the original image to create the watermarked image. Here, a noisy image is generated using a user-defined key and serves as a watermark. Conversely, given a watermark embedded image, the watermark can be detected by using correlation techniques to find out the presence of the noisy pattern. The noisy image signal that serves as a watermark is not any random signal but has specific properties. It is a binary signal consisting of 0 and 1 bits and has the same dimensions as the host image. Additionally, it is generated in such a way that the average energy is zero, indicating that locally in all areas of the binary watermark the number of ones and zeros are the same. Because of such a distribution, the binary noise pattern remains visually imperceptible when added to the image. On the other hand, when a watermarked image is obtained, correlation techniques between the image and the noise pattern decisively yield whether that noise pattern is present in the image. The embedding methodology is explained in the next few paragraphs.

In practice, these methods use a pseudorandom noisy pattern that can be generated by means of a user-defined key. The pseudorandom noise pattern is a binary pattern consisting of {0,1} and is generated based on a key using, for instance, a seed image or randomly shuffled binary images. The only constraints are that the energy in the pattern should be uniformly distributed, and that the pattern should not be correlated with the host image content. This pseudorandom pattern serves as the watermark signal $W(x,y)$. The values of $W(x,y)$ are mapped to the $\{-1, 1\}$ range to produce a zero energy signal. To generate a watermarked image $I_W(x,y)$ from the original image $I(x,y)$, the pseudorandom pattern $W(x,y)$ is multiplied by a scaling factor and added to the host image $I(x,y)$ as illustrated in Figure 13-4.

$$I_W(x,y) = I(x,y) + k \cdot W(x,y)$$

Authentication of the watermark in watermarked image $I_W(x,y)$ can be accomplished by computing the correlation between the image $I_W(x,y)$ and the pseudorandom noise pattern $W(x,y)$. If this correlation exceeds a threshold, the image $I_W(x,y)$ can be considered to contain the watermark $W(x,y)$. $W(x,y)$ is normalized to a zero mean before correlation. The correlation process can be mathematically expressed as shown:

$$\begin{aligned} R_{I_w(x,y)W(x,y)} &> T \quad \textit{implies } W(x,y) \textit{ detected in } I_W(x,y) \\ &< T \quad \textit{implies } I_W(x,y) \textit{ is not watermarked by } W(x,y) \end{aligned}$$

where

$$\begin{aligned} R_{I_w(x,y)W(x,y)} &= \frac{1}{N}\sum_{i=1}^{N} I_W(x,y)\,W(x,y) \\ &= \frac{1}{N}\sum_{i=1}^{\frac{N}{2}} I_W(x,y)\,W^{+}(x,y) \;+\; \frac{1}{N}\sum_{i=1}^{\frac{N}{2}} I_W(x,y)\,W^{-}(x,y) \\ &= \frac{1}{2}\left(\mu I_W{}^{+}(x,y) \;+\; \mu I_W{}^{-}(x,y)\right) \end{aligned}$$

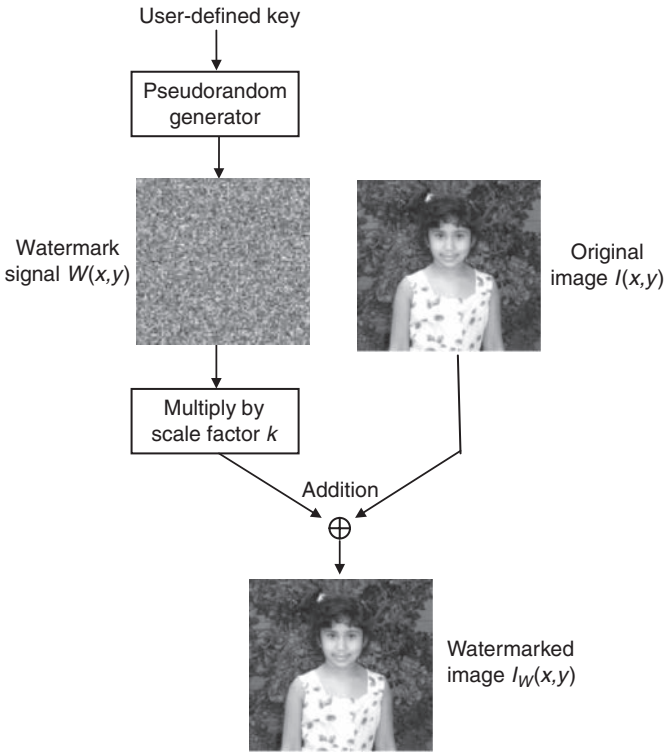


Figure 13-4 Watermarking in the spatial domain. An image pattern having a uniform energy distribution is embedded in the host image.

Here, N is the number of pixels in the image. $I_W{}^+(x,y)$ and $I_W{}^-(x,y)$ indicate the set of pixels where the corresponding noise pattern is positive or negative. $\mu I_W{}^+(x,y)$ represents the average value of set pixels in having positive noise patterns and $\mu I_W{}^-(x,y)$ represents the average set of pixels having negative noise patterns. From the preceding equation, it follows that the watermark detection problem corresponds to testing the hypothesis whether two randomly selected sets of pixels in a watermarked image have the same mean. Because the image content can interfere with the watermark, especially in low-frequency components, the reliability of the detector can be improved by applying matched filtering before correlation. This decreases the contribution of the original image to the correlation. For instance, a simple edge-enhance filter F_{edge} with the convolution kernel shown can be used:

$$F_{edge} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} /2$$

Experimental results show that applying this filter before correlation significantly reduces the error probability, even when the visual quality of the watermarked image is seriously degraded before correlation.

2.4.3 Frequency—DCT Coefficient Ordering

A wide range of watermarking techniques exists for embedding data in the different domains. For example, prior to embedding or extracting a watermark, the original data can be converted from the spatial domain to the frequency domain (Fourier, Wavelet, Discrete Cosine transform) or even the fractal domain. Here, we only discuss the use of the frequency domain obtained by DCT because of its wide use in standards compression algorithms such as MPEG and JPEG. The general method of DCT coding under the JPEG and MPEG standards involves dividing the original spatial image into smaller blocks of pixels, and then transforming the blocks to obtain equal-sized blocks of transform coefficients in the frequency domain. These coefficients are then quantized to remove subjective redundancy in the image. The whole process for image and video compression is described in the chapters 7 and 8. The quantization tables used in JPEG have been arrived at by a study of the sensitivity of the human visual system to the DCT basis images. These studies have also influenced decisions of watermarking techniques in deciding the location/strength of the watermark.

Koch and Zhao proposed one of the first methods to embed information by reordering DCT coefficients. For each 8×8 block, the DCT transform is calculated, resulting in 64 frequency coefficients. The embedding process works by altering these DCT coefficients, if needed, so as to reflect the embedding of a zero or one. However, the alteration should not cause any large perceptual change when the changed DCT coefficients get decoded to re-create the image in the spatial domain. The next logical questions are, then, which coefficients to alter and how much alteration is acceptable.

To that end, HVS studies have shown that the midband frequency range F_M shown in Figure 13-5 is where the human eye is more tolerant to changes. This is so because most natural images have a greater variance in frequency coefficient ranges in

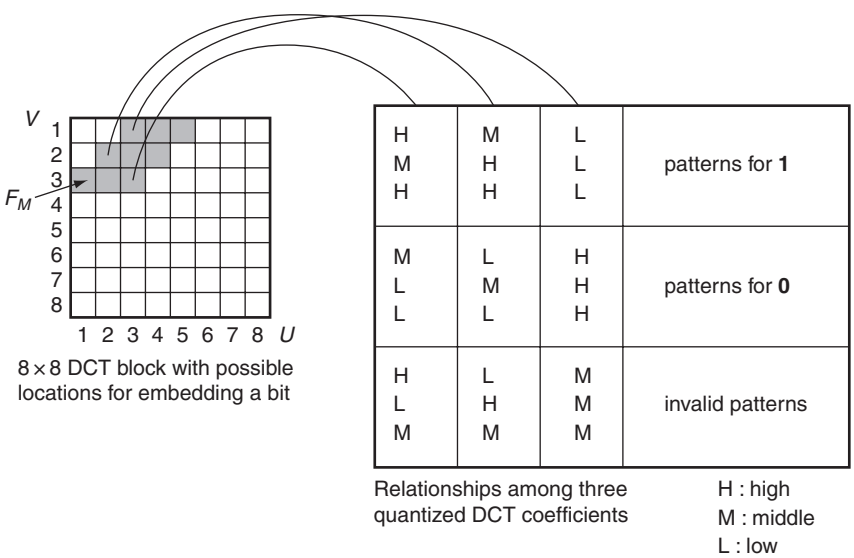


Figure 13-5 Watermarking based on adapting the relationship between three middle frequency DCT coefficients

this band. The core embedding process works by studying the distribution of three coefficients in the midfrequency range. These three coefficients can be chosen pseudorandomly. Once chosen, the distribution of coefficients can follow any one of the nine ordered patterns as illustrated, where H, M, and L stand for high, middle, low numeric values of the coefficients. In other words, one coefficient is going to be highest among the three, one of them will lie in the middle, and one of them will be the lowest. The possible distribution of these high, medium and low values is shown in the table of Figure 13-5. Three of these patterns {HML, MHL, HHL} are reserved to show an embedding of bit "1", where the only requirement is that the third coefficient be the lowest of all three. Three of the patterns {MLH, LMH, LLH} are reserved to show an embedding of bit "0", where the third coefficient is the highest of all three. The other patterns have the third coefficient is in between (or equal to) the first two signifies invalid patterns. These indicate no embedding. Then, given a choice of three coefficients that are psuedorandomly picked, a bit can be embedded as follows:

- *To embed a* 1—Check to see if the pattern falls in one of the {HML, MHL, HHL} categories. If it already does follow the pattern, a one can be considered embedded. If it does not, the values should be altered within a specified minimum perceptual threshold distance to one of the previous patterns to signify the bit 1 being embedded. If it cannot be altered within the threshold, the pattern can be altered to show that it is invalid. The bit 1 in this invalid case is embedded in the next DCT block.
- *To embed a* 0—Check to see if the pattern falls in one of the {MLH, LMH, LLH} categories. If it already does follow the pattern, a zero can be considered embedded. If it does not, the values should be altered within a specified minimum perceptual threshold distance to one of the previous patterns to signify the bit 0 being embedded. If it cannot be altered within the threshold, the pattern can be altered to show that it is invalid. The bit 0 in this case is embedded in the next DCT block.

The order of blocks, which are tested for embedding the watermark, and also the triplet sequence of coefficients, which are used in the decision making, can be chosen in a pseudorandom manner generated by a user-specified key, which makes the process more secure. The detection process works by traversing the same blocks, and the same triplet sequence in each block, checking to see if that sequence shows an embedding of 1, an embedding of 0, or an invalid pattern, in which case the next block in the pseudorandom sequence is analyzed.

The overall algorithm to embed a watermark bit can be described as shown in Figure 13-6. In the figure, the bits to be embedded are $\{b_0,b_1,b_2 \ldots b_n\}$ and B is the set of 8×8 DCT coefficient blocks in which bits are embedded. We start by initializing B to an empty set.

Figure 13-7 illustrates an example of watermarking using this technique0, with a heavily amplified difference between the original image and the watermarked version shown to the left. In general, methods such as this one, which operate on DCT domains, have proved to be robust. They are also proving to be methods of choice because of the large use of DCT in JPEG and MPEG encoding, prevalent in industry now.

To write a bit b_i:

1. *Select a block using a pseudorandom sequence generator.*
2. *If the block already exists in B, go to 1, else add the block to B.*
3. *Try to embed b_i in B by altering a pseudorandomly chosen triplet sequence of DCT coefficients. If the embedding cannot be done, change the pattern to an invalid pattern and go to 1.*

To read a bit b_i:

1. *Select a block using a pseudorandom sequence generator.*
2. *If the block already exists in B, go to 1, else add the block to B.*
3. *Choose a pseudorandom triplet sequence of DCT coefficients. If the pattern corresponds to a 1 or 0, output b_i accordingly. If the pattern is an invalid pattern then ignore the current block and go to 1.*

Figure 13-6 Pseudocode to embed (above) a watermark bit b_i and extract (below) the watermark bit b_i using DCT coefficients. The bits $\{b_0, b_1, b_2 \ldots b_n\}$ are embedded in 8×8 blocks specified by B.

Figure 13-7 Original image, heavy watermarked image using adapting relationships between DCT coefficients and $W(x,y) = I(x,y) - I_w(x,y)$

2.5 Watermarking in Audio

Audio watermarking schemes rely on the imperfections of the human auditory system. Because the human ear is more sensitive than the visual and other sensory systems, good audio watermarking techniques are difficult to design. Among the various proposed audio watermarking schemes, blind watermarking schemes, which do not need the original (no watermark) signal, are more practical. These methods need self-detection mechanisms.

2.5.1 Quantization Method

In this scalar quantization technique, a sample value x is quantized and assigned a new value depending on the payload bit to be encoded. For instance, the watermarked sample y can be represented as follows:

$$y = \begin{Bmatrix} Q(x,I) + I/n, \text{ if watermark bit } = 1 \\ Q(x,I) - I/n, \text{ if watermark bit } = 0 \end{Bmatrix}$$

where $Q(x,I)$ is the quantization function with quantization interval I and n is an integer that qualitatively specifies how much the quantized value can be perturbed to encode the payload bit. This scheme is simple to implement and robust against noise attack, as long as the noise level is below I/n. If the noise is larger, the watermark extractor or detector could misinterpret the watermark bit. The choice of n is crucial for robustness, and is chosen based on the signal to noise ratio, as well as how much perturbation can be tolerated by the human ear, which, in turn, depends on the quantization interval I.

2.5.2 Two Set Methods

Robust audio watermarking can be achieved by observing that the relative mean sample value of a set of audio samples remains nearly constant as the number of samples increases in number. Correspondingly, two sets of samples will tend to have similar means. One way to exploit this observation is by changing the sample sets statistics to reflect the watermark. If two sets are different, we can conclude that a watermark is present. There are two major steps in this technique:

- Given a signal, choose different contiguous sets using a pseudorandom pattern.
- Once sets of samples are chosen, choose two sets and add a constant small value d to the samples of one set and subtract the same value d from the samples of the other set to embed a binary one, and vice versa to embed a zero.

Mathematically, if C and D are two sets, this can be expressed as follows:

$$\begin{array}{llll} C_w(n) = C(n) + d & D_w(n) = D(n) - d & \textit{to embed a one} \\ C_w(n) = C(n) - d & D_w(n) = D(n) + d & \textit{to embed a zero} \end{array}$$

The original sample values are, thus, modified to reflect the presence of a one or a zero. Because the relative mean or expected value of both C and D is similar, $E[C(n)] = E[D(n)]$. The detection process works by finding the difference between the expected values to the two embedded sets, $E[C_w(n) - D_w(n)]$. This is used to decide whether the samples contain watermark information. Because two sets are used in this technique, the embedded watermark can be detected without the original signal, as follows:
If a one is embedded –

$$E[C_w(n) - D_w(n)] = E[(C(n) + d) - (D(n) - d)] = E[C(n) - D(n)] + 2d$$

If a zero is embedded –

$$E[C_w(n) - D_w(n)] = E[(C(n) - d) - (D(n) + d)] = E[C(n) - D(n)] - 2d$$

Under the assumption that the means of two sets C and D are similar, their expected values are the same. This reduces the preceding expressions to $+2d$ if a one is embedded or $-2d$ if a zero is embedded.

2.5.3 Spread-Spectrum Methods

Spread-spectrum methods embed pseudorandom sequences in the signal, and detect watermarks by calculating the correlation between pseudorandom noise sequences and watermarked audio signals. Similarly to the image domain spatial correlation

techniques described in Section 2.4.2, a pseudorandom sequence is inserted into the signal or signal parts. The pseudorandom sequence, which is a wideband noise signal, is spread either in the time domain or the transform domain of the signal. The binary watermark message consisting of the string of zeros and ones is converted to a bipolar message of $\{-1,+1\}$. This bipolar message is modulated by the pseudorandom sequence $r(n)$, which is frequently generated by means of a key. A modulated watermark $w_b(n) = br(n)$ is generated for each bipolar bit b. Hence, $w_b(n)$ can be either $r(n)$ to embed a 1 or $-r(n)$ to embed a zero. The modulated watermark is then added to the original signal $s(n)$ to produce the watermarked signal $s_w(n)$ as shown next. Here, α serves as a scale factor whose value is determined by psychoacoustic analysis.

$$s_w(n) = s(n) + \alpha\, w_b(n)$$

2.5.4 Replica Method—Echo Data Hiding

In this class of techniques, the original audio signal, or part of it, is used as a watermark and embedded into the original signal in the time domain or frequency domain. Hence the name—replica—which suggests that a properly modulated part of the original signal is embedded in itself. The detector can also generate the replica from the watermarked audio and calculate the correlation between this and the original in the time or frequency domain to ascertain whether the replica is embedded. An example of this method is echo data hiding. Here, part of the original signal is embedded into the original signal by introducing an echo in the time domain. If $s(t)$ gives the sample value at time t, echo hiding produces a signal.

$$x(t) = s(t) + \alpha \cdot s(t - td)$$

Here, t_d is the delay offset, as illustrated in Figure 13-8.

Binary messages can be embedded by echoing the original signal with two different delays—t_d1 is used to embed a 0, and t_d2 is used to embed a 1. The detector can

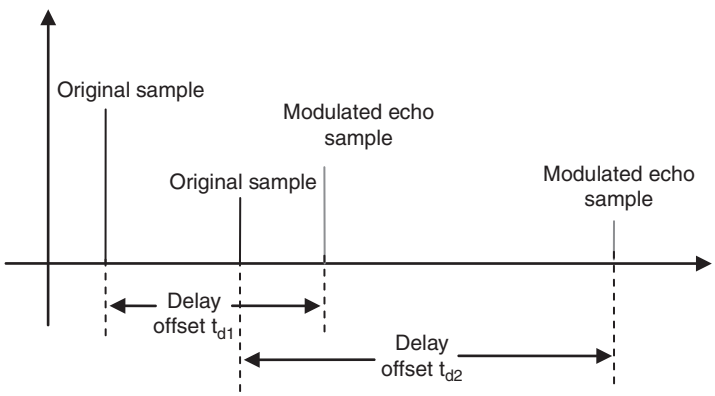


Figure 13-8 Echo hiding. Two original samples are shown added to the original using delay offsets after modulating their amplitudes. The delay offsets used differ depending on whether a zero or one needs to be embedded.

extract the watermark by performing an autocorrelation between the current sample and samples in the past to find the delay offset t_d, which suggests the value of the embedded binary bit. Echo hiding is normally not perceptible to the human ear and frequently makes the sound rich. These methods have also been used for coarse synchronization between audio signals. The disadvantage, however, is its higher autocorrelation complexity during detection. Though complex, detecting echoes is not a hard function and, hence, anyone can detect the watermark without any prior knowledge, exposing the techniques to easy attacks.

3 ENCRYPTION TECHNIQUES

Encryption techniques complement watermarking techniques. Rather than hiding information in the data, encryption protects the information by rendering it unintelligible and unviewable. Traditional encryption schemes, also termed as classical or *hard* encryption such as those used in the Data Encryption Standard (DES) and Advanced Encryption Standard (AES) were designed to allow encryption of small amounts of data in transactional applications. For instance, credit card information, online banking, and even secure e-mail communication are the most common secure applications.

Conventional encryption systems work by specifying a secret key known to the sender and receiver only. This key is used in conjunction with a ciphering/deciphering algorithm to encrypt messages. This is somewhat fragile, as anyone finding the key can then access all the data. Instead, modern encryption schemes use a public key and the Public-key Infrastructure (PKI) with digital certificates. Unlike private key encryption, public-key cryptography allows users to communicate securely without the need to access a common shared key. Rather than one key, the encryption work using two designated keys—a *private* key and a *public* key, which are mathematically related but one cannot be deduced from the other. The private key is kept secret while the public key is widely distributed. Figure 13-9 describes the secure encryption and decryption. As shown, person A generates a public key and a private key. The private key is kept with A and not distributed. The public key is distributed to B. Similarly, B generates a private key and a public key. The private key is kept with B while the public key is available to A.

When B wants to send a secure message to A, B encrypts the message using A's public key. A decrypts the message using his private key. Anyone can encrypt using the public key but only the private key can decrypt the public-key encrypted messages. Conversely, when A wants to send a secure message to B, A encrypts the message using B's public key, which is available to A as part of a digital certificate from the key-generating authority. This message encoded by A for B using B's public key can only be decoded by B's private key, which is available only to B. Today, the allotment of such digital keys, their distribution to trusted parties, and their authentication when used is handled by the Public-key Infrastructure (PKI).

The amount of data involved in the text-based encrypted communications, such as financial transactions, secure e-mail, and so on, is small. This makes the encryption performance less important in the overall communication performance. Additionally, the order of data arrival using packet-networked communication is not critical because

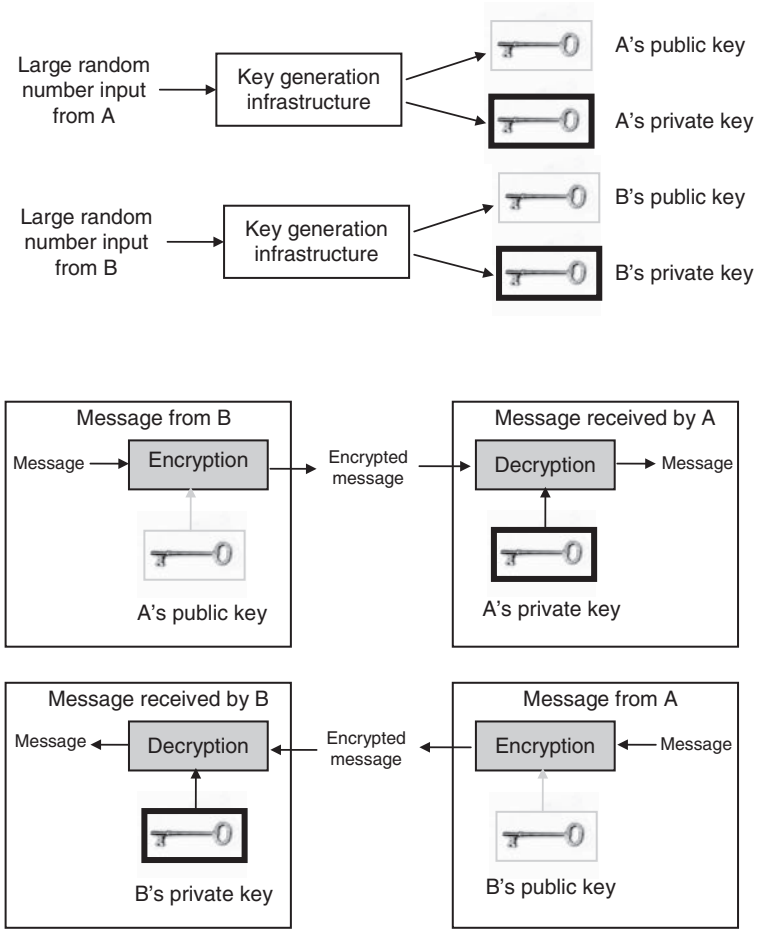


Figure 13-9 Public and private key usage. Person A generates two keys, a public key, which is distributed to B, and a private key, which is kept secret with A. B sends secure messages to A using A's public key to encrypt. A can decrypt this with his private key. Messages sent by A to B are encrypted using B's public key and can be decrypted by B using his private key. This ensures secure communication.

communication does not take place in real time. With regard to multimedia data, files are much larger and also have real-time needs. The standard encryption techniques mentioned previously, if used on such large files, would have nonscalable complexity in time and computation power, thus preventing them from being of any practical use. This challenge is further magnified with the real-time needs that come with streaming multimedia data. Here, content might not only have to be encrypted in real time, but also as content is being viewed in a stream, the data has to be decrypted, decoded, displayed, and then reencrypted if it is stored prior to displaying the next media segment. Besides, the desired real-time interactivity of fast-forwarding or rewinding media further complicates the

situation. Unlike traditional methods, where the whole file needs to be encrypted, encryption of media data is termed as *soft* or *selective* encryption to differentiate the process from the classical *hard* encryption. Such schemes do not strive for maximum security, and trade off security for computational complexity. For example, real-time encryption for an entire video stream using classical ciphers requires much computation time because of the large amounts of data involved; on the other hand, many multimedia applications require security on a lower level. Therefore, the search for fast encryption procedures specifically tailored to the target environment is necessary for multimedia security applications.

3.1 Desirable Qualities of Encryption

Similar to watermarking requirements, encryption methods also have to ideally satisfy the following criteria:

- *Visual acceptance*—Encryption does not necessarily require that all of the information be made unintelligible, and part of the information might be visible. But, the encrypted data should look noisy (in the case of images or video), or it should not be clearly perceivable (in the case of audio). This is the opposite of what watermarking should do.
- *Selective or partial encryption*—Although the whole bit stream can be encrypted, encryption should be able to operate on selective sets of data. Normally, in the case of video streams, encryption occurs after compression that generates frame dependencies in a standardized bit stream format (MPEG-2, H.264). These bit streams needs to be parsed, and frames can be dropped because of real-time needs. The encryption here should work on selective parts of the bit stream, independent of other parts.
- *Time restrictions*—Unlike hard encryption schemes, which are very secure but also are computationally complex, encryption for media calls for efficient processes. Complete security is sacrificed for time.
- *Constant bit rate*—Encryption should preserve the size of the bit stream. This is most suitably required for video/audio where even a few added extra bits at regular intervals could cause the bit stream to no longer adhere to CBR requirements for which it was compressed.
- *Bit stream compliance*—The encryption step should produce a *compliant bit stream* according to the chosen format definition.

3.2 Selective Encryption Based on Data Decomposition

Many compression algorithms for digital media data decompose the original signal into a number of different parts. For instance, region-based compression in images and video compresses different regions using varying parameters. The JPEG and MPEG standards use transform coding techniques based on the Discrete Cosine transform (DCT), producing coefficients for the DCT basis functions. Similar analysis follows in most audio compression algorithms for speech and wideband audio. Rather than encrypting all the coefficients, which has large time complexity, encrypting selective ''important'' coefficients is more practical for the needed time restrictions.

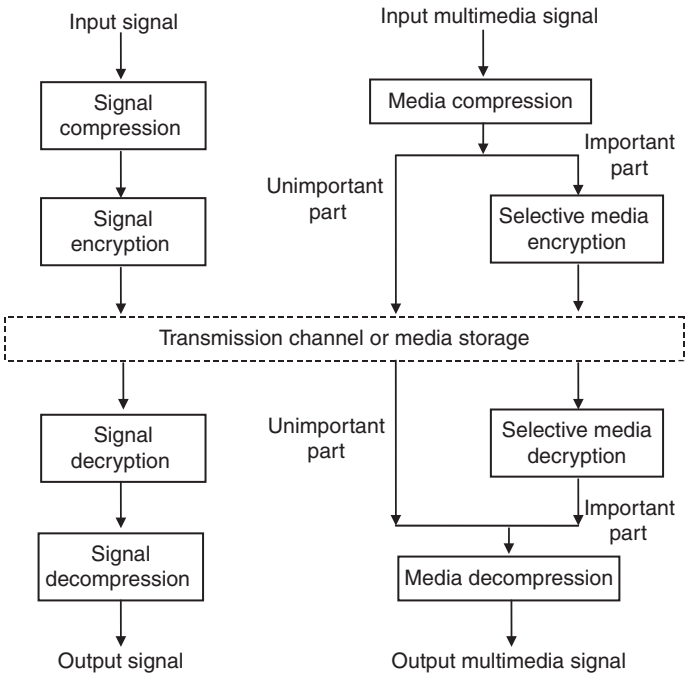


Figure 13-10 Standard encryption versus selective encryption. In selective encryption, only the important parts of the encoded media signal are encrypted.

In multimedia, real-time and secure media communication is almost impossible without this separation. The important part is transmitted after encryption, while the unimportant part is transmitted in parallel without any change, so that the encryption time is negligible. Figure 13-10 shows an illustration of this approach.

The next question for any partial encryption algorithm is to decide which are the important parts in image, video, or audio segments. Compression algorithms tend to reduce redundancy by quantizing coefficients that are not perceptually important. Some algorithm might decompose the input signal into statistically uncorrelated parts to obtain efficient encoding. The same criteria can be used to separate the important and unimportant parts, which are noncorrelated, for encryption reasons. This is a good guard against cipher attacks because the noncorrelated unimportant part cannot be taken advantage of to decipher the important part. The next sections describe encryption techniques on important and unimportant parts for images, video, and audio.

3.3 Encrypting Images and Video

Like watermarking, the encryption techniques can operate in the spatial domain of pixels as well as in the frequency domains. A naïve algorithm would be to encrypt the entire stream—the entire MPEG stream, which allows for security but imposes a heavy computational burden on computing the encrypted stream and decrypting it. This is not favorable for real-time implementations. The following sections discuss a

few techniques that are representative of the spatial and frequency domains. The latter are important for real-time architectures that involve distribution of large media data (video, audio) using MPEG-based compression.

3.3.1 Selective Bit Plane Encryption

Here a "selective" portion of the data set is encrypted. In the case of images using this class of techniques, a selected bit plane that corresponds to, say the most significant bit, is chosen for encryption. A very effective method to encrypt a binary image (bit plane) consists in mixing image data and a message (the key in some sense) that has the same size as the image: A XOR function is sufficient when the message is only used once. With this approach, no distinction between bit planes is introduced, although the subjective relevance of each bit plane might not be equal. As can be seen from the images shown in Figure 13-11, at least four to five bit planes need to be encrypted

Figure 13-11 Encryption of image bits planes. The original is shown on the top. Images in each row show successive bit planes encrypted. The middle row shows 2 bits encrypted, 3 bits encrypted, and 4 bits encrypted. The bottom row shows 5 bits encrypted, 6 bits encrypted, and 7 bits encrypted.

before the degradation becomes visible. In theory, the simple spatial domain method is relatively robust to attacks because it encrypts bit planes that contain nearly uncorrelated bit values.

3.3.2 Encryption in the MPEG Domain

In the MPEG domain, the voluminous data in most multimedia applications involving video and audio puts a significant burden on the encoding process. Fast encryption/decryption algorithms that are secure are required. Most of the methods proposed for use here normally encrypt the MPEG stream only partially or selectively. The encryption process also has to make sure that, wherever required, bit stream compliance to the MPEG family is maintained and the bit rate remains unchanged.

Early work on encrypting video encrypted all the I frames in the MPEG stream. The understanding was that because P and B frames depend ultimately on I frames, the inability to reconstruct I frames meant that you could not reconstruct P and B frames either. However, the P and B frames may contain I macroblocks, which do not depend on I frames. You could reconstruct the I blocks and then use them to further reconstruct partially, though not accurately, other parts of the frames. One way to get around this is to also encrypt I macroblocks. The I frames occupy around 40%–50% of the bit rate. Adding more data to encrypt only increases the time required, which is not desirable. Because all the I frames and I macroblocks in the predicted frames are made of DCT coefficients, an encryption system could encrypt not all, but only some of the coefficients of each block. Experimental results show that encrypting only the DC coefficient, which normally has the most energy, or a few AC coefficients is not always sufficient because the content still contains sufficient information by decoding the unencrypted coefficients, as shown in Figure 13-12.

Another effective way to selectively encrypt is not to encrypt selective coefficients but to encrypt selective bits of all coefficients in the blocks. This method has been shown to make images and video frames less recognizable.

3.4 Audio Encryption

Digital audio content needs confidentiality during transmission in various applications. A good example of this occurs in the music industry because digital music has high commercial value. Record companies are hesitant to release music in standard digital formats like MP3, wav, and AAC because of the potential for piracy. Most secure music distribution sources work by assigning each user a personal key or passport, which can only unlock songs purchased by that specific user. Also, various voice communications either over the Internet (VOIP) or cell phones might require a substantial level of security. Here again, naïve approaches such as using DES or AES, where the whole bit stream is encrypted, might not always work. For instance, most audio end devices are small, such as a cell phone or portable MP3 player, and do not have the power to perform efficient decryption in the required time.

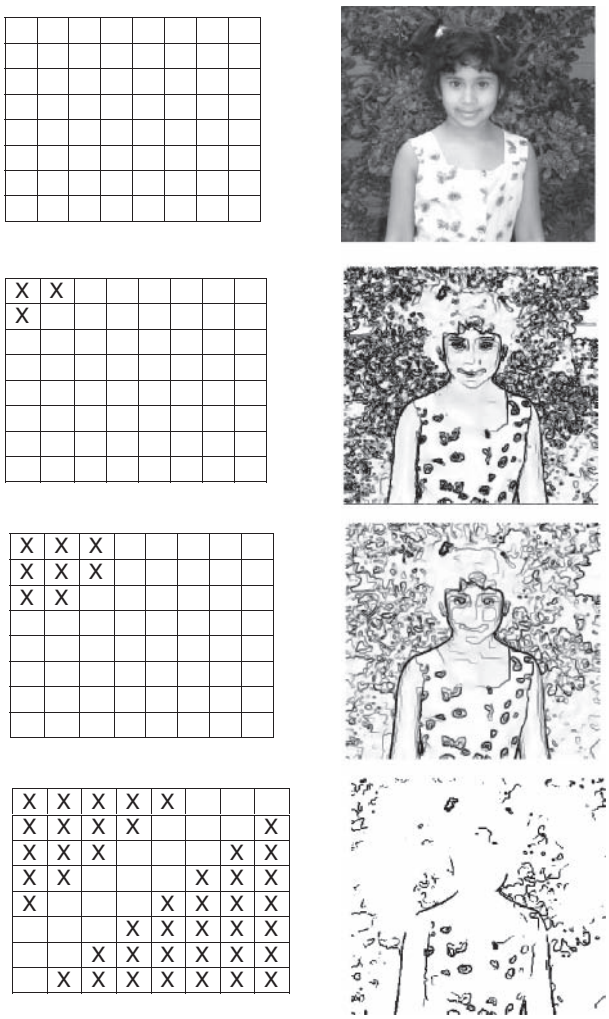


Figure 13-12 Encryption using select DCT coefficients. The left column shows the frequency coefficients that have been encrypted with an "X". The right column shows the decrypted decoded images. During the decoding, the encrypted Xs were not used in each block's reconstruction.

3.4.1 Encryption of Compressed Speech

Traditional speech scrambling has been used in the last century where analog and even digital speech signal samples were permuted in the time domain or distorted in the frequency domain by applying filters and inverters. However, given today's computational power, these methods have become insecure. Most of the speech communication today uses compressed formats such as the G.723 and others based on linear

predictive coding techniques mentioned in Chapter 9. The encoder encodes the signal in a closed loop fashion by incorporating the decoding unit, which synthesizes a speech segment according to excitation parameters computed by the prediction methods given a codebook. These excitation parameter coefficients are changed until the synthesized speech is close to the input speech, within an acceptable tolerance. The coefficients along with the codebook are sent to the decoder to decode speech. Rather than encrypting all the data that is generated by the encoder, selective encryption of some coefficients and code vectors or their most significant bits has proved useful. The important parts for which selective encryption can be applied in this case has been identified to be the codebook indices, the lag of the pitch predictor, codebook gain vectors, and so on.

3.4.2 Encryption of Compressed Audio

Selective encryption for audio streams can also be effectively performed in applications that need the audio data to be protected. Perceptual audio coding methods are commonly used to encode audio, the most common format being the MP3 or AAC. The encoders here use perceptual masking effects to remove the redundant frequencies from the raw signal. The remaining important hearable frequencies, which keep changing from audio segment to segment, then undergo normal entropy coding. Rather than encrypt all the compressed data, the frequency coefficients are divided into different layers called the audio quality layers. Only higher-quality coefficients need to be encrypted. For instance, the 20 Hz–4 KHz range represents the lowest audio quality. Higher ranges have more audio quality. These audio quality layers can be explicitly identified and ranked. The coefficients of contiguous segments can be grouped by layers into blocks of equal size, which are then ciphered. The encrypted data can then be inserted into the MP3 stream without affecting the bit stream conformance.

4 DIGITAL RIGHTS MANAGEMENT IN THE MEDIA INDUSTRY

In today's digital entertainment world, multimedia content is made available for consumers through a variety of channels, such as DVDs, digital cable television, wired and wireless channels enabling Internet access, and so on. Information technology and tools have also advanced far enough to expose all these channels to a variety of risks that involve illegal copying of the digital media, as it makes its way from the source into our homes. Protection of multimedia content, therefore, becomes a critical element, with a need for robust solutions. The following major industries have a vested interest:

- Music industry
- Motion picture industry
- Consumer electronics (CE) industry
- Information technology (IT) industry

The content owners of the music industry include music studios and individual artists who are now able to sell their songs online. For motion pictures, the content owners are the studios. Their content (movies) is displayed or recorded on devices manufactured by consumer electronics companies. The information technology industry manufactures general-purpose computing devices, such as personal computers, which can also be used to display and store content. In this section, we give an example of how DRM has been effectively addressed in each of these industries.

4.1 DRM Solutions in the Music Industry

One of the first companies to distribute digital songs using MP3 was Napster. However, it was the target of numerous lawsuits because the music files that it traded freely were copyrighted and Napster made no attempt to make use of any content rights management in its distribution. Eventually, it was ordered to shut down its operation. It became clear, though, that people were willing to pay to listen to digital music and welcomed the idea of paying for music entertainment song by song rather than being forced to buy an entire album. Following Napster's story, many other companies are now successfully distributing music with the appropriate digital rights management technologies.

One example is Apple Computer's iTunes technology. iTunes is a proprietary digital media player application that allows users to listen to, organize, download, and buy music files as well as video files. iTunes is also an interface to manage musical content on its popular iPod music players and the iPhone. Alongside, Apple has also created an iTunes Store via the Internet to host media created by artists and where users can go to purchase and download media content, including music, television shows, iPhone applications, iPod games, movies, and so on. Much of the content hosted in the iTunes Store is copy protected with Apple's own DRM solution called *FairPlay*. FairPlay has been used to control the listening of musical tunes on computers as well as Apple Computer's mobile devices such as the iPod and the iPhone. Apple has adopted the AAC format to encode musical tunes for two primary reasons. First, it has better quality and compression ratios compared with the former MP3, and second, it provides an open mechanism for companies to extend the format using their own DRM implementation. Apple has made use of this to successfully create a system that manages access to billions of songs sold to many users, while controlling the digital rights appropriately and keeping things simple and flexible.

Though FairPlay manages digital rights of media played on all devices authorized by Apple, it does so in a relatively less stringent manner for portable players on the iPod and iPhone than for computers. This is done so as to make the portable devices as simple and usable as possible. Apple's DRM process is illustrated in Figure 13-13 and Figure13-14. It works as follows:

- Before a user buys content from the iTunes Store, he has to create an account with Apple's servers and then authorize the iTunes player to play on a PC, MAC, notebook computer, and so on. During this authorization process, iTunes creates a unique ID number for the computer it is running on, and then sends it

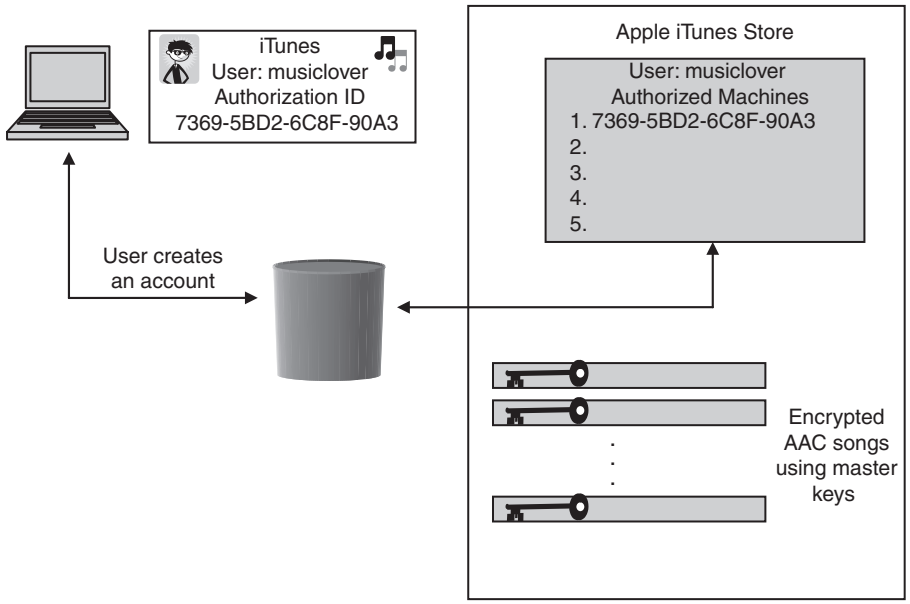


Figure 13-13 DRM in iTunes. The user uses his machine to create an account with the iTunes Store and stores up to five computers on which he can play music that he buys.

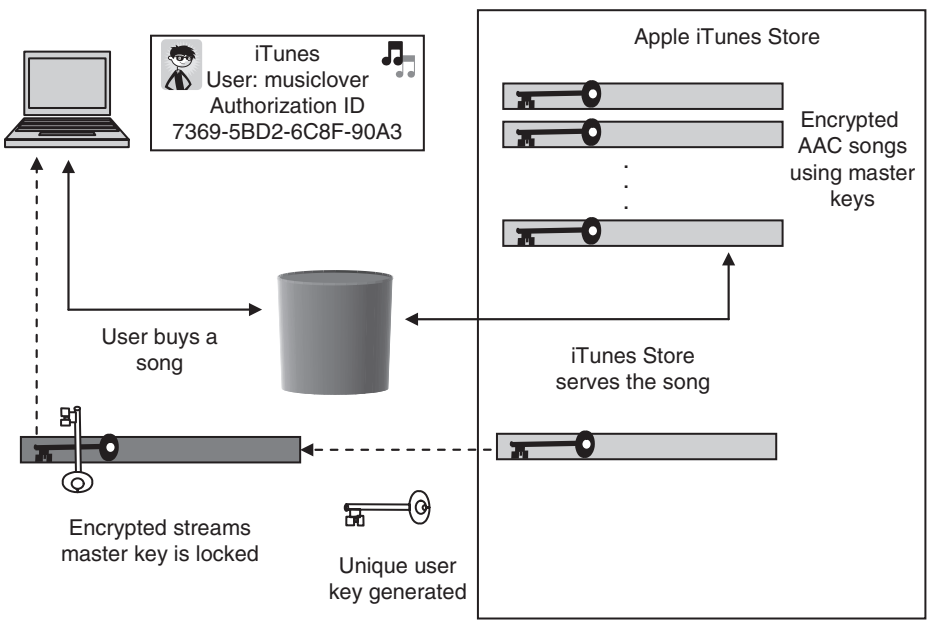


Figure 13-14 DRM in iTunes. When the user buys a song, a unique user key is generated that locks the master key of the encrypted song. The keys along with the encrypted song stream are delivered to the user's machine.

to Apple's servers, where it is stored under the user's account. The user can do this for up to five different computers, thus authorizing any content he buys to play on up to five different devices.

- Songs are stored in the iTunes Store as encrypted AAC bit streams. The AAC song is encrypted using a separate master key, which is stored along with the encrypted AAC stream.
- When a user buys a song from the iTunes Store, a user key is created for the purchased file. The master key is locked using the user key, both of which are sent to the user machine running iTunes. So all songs that a user buys are always protected and locked within iTunes.
- To play a purchased song, iTunes first gets the matching user key to unlock the master key stored within the song file, which is then used to unscramble the song data. All this happens when a song is played in the player, thus minimizing all the computation DRM efforts at serving time.
- A new user key is created every time a new song is purchased. All keys are encrypted and stored on the authorized iTunes computer and also backed up on Apple's servers that host user account information. That way when a new computer is authorized, all the encrypted user keys can be downloaded and the new computer will now be able to play any purchased songs by the user.
- It is also not necessary to lock a computer or device to one account. Many users can create individual accounts and have the same computer or device authorized for those accounts. For every account, iTunes maintains a set of user keys—one for each song purchased.

Thus, all content is stored in an encrypted AAC file in the iTunes Store. It is purchased and downloaded in an encrypted format. It is only decrypted by the iTunes player at the time of playing. The encryption uses master keys and user keys. The multitier keys and encryption used in FairPlay, therefore, ensure security of playing music on Apple's media player on computer platforms. Also this structure ensures that even if the system is cracked, it still limits the damage.

Although Apple's DRM process limits the number of computers on which a purchased song can be played on to five, there is no such restriction on the number of iPods that can be used. Any number of iPods can be used with an authorized computer running iTunes. When an iPod is connected to a computer, it downloads all playable tracks, which include unprotected songs and protected songs with their user keys. The iPod is computationally simple, making no decisions of which songs it should play, and it simply plays all that is downloaded to it, maintaining the decision making to the iTunes interface on the larger computer. This again puts the burden of DRM decisions on iTunes and makes the iPod a simpler and reliable user experience. However, because all control is overseen by iTunes, there is no way for an iPod to download multiple songs from different iTunes libraries. It can synchronize up to only one iTune library at a time. For the latest version of iTunes though, you can now add an iPod as a device that can be authorized to your iTunes account, which allows it to download songs with keys from all five computers that you have registered with your account.

4.2 Encryption in the DVD Standard

The DVD consortium was started as an ad hoc group in 1995 with a few founding members, which included Hitachi, MEI, Mitsubishi, Philips, Pioneer, SONY, Thomson, Time Warner, Toshiba, and JVC. One of the important issues of the DVD forum was to propose and implement a methodology to secure audio/video media content prior to distributing movies on videodiscs. Although several proposals were reviewed, the DVD consortium chose the system developed by MEI and Toshiba, known as the Content Scramble System (CSS). The system consists of a private encryption module with multilayer key management, as illustrated in Figure 13-15. Every DVD disc is encrypted with a set of disc keys that encrypts the data and identifies the disc. Every player is coded with a small set of keys known as *player keys*. A DVD player manufacturer has to buy the DVD license to produce a legal player that can properly decrypt encrypted DVDs. A total of 409 keys have been created for use by different producers and manufacturers. The encrypted content cannot be delivered to the DVD player unless both sides are authenticated with each other with keys on both sides matching. There are many keys, as follows:

- *Region key*—The region key is used to ensure that encrypted DVDs for a region can only be played by DVD players manufactured for that region and vice versa.
- *Authentication key*—This key is used to authenticate both the DVD and the player, ensuring that both devices are licensed.
- *Session key*—The session key is used to encrypt data en route from the player to the host display device.
- *Player key*—The DVD Copy Control Association has allotted a total of 409 keys to be given to each manufacturer. Manufacturers, therefore, have a unique key(s) for all the DVD players it is manufacturing. For example, Panasonic has its own player key, SONY has its own player key, and so on.
- *Title key*—This is used along the sector key to encrypt data.
- *Sector key*—This is used along with the title key to encrypt data.
- *Disk key*—The disk key is used to encrypt the title key.

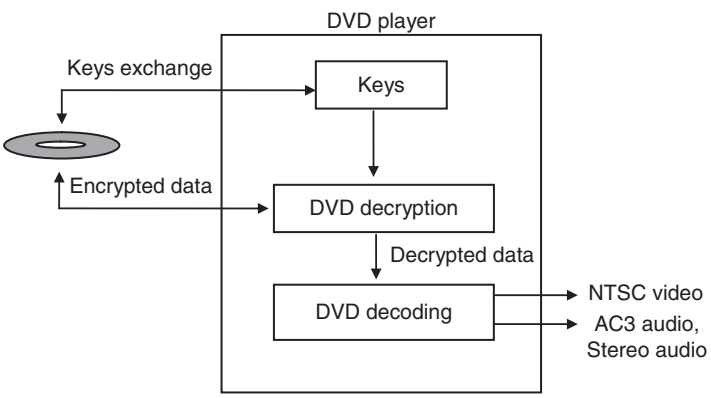


Figure 13-15 Content Scramble System on a DVD player

The DVD player follows the following sequence:

1. First, the DVD and the player check each other's licenses and perform a mutual authentication.
2. Next, the DVD player decrypts each encrypted disk key until a valid disk key matches up with one of the player keys.
3. The title key and the matching disk key are sent to the player. Each title key is decrypted by the disk key.
4. Each sector key is used to decrypt the sectors along with the appropriate title key. The sector data is then sent to the viewing client.

The content is no longer secure once it has been decrypted and decoded, for instance when connecting the output signal to a television for display. The first-generation players only had NTSC (analog) outputs only. The NTSC signal is insecure but could not be copied onto VHS recorders because of the analog protection system (APS) in VHS players developed by Macrovision. APS results in degradation of unauthorized content. As digital content became popular along with the growth of digital devices, DVD players, Digital Video Recorders (DVRs), PCs, Digital Televisions (DTVs), and so on, a more comprehensive framework was needed to secure digital content that would now be hosted and exchanged on multiple devices. The next generations of DVD players have digital video and audio outputs delivered to different digital devices for display, recording, and so on. This prompted the development of the Digital Transmission Content Protection (DTCP) to protect audiovisual digital content from unauthorized copying as it is transported over high-performance networks, devices, and device interconnections.

For protecting the next generation of DVD players and their digital outputs to home devices that receive, exchange, and store digital data, DTCP defined a cryptographic protocol to secure audio/video entertainment content from unauthorized copying, intercepting, and tampering as it traverses digital transmission mechanisms. The DTCP standard is also called the 5C standard because it was developed mainly by five companies—Intel, SONY, Hitachi, MEI, and Toshiba. It is licensed to device manufacturers by the Digital Transmission Licensing Administration authority (DTLA) and can be implemented on any device that streams and stores data through a compatible interface such as the IEEE 1394 serial interface between DVD drives. Figure 13-16 depicts sample ways of how DVD content is protected in home networks. Every digital device that takes digital outputs from DVD players may not necessarily perform encryption to protect data stored on it. For instance, some DVRs may encrypt data for storage/recording using the Content Protection for Recordable Media and Pre-Recorded Media (CPRM/CPPM), while other DVR manufacturers may not offer encryption on their DVRs. Furthermore, the data is communicated from the DVD after CSS decryption to the DVR, before CPRM encryption needs to be secured by DTCP.

Another aspect of digital signals is high-definition video. The DTCP protocol on the IEEE-1394 interface works well to transmit secure content up to 400 Mbps. With the rapidly dropping price of high-definition equipment, higher-definition signals need to be secure.

A common way to deliver high-quality, high-bandwidth digital video between instruments is using the Digital Video Interface (DVI). This high digital quality caused

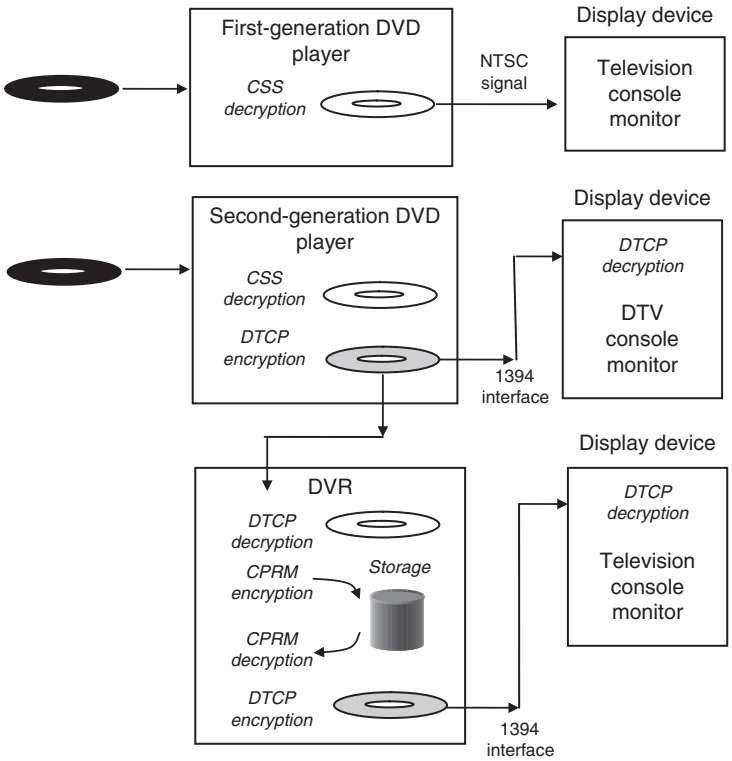


Figure 13-16 Three copy protection systems for DVD protection

concern for content owners because anyone who can access the DVI signal can also illegally copy and distribute content. To curb this, the High-bandwidth Digital Content Protection (HDCP) standard was developed by Intel and Silicon Image in conjunction with the major Hollywood studios. Similar to DTCP, HDCP works by adding secure circuitry within the DVI connection on both the transmitter (for example, DVD players, set-top boxes, high-definition cameras) and the receiver (for example, projectors, HDTVs, LCD displays). This encrypts video in transit across the DVI interface.

4.3 MPEG-4 and IPMP

The MPEG community has developed standard formats to distribute audiovisual data at effective bandwidth. The MPEG-1 format was designed for VCDs, whereas MPEG-2 was used for digital television. For the most part, the data delivered in these formats was not made secure. MPEG-2 does have support for limited DRM in the form of conditional access support, which is reasonably interoperable though not internationally. MPEG started its media security effort by developing Intellectual Property Management and Protection (IPMP) in MPEG-4. MPEG formats are the industry's standard of choice because they support a high level of interoperability apart from making use of effective compression technologies. However, specifying security amid

the desire to be interoperable is not easy. Unlike security implementations in DVDs or Windows Media Player, where the specifications are proprietary and there is limited or no interoperability, the goal of the MPEG world is to provide interoperable security by supporting open and public security implementations. MPEG-4 and its IPMP architecture is defined in more detail in the MPEG-4 chapter discussed in Chapter 14, but a few salient points are mentioned in the following list:

- The MPEG-4 IPMP architecture is message based using a communication interface.
- IPMP in MPEG-4 supports the use and design of defined IPMP modules that perform functions such as authentication, encryption, decryption, watermarking, and so on. Each IPMP function is implemented by a unique module that ultimately produces its own elementary bit stream with unique IDs. This makes the security implementation independent from any content or implementation dependence.
- The MPEG-4 IPMP architecture supports mutual authentication of and between tools.
- All tools produce elementary bit streams (similar to audio and video graphics in MPEG-4), with its own IPMP descriptors. These streams get multiplexed and saved in an MPEG-4 file or transported over digital networks.

4.4 Digital Cinema

Cinema has been traditionally distributed on film. Film is poised to be replaced by digital images that have equal or superior quality. The principal players involved in converting the film-based distribution to a digital distribution are the studios and distributors, such as FOX, SONY, Paramount, and so on, the Digital Equipment manufacturers, delivery providers, standards organizations, and so on. Foremost among these are the organizations and companies working to provide digital security and conditional access in the distribution chain. In this section, we explain the security aspects of the digital cinema distribution pipeline. Figure 13-17 shows a broad overview of the digital cinema distribution. A digital feature film can be sent to the theater in a number of possible ways, including physical media such as a DVD-ROM, satellite broadcast network, or local and wide area networks. Real-time transmission of the media files is not required here because the screen server captures, stores, and plays back the media. To prevent theft and interception during transmission, the digital cinema distribution chain uses watermarking and encryption in a variety of ways. First, the images delivered by the studio are compressed, watermarked, and encrypted using private and public keys. This encrypted content can then be delivered to servers located at different theaters using a conventional physical medium, such as a digital disc, or transmitted via satellite digital networks, or other cable, local, and wide area networks. Once received at the theater unit, the play to screen server decrypts and locally stores the data for displaying later. At display time, the images are locally reencrypted and sent to the projector where they are decrypted and displayed to the screen.

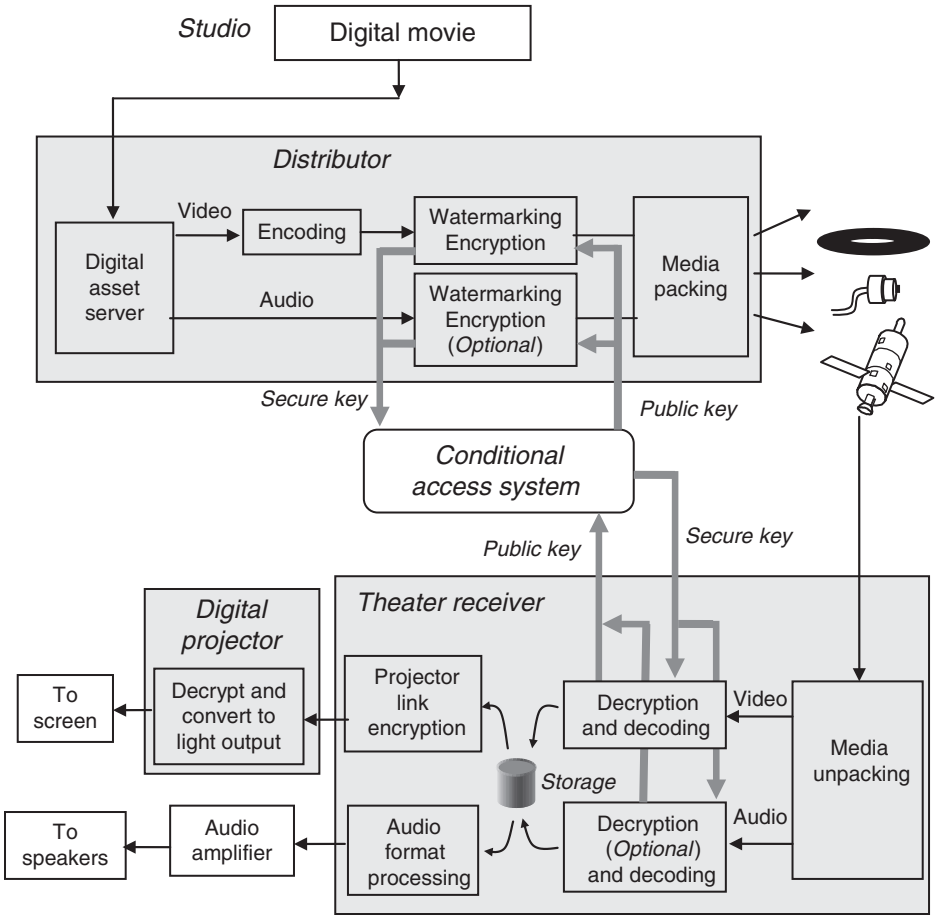


Figure 13-17 Digital cinema distribution. Movies are watermarked, and then encrypted at the distribution site using secure keys. At the receiver, the content is decrypted after being received and locally reencrypted before being sent to the projector.

5 EXERCISES

1. **[R02]** The photograph of an original and rare museum painting is watermarked prior to its digital distribution around the world. Someone illegally copies the image, prints out large copies, frames them, and starts selling them. To help track down the pirate, you scan the image back in and proceed to detect the watermark.
   - Explain some of the steps that you would take.
   - Do you see any potential problems of why and when you would not be able to detect the watermark?
2. **[R02]** Cryptography, which is now widely used, is also used for copyright-protection purposes. What is the main difference employed by cryptography techniques when compared with watermarking techniques?

3. [**R03**] Mention classes of images or image/video data in which watermarking can be hard.
4. [**R05**] In this exercise, reason intuitively and sketch a 3D graph to show how perception might be affected with the presence of a watermark. Clearly the amount of information in the watermark (payload) and the robustness of the watermark have an effect on the perception of the watermarked multimedia data. Plot a graph with payload and robustness on the x- and y-axis showing perceptual impact on the z-axis.
5. [**R03**] One naïve way to encrypt media such as video is by encrypting the entire bit stream (whether image, audio, video) using an operation such as single, double, or triple DES. Regarding this approach, answer the following:
   - Find out what DES does first and explain whether you think this method to encode the media is very secure.
   - What advantages do you see here besides its simplicity? Give a few examples where this method seems useful.
   - What drawbacks do you see with this method? Mention two scenarios where it cannot be used.
6. [**R04**] In this exercise, reason about multiple watermarking.
   - What will happen if you watermark a watermarked image?
   - Could this be regarded as a kind of attack? If so, what is its effect?
   - If person A and person B both insert watermarks, mention what steps can be taken to prove the rightful owner of the content.
7. [**R04**] In Chapter 8 we explained the process of video encoding/decoding. It was mentioned there that to improve video quality of decoded video data, a deblocking filter is employed. Discuss how the deblocking filters affect watermarking.
8. [**R04**] Encryption and watermarking are both used to protect media content.
   - Which one normally precedes the other? Why?
   - Are there any advantages and disadvantages of doing the reverse? Can you explain any practical scenarios where this might prove useful?
9. [**R05**] One potential way to encrypt compressed image and video bit streams that use DCT based techniques during compressed can be explained as follows—The compression step quantizes the DC and AC coefficients of the 8×8 DCT block. However, instead of mapping each 8×8 block to a 1×64 vector in zigzag order, we decide to individually reorder all the 64 coefficients in the 8×8 quantized DCT blocks to 1×64 vectors according to a random permutation list.
   - Can this method be used to encrypt? Is this scheme secure?
   - What problems do you see with the encryption process? Mention at least two major problems.
   - Knowing that you cannot change the encryption module here, how will you make adjustments to the other parts to take care of these problems? Explain.

10. [**R05**] Here is another potential encryption algorithm. A bit stream is produced by encoding a video stream using the MPEG class of algorithms. During the encryption process, the stream is divided into data segments of 128 bytes. Then, two 64-byte lists are generated depending on the binary value of a 128-bit key. If the binary value is equal to 0 (resp. 1), the corresponding byte is put in the first (respectively second) list. The lists are XORed and one of them is encrypted with a second key.
    - Does this scheme provide overall security? Can you prove it?
    - What problems do you see with this scheme? Mention at least two major problems.
11. [**R05**] We discussed the Koch/Zhao algorithm in the text, which tries to embed watermarking bits in the DCT coefficients, which lie in the midband frequency coefficient range.
    - When this algorithm is applied to embed watermarking information for video, it does so in I frames. Can it use B frames? What about P frames? Explain both cases giving reasons.
    - For this algorithm, you normally do not need the original image/video to detect the embedded watermark. Can you think of a case when you will need the original image to detect the watermark? Explain any assumptions that you make.
    - You want to embed watermarks using this algorithm in two videos—a 2D cartoon animation, such as Mickey Mouse, and a feature film such as *Gone with the Wind*. If the watermark to be embedded has the same payload, do you see any reasons why embedding will be harder for one over the other, or would it not matter? Explain.
12. [**R08**] Watermarking an image/video frame once can help determine rightful ownership in cases of piracy. However, watermarking once is normally not effective. Consider an end-to-end system where a video-on-demand server is serving a video from a station to end customers via intermediary distributors. If in the future, an illegal copy is found, you would want to track it down conclusively to the distributor or the end customer who might have perpetrated a crime, and if possible also identify the time when this illegal copying was done. Propose an architecture that will help you in this goal. You may assume that the video stream that is served at the source is an MPEG-2 video stream.

PROGRAMMING ASSIGNMENTS

13. [**R05**] One of the basic algorithms discussed for watermarking uses bit planes to embed information. Take one of the sample images provided; create a pseudorandom 1-bit image (black-and-white noise) of the same size. You can write a program to create this bit image or even use other imaging tools. Adjust sections of the random image visually so as to write your

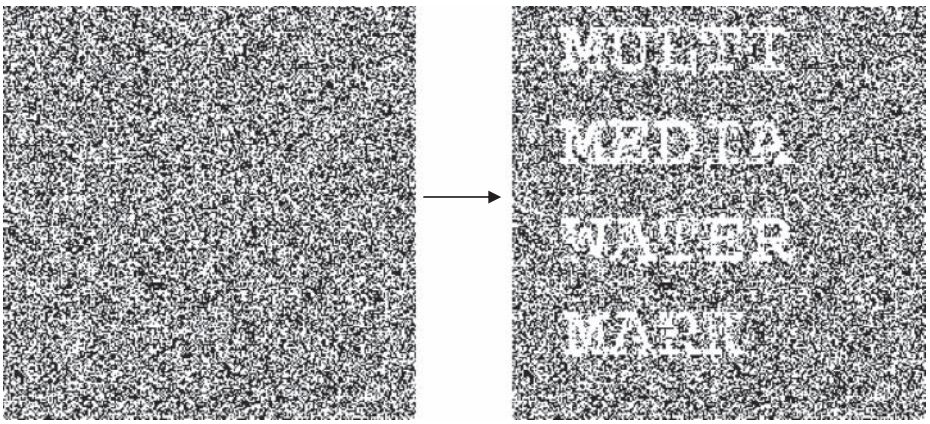


Figure 13-18

name. Figure 13-18 shows an example of such a pseudorandom image with the text "multi media water mark." Now, replace the lower bit plane of the selected image with this new pseudorandom bit image. Try replacing the same image in each successive higher bit plane and see the effect. Comment on your results.

While programming, remember that the energy distribution of the watermark information has to be uniform. So, you will need to add some randomness to your composited name.

14. **[R04]** In this assignment, you will write a simple program to add a watermark in an audio signal at varying strengths and study its perceptual quality with compression. There is code in the Student Downloads section of *www.cengage.com*, which has been provided for you to render PCM-coded audio stream. Here are a list of tasks to perform in this exercise:

    - Read a PCM-coded file and render it.
    - Modify the code to take a string (which is the watermark) and an integer *M* (which specifies the strength of the watermark). Normalize the character string by dividing it by 255. You will get a floating-point value for each character. Multiply this with the strength and round it off.
    - Add this to every *nth* sample of the audio and create a new file.
    - Also implement the reverse process where you read the new PCM file and extract the watermark.
    - For what minimum value of *M* can you fully recover the watermark?
    - Study the distortion in the rendered new file by playing it (and listening) as *M* increases.

15. **[R06]** In the programming assignment of exercise 17 in Chapter 7, you implemented the DCT algorithm and compressed the image by quantizing the

frequency coefficients. In this assignment, you will employ a simple encryption algorithm on the DCT coefficients.

- Read an image, divide it into 8×8 blocks, and convert it to the DCT domain. Quantize the DCT coefficients as suggested in Programming Assignment 17 of Chapter 7.
- Encrypt the DC coefficient by shuffling the bits of the coefficients. You can implement a pseudorandom shuffling method that takes the coefficient and an index (which is the position of the coefficient in the zigzag order of coefficients) and generates another value where the bits are shuffled around. Dequantize all encrypted coefficients and display the image. What kind of image do you see with just the DC coefficient encrypted?
- Do the same for five of the AC coefficients along with the DC and compare the results.
- How many coefficients do you need to encrypt to see visual degradation when the image is decoded, assuming the coefficients are fine? Do you think this is dependent in any way on the content being encrypted or does the content not matter?

This page intentionally left blank

CHAPTER 14

MPEG-4

The advent of the compact disc technology enabled the distribution of digital audio in stereo. The main advantage was the delivery of higher fidelity compared with its then analog counterparts, and it was a harbinger for the success of the digital media format. This generated overall industry interest and investments in the next digital wave that involved a wide-scale distribution of digital video, audio, and images. One of the foremost requirements for the industry was the setting of common standards to allow the different content creators, distributors, equipment manufacturers, and so on to provide consumers with the ability to view content in an interoperable manner. There are two main bodies whose primarily responsibility is the standardization process—the International Telecommunication Union (ITU) and the Motion Pictures Expert Group (MPEG). MPEG was formed in 1988 with the initial mandate to develop video and audio coding techniques to produce good quality at 1.5 Mbps for the video compact disc (VCD) as a target application. This resulted in the establishment of the MPEG-1 standard, with the objective to have an open, interoperable, practical coding standard for video and audio that provided the leading technology at a low implementation cost and the potential to improve performance even after the standard was completed and in use. Around the 1990s, MPEG tackled the need to set standards for another large future market of digital television. Thus began the development of the MPEG-2 standard, with the objective to distribute high-quality, efficiently coded interlaced video. The initial version of the MPEG-2 standard was approved in 1994. MPEG-2 aimed to provide very good audio video quality at about 4–15 Mbps with up to five channels of audio. The lower range was used for digital broadcast television and the higher range was incorporated into the DVD standard. MPEG-2 has also been extended for other applications such as high-definition television (HDTV) at 15–30 Mbps, video over Asynchronous Transfer Mode (ATM) networks, and IP networks

(IPTV). MPEG-2 does provide serious improvements over MPEG-1 in random bit stream access capabilities for fast forward and fast rewind, support for scalable coding in varying bandwidths; however, similar to MPEG-1, MPEG-2 provides only single stream video and audio (though with many audio tracks synchronized to one video) functionality. A more defining difference exists at the systems level. MPEG-1 was designed for digital storage on discs, whereas MPEG-2 is more generally addressing digital distribution. MPEG-2 defines a transport stream that packages the compressed digital video and audio for transport over digital networks.

MPEG-4 is the next modern standard set forth by the Motion Pictures Expert Group and is adapted to the now proven success of digital television, interactive graphic applications, and content access capabilities provided by the Internet. It was originally introduced in 1998 and has gone through various versions until 2003. MPEG-4 addresses content creation, compression, and delivery using all the different media types—audio, video, images, text, and 2D/3D graphics—and, in that sense, is the first standardized multimedia platform. There are different sections in the standard, each of which individually addresses the different media type compression formats and their delivery over a wide range of bandwidths from low bandwidths such as multimedia over cell phones, to higher bandwidths for broadcast. However, the prime difference between the MPEG-4 standard and its predecessors lies in the systems layer. The systems layer describes ways to assemble content using the different media types, to compress and multiplex them for synchronized delivery, and to define intra- and interdependencies streams to create good content-based interactivity.

Consequently, in this chapter, aside from describing the overview of the MPEG-4 standard and the individual parts that relate to the visual, audio, and graphics media streams, we present the systems-level concepts with more rigor. Section 1 gives a general overview of MPEG-4 in comparison with its predecessors and also describes examples that illustrate the paradigms used in MPEG-4. Section 2 describes the systems layer, its representation, and how it works in MPEG-4. Section 3 illustrates the audiovisual objects and how they are represented, synchronized, and compressed in a standardized manner using MPEG-4. Sections 4 and 5 talk about the audio and video objects in MPEG-4, commenting on the compression advancements that MPEG-4 makes for these media types in relation to the predecessor formats. Section 6 describes the Transport layer used to stream MPEG-4 data using a variety of network protocols, and Section 7 illustrates a few commercial applications that have been deployed using aspects of the MPEG-4 standard.

1 GENERAL FEATURES AND SCOPE OF MPEG-4

MPEG-4 is designed as a true multimedia format for applications that involve all of the various digital media types. As mentioned previously, though the original scope of MPEG-4 was very low bit rate audio and video coding, it was later extended to include generic coding of audiovisual objects for multimedia applications. Sample scenarios or applications that the MPEG-4 standard aims at include video-on-demand over Internet and intranets, video audio delivery over wireless networks, home movie distribution, interactive home shopping, interactive television, voice e-mail and video mail, virtual reality games, simulations that involve real video with overlaid virtual objects, low-bit

videoconferencing applications, media object databases, and so on. Typical commercial usage in these applications requires an end-to-end platform setup that involves content creation in a standardized bit stream, distribution, and transport of this stream(s), as well as decoding and rendering with the allowed interactivity support on the end client platform. MPEG-4 aims at creating a universal standard that can be used for all these applications. The standard allows this by providing a means to separate the application's functionality from the application itself. The functionality of the application resides in how the user interacts with the content. For instance, clicking on an image starts a video, or moving a mouse pointer increases/decreases the volume. To separate this functionality from the application and allow an end client to interact with the content requires that the media objects keep their identities individually, be compressed and transported individually, and be composited only at the time of rendering on the end client system. This will become more apparent in the succeeding sections.

MPEG-4 does include advancements that relate to compression and coding of individual media types, such as video and audio over its former standards like MPEG-1 and MPEG-2. For instance, the coded bit stream for video has placeholders for local and global motion compensation, and audio encoding has better quality for the same compression ratios. However, the systems layer of MPEG-4 makes it possible to define content in a way that allows several applications to be created and defined by the content itself. This is one of the foremost differences between MPEG-4 and its earlier MPEG predecessor. Some of the key aspects of MPEG-4 are as follows:

- *Independent coding of media objects*—Rather than combing and compositing all media objects in one object (such as a video) and then compressing/delivering the composited media object, each media object is compressed individually and delivered as an individual stream. The composition of the media objects occurs at the end client terminal.
- *Combining 2D/3D graphics animated objects with video, audio, and images*—Although the prior MPEG standards involved only one video and audio, MPEG-4 additionally involves combining synthetic 2D/3D graphical objects with natural video and images. Furthermore, all these objects may be animated.
- *Compositing audiovisual objects into a scene at the end terminal*—Although this is partially mentioned in the first point previously, it should be clear that maintaining the individuality of objects during compression and transmission allow a user to interact with them individually at the end terminal.
- *Content-based interactivity*—MPEG-4 allows the ability to interact with important scene objects. Scene objects consist of natural media elements, such as video, images, and audio, as well as synthetic elements, such as graphics and animations. MPEG-4 allows intramedia and intermedia interaction.
- *Improved compression*—MPEG-4 was started and ratified at a time when compression technologies for all the media types such as video and audio had undergone vast improvements. Additionally, because of its object-oriented nature, MPEG-4 allows the degree of compression applied to any stream to individually adapt to the channel bandwidth or media storage capacity.
- *Universal adaptability*—As a standard, MPEG-4 makes it possible to access audiovisual data over a diverse range of storage, transmission channels using

different end terminals. For instance, MPEG-4 provides access over IP networks to computers as end terminals, but it also allows the same content to be sent over wireless networks and cable networks, all having different bandwidths and network QoS. This is possible because MPEG-4 provides robust tools in error-prone environments and also tools that allow fine granularity of media scalability.

As a standard, the first version of MPEG-4 was ratified in the middle of 1999, followed by version updates. MPEG-4 is formally referred to as ISO 14496, consists of many parts that includes systems (14496-1), video (14496-2), audio (14496-3), conformance (14496-4), reference software (14496-5), and a Delivery Multimedia Integration Framework (DMIF; 14496-6). The conceptual architecture of MPEG-4 is shown in Figure 14-1. It is formed of three logical layers: the *compression layer*, the *sync layer*, and the *delivery layer*. At the source, where content is created, the compression layer encodes all the media elements

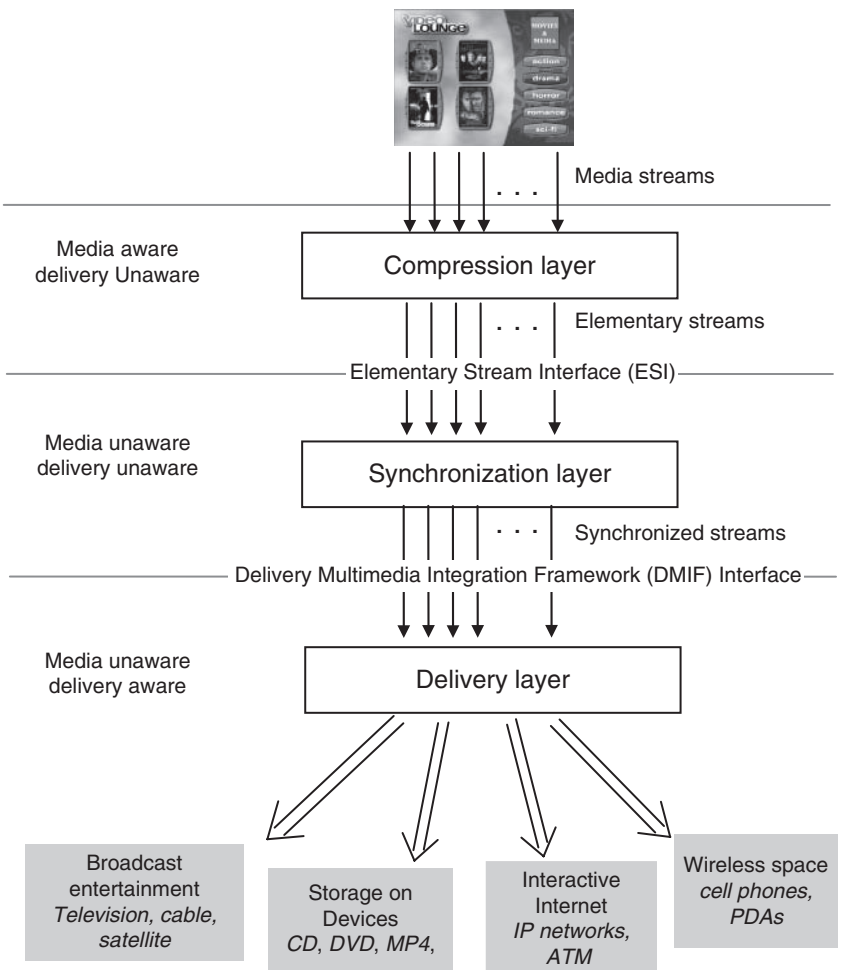


Figure 14-1 Overview of the MPEG-4 standard

and scene descriptions into their respective individual streams called elementary streams. On the client side, the decoder is responsible for decoding the individual elementary streams. The sync layer manages the synchronization between elementary streams and hierarchical relations between them. The delivery layer ensures transparent access to content, irrespective of the technology used to deliver data such as IP networks, wireless networks, and so on. It should also be noted that the compression layer directly deals with media content but has no knowledge of delivery. Hence, it is termed *media aware* but *delivery unaware*. The delivery layer, on the other hand, is completely agnostic with respect to content-related details but has all the delivery-related information to deliver streams from nodes using different protocols. Hence, it is termed *media unaware* but *delivery aware*. The intermediary sync layer, which interrelates synchronization between various elementary streams, knows neither about the specific media types nor the delivery mechanisms to be used for delivery. Correspondingly, this layer is termed as *media unaware* and *delivery unaware*.

1.1 MPEG-4 in Relation to MPEG-2 and MPEG-1

Before a detailed description of the MPEG-4 format, it is important to compare and contrast MPEG-4 with its predecessor ISO and ITU formats. The earlier formats were mostly geared toward compressing media that involved single video streams (H.261, H.263 etc) or single mono/stereo/multichannel audio streams for voice and wideband music (G.711, G.722, MP3, and so on), or even a combination of a single audio and a single video stream together to deliver A/V content (H.324, MPEG-1, MPEG-2). In all instances, the respective standard committees analyze the then recent research and define a bit stream format that can deliver content at a specific bit rate at the required levels of quality, fidelity, and resilience. Also, with the exception of MPEG-2, most standards aim to compress down to a formatted bit stream without defining any specialized means to transport the compressed data over digital networks. MPEG-2 has additionally defined a transport stream that consists of 188-byte packets to transport audio and video streams to end terminals.

The MPEG-4 format focuses on creating standardized bit streams that aim to create and deliver *interactive multimedia applications* rather than a single video stream or audio stream. In comparison with earlier formats that aim to fulfill certain requirements for a specific application having well-defined bandwidth requirements, MPEG-4 aims to fulfill application requirements needing delivery at very low bit rates and also high bandwidths. The media types in MPEG-4 applications involve multiple audio, visual, and graphics streams—each called an audiovisual object stream. Furthermore, MPEG-4 specifies a sophisticated systems-level bit stream organization that allows individual streams to be defined spatially, interrelated temporally, and transported across digital networks. A powerful concept that needs elaboration here is that any media application's functionality can be created by appropriately defining spatial and temporal relations between media objects as well as setting up interactivity that allows media objects to change their state either temporarily or spatially. MPEG-4's systems-level provisions allow such interactivity to be architected to define applications. This means that if the MPEG-4 terminal is made to emulate the standardized systems layer on any platform, then any application can be created and run on the MPEG-4 terminal by appropriately defining the functionality of

the content. This should be made clearer with the examples in the next section where a variety of typical applications are described, each of which is different in its own right but does not change the MPEG-4 terminal or the way it interprets audiovisual objects and the functionality. In other words, the MPEG-4 terminal has to be created and deployed only once on any device—computer, set-top box, cell phone, PDA, game console, and so on—and it will correctly decode and create applications as defined by the MPEG-4 content.

1.2 MPEG-4 Sample Scenarios

This section illustrates sample scenarios where MPEG-4 adds value to wide-scale content distribution, content management, and content interactivity. The first example in Figure 14-2 illustrates a typical A/V player. The typical A/V player shown on the left has its looks and user interface defined as part of the player. The play, pause, and stop functionality is part of the player, which resides at the client and controls the state of a single audio and video stream sent to the player. The middle image shows the same setup on an MPEG-4 terminal. Here, the play, pause, and stop user interface images

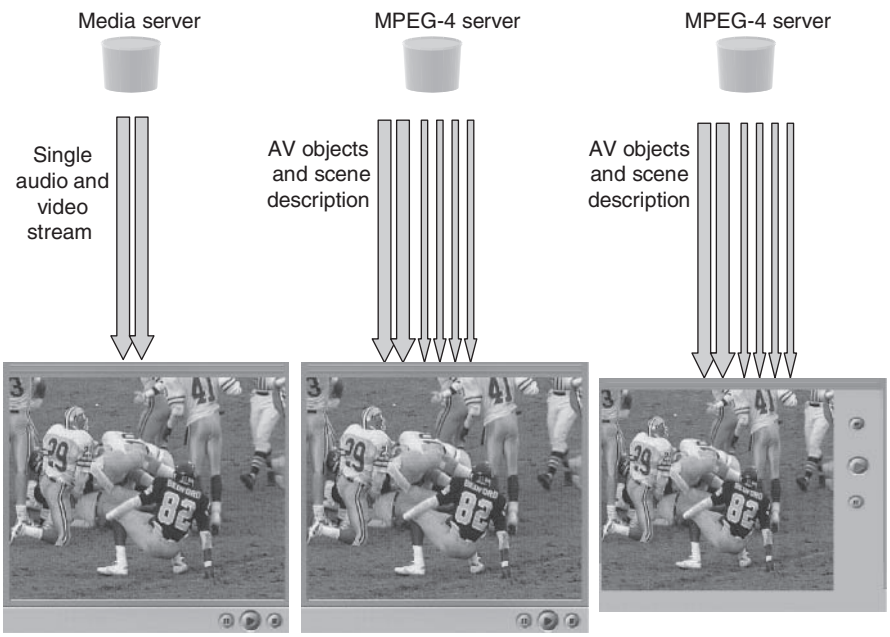


Figure 14-2 Differentiating MPEG-4 content. The left image shows a simple A/V player with all the functionality built in to the player; a single audiovisual stream is sent to the client. The middle image shows the same functionality with MPEG-4 audiovisual objects. The images that correspond to play, pause, and stop are sent to the player as objects with positional and interactive properties. The right image shows how the spatial properties that are conveyed to the MPEG-4 player of the image objects have been changed to create a new look for the content.

are objects described by the content and are communicated to the MPEG-4 player by the content. Each object has spatial properties that relate to the location of the object images on the screen and functional properties that control its own state or the state of other object streams being sent to the terminal. For example, clicking or selecting the play button starts the video object stream. The third image illustrates how the user interface can now be repositioned and even animated by the content because it is part of the MPEG-4 content and not inherent in the player. In other words, the MPEG-4 content is authored to define the spatial locations of the buttons in this case and the images that correspond to the buttons. This allows you to have a look-and-feel spatially as well as functionally defined within the content and not by the player.

Another application to explain the value added by MPEG-4 is shown in Figure 14-3. This example illustrates interactive television programming delivered over high-bandwidth IP networks. Most sports games are recorded using multiple cameras with all the live video feeds sent to a centralized production station. A director picks one of the streams, which is sent to all the viewers over satellite/cable networks and this form of delivery results in a passive viewing experience. Instead, with enough bandwidth capabilities, all streams can be delivered in a synchronized

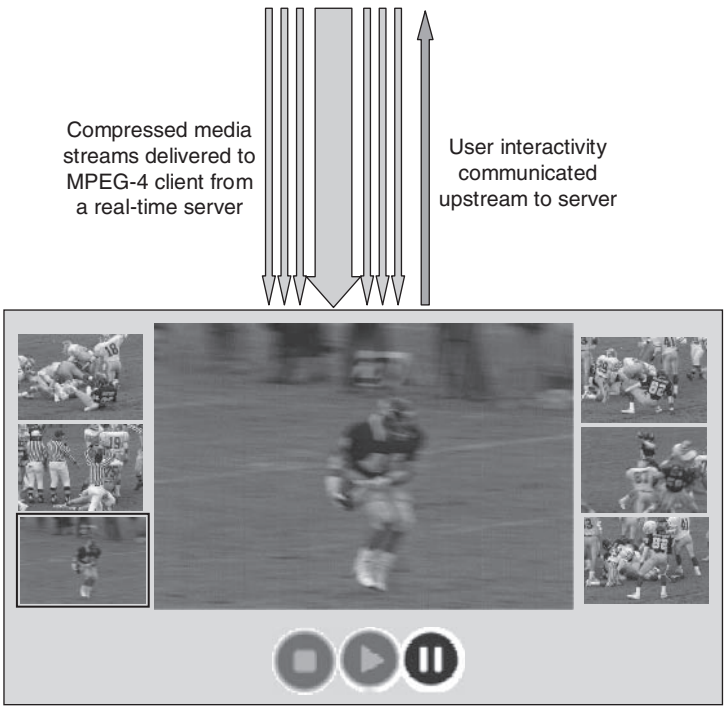


Figure 14-3 MPEG-4 interactive television application to view sports. A game is recorded using six cameras. All the six videos are delivered to the user at low resolution. Only the user-chosen stream is delivered at higher resolution, which can be changed at any time. This enables the game to be seen by the viewer from any angle at any time, depending on the user's choice.

way. Shown in the figure are seven streams delivered, six downsampled video streams, and one main high-definition stream. The user can select any one of the six streams at any time. This action is communicated back to the server, which switches the high-resolution stream sent to the client. The user is, therefore, able to see and view the game from different viewpoints simultaneously and make a camera choice. The same paradigm can be extended to different markets such as security or any event viewing.

1.3 Representational Features

The content created in the MPEG-4 format consists of different media types that form a scene. The media types in a scene have spatial (or positional) and temporal attributes, which are defined in the systems layer and can change or be animated over time. Each media type, whether video, audio, image, or graphics, is represented individually as an object called an audiovisual object (AVO). For instance, in the interactive television application discussed in Figure 14-3, the scene consists of several videos, audio, and images—the six low-definition video streams; the central high-definition video stream; the audio stream that corresponds to the central video; and the play, pause, and stop images. Each AVO has a spatial location with interactive properties that have been set up. For instance, selecting a video changes the central high-definition stream. Play, pause, and stop cause appropriate functionality changes to the content.

Additionally, the AVOs in a scene are organized in a hierarchical fashion. At the bottom of the hierarchy is the primitive AVO, which can consist of an image of a person, or the video of a person talking, or the voice of that person. A number of primitive or elementary AVOs in MPEG-4 are capable of representing natural and synthetic content types in both 3D and 2D. The 2D and 3D graphical objects and text along with fonts are specified as a description in the scene. Each AVO is represented separately and encoded as an independent stream. The AVOs are given spatial properties, which make them reside in 2D or 3D. Audio objects are placed in a sound space. Each of the individual AVOs is packed around a descriptor that qualifies the AVO streams and is streamed to the client as and when needed. The representation and composition of AVOs is better explained in Section 3.

1.4 Compositional Features of MPEG-4

MPEG-4 provides standardized ways to compose audiovisual objects to form audiovisual scenes on an MPEG-4 terminal. Each AVO is streamed separately to the client MPEG-4 terminal. The systems layer defines the content to have proper time and space synchronization. For instance, in the interactive television application described in Figure 14-3, each of the videos has to be rendered at its respective positions and frame rates. Additionally, user interactivity can change the scene, like videos being replaced, images changing their positions on screen and so on. When dealing with 3D objects in 3D space, the user could change viewpoints or have objects animate. For instance, although not described in the previous interactive television example, the AVO could be animating on transitions when a certain video is clicked.

The video chosen in the corner could increase in scale and change its position to cover the central area in a limited time, all while still rendering synchronized frames.

1.5 Multiplexing and Synchronization Features of MPEG-4

With the large number of AVO objects supported in MPEG-4, synchronization is critical for the content to be viewed in the manner it was authored. Though each AVO is encoded separately, each has intramedia synchronization as well as intermedia synchronization. Intramedia synchronization issues require that the stream be rendered at the required rate in the MPEG-4 client terminal. For instance, each video or audio has a frame rate, each image can be displayed for a certain instance of time, and animations of graphics can start/stop during a presentation. Furthermore, these temporal properties can be different for different AVO streams. For instance, two videos or a video and the corresponding audio can have different frame rates. To maintain intramedia synchronization for a stream, the entire pipeline, which consists of the transport/delivery of the compressed AVO stream, the decoder, and the rendering engine at the client all need to cooperate so that, as the data arrives, it is decoded and displayed at the necessary rate. Also, if the necessary rate cannot be maintained due to delivery traffic, bandwidth jitter and so on, data might need to be dropped. Intermedia issues relate to interactive actions by a user that change the properties of other AVO streams. For instance, in the interactive television application described previously, clicking on a small video stream necessitates stopping transport, decoding and rendering the current large high-definition video stream, and restarting the same process with another high-definition stream along with the changes in audio streams if necessary. Similarly, MPEG-4 content can be used by a user to start animations, by either clicking or rolling over other AVO objects.

To enable and maintain all these synchronization properties within and across AVO streams, the streams have to be structured for easy delivery. Each audio visual object's data is stored as an elementary stream (ES). The elementary stream is further broken down in access units (AUs). For example, in the case of an encoded video stream, AUs may consist of encoded information that pertains to an I frame, P frame, or B frame; in the case of audio streams, AUs may consist of encoded audio frames, and so on. Additionally, every elementary stream has a qualifying descriptor called an object descriptor (OD). The OD is structurally similar to a URL, and can either embed the entire elementary stream or contain a pointer to it. The OD of an ES is used to inform the MPEG-4 terminal about all the properties that relate to the elementary stream and contains (among other information) the following:

- QoS an elementary stream could request for transmission, including maximum bit rate, error rate, and so on
- Stream type information to help the MPEG-4 terminal determine which decoder to instantiate to decode the stream
- Precision information for deciphering the encoded timing information

The constructs and issues used in synchronization and multiplexing of AVO streams are explained in Section 3.3 with illustrative examples.

2 SYSTEMS LAYER

The MPEG-4 systems requirements can be grouped into two main categories:

- *Traditional MPEG systems requirements*—These relate to delivering standard video, audio, and user-defined private data to the end clients.
- *Specific MPEG-4 systems requirements*—This deals with delivery of audiovisual objects and scene descriptions. They are central to MPEG-4 and not there in the traditional systems specifications for the earlier MPEG versions.

The traditional or general requirements of MPEG-4 are similar to the specifications of MPEG-1 and MPEG-2. Audio and video information has to be delivered in a streaming manner that is suitable for static and live broadcast. This implies that the data is delivered in small/large segments to match the delivery of content to clients with limited terminal and network capabilities. Additionally, the different video and audio components of any multimedia presentation are closely related in time, for instance audio samples with their associated video frames have to be delivered, decoded, and rendered at precise instants in time. The MPEG representation here needs to allow a precise definition of the time for all the segments so that the receiving clients can decode and present the information to the user in a synchronized manner. Finally, the delivery of the overall audiovisual presentation needs all the individual streams to be managed during and across sessions. This includes mechanisms for content locations, stream identification, description of interdependencies, and access to intellectual property information associated with the data.

All the above-mentioned requirements still prevail for MPEG-4. Additionally, the audiovisual object representation paradigm makes it imperative to transmit information that describes the spatial relationships between all the objects, their temporal interdependencies, if any, and the overall management of audio visual objects during a presentation. Much of the functionality that MPEG-4 provides over MPEG-1 or MPEG-2 comes from the systems part of the standard. With every media stream encoded separately, the systems layer functions as a wrapper, providing a way to efficiently manage streams and to visually/functionally describe the streams. In particular, MPEG-4 systems provide the following:

- A coded hierarchical representation that describes an interactive audiovisual scene description. This is known as Binary Format for Scenes (BIFS), which also includes spatial and temporal descriptions of objects and describes functionally their interrelated behaviors.
- A stream management framework with time and buffer management of every coded elementary stream. This includes stream identification, decoder configuration requirements, object content information, and synchronization information all included within an object description representation (object descriptor).
- An Intellectual Property Management and Protection (IPMP) interface that manages secure delivery of media object streams when necessary.
- A file format representation (MP4) that allows a flexible and efficient storage characterization of the numerous elementary streams with their object

descriptors. This representation allows extraction, rewriting, interchange, and editing of descriptors and individual units (access units) of each elementary stream.

- A logical presentation engine at the client that allows programmatic and synchronized rendering of the presentation.

The systems layer details information related to the presentation of the scene, provides a robust functional methodology to describe content that is authored by a user, provides appropriate synchronization between AVOs, and also provides metaphors to consume content at the client end terminal. It is best to understand how MPEG-4 systems functions by understanding how a presentation scene is modeled using AVOs, encoded, and transported. This is explained in the next section using illustrations shown in Figures 14-4, 14-5, and 14-6.

3 AUDIOVISUAL OBJECTS

Any MPEG-4 presentation consists of audiovisual objects having spatial or positional properties and temporal properties. This section describes how the systems layer describes a presentation consisting of AVOs using a scene description language. Figure 14-4 shows an example of a video-on-demand application implemented in MPEG-4. The content is authored so as to function as a video-on-demand interface that allows users to select a video category. The selection of a category changes the scene being viewed. The user can choose a specific video at any time, starting the transport of media object streams. There might be additional interactivity where the user is authenticated, information regarding account charges need to be accessed, and so on. The level of scene interactivity where a user can interact with the presentation and decide which of the stream(s) need to be sent, stopped, or paused requires the content to be described in a way that specifies the spatial object layout and temporal relationship. The scene representation is an integral part of MPEG-4 systems allowing the standard to represent simple and complex interactive content. A scene description can be represented textually in an XML format, extended VRML format, or even a proprietary description language. However, when encoded into MPEG-4, it is translated into a standard, coded representation called Binary Format for Scenes (BIFS).

The scene layout also refers to the audiovisual objects, which get encoded separately in their own elementary streams. The illustration shown in Figure 4-14 visually shows such an encoding where the scene representation is encoded separately as a BIFS stream, whereas all the other referred AVOs get encoded in their own individual elementary streams. Next we describe how scenes are represented and encoded.

3.1 Representation of Scenes and Interactivity Setup Using AVOs

The scene is described hierarchically as a tree of nodes. Each node can describe positional information, positional hierarchies of a group of nodes, and so on. Nodes used

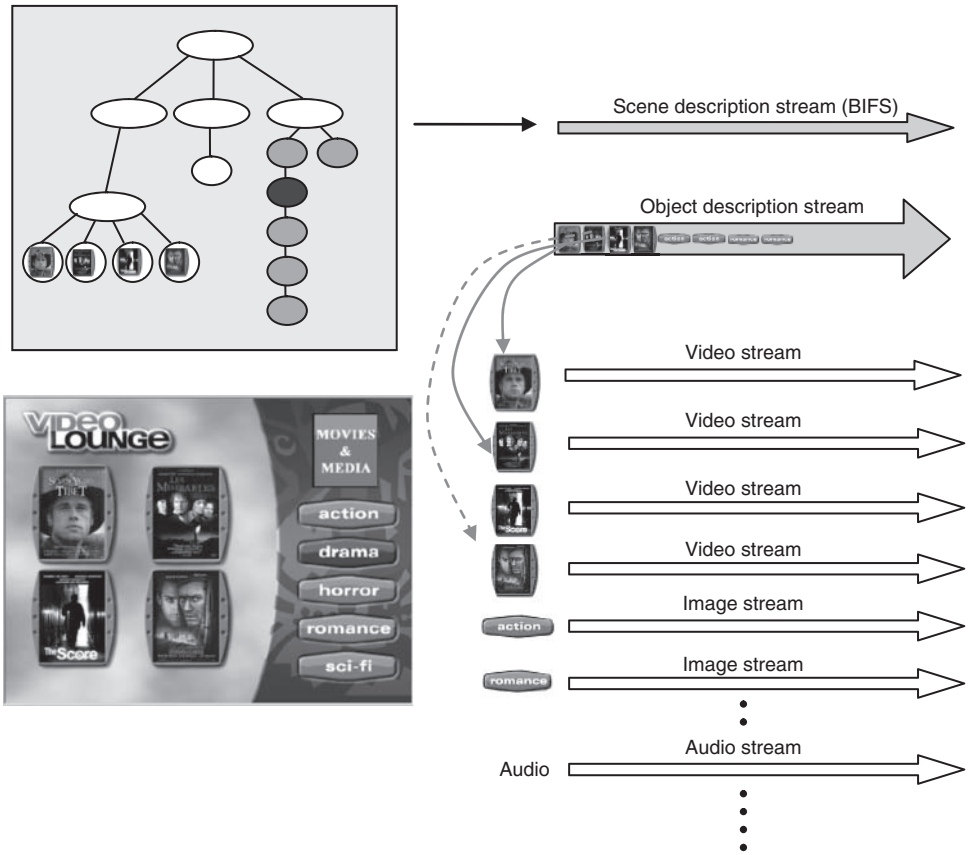


Figure 14-4 Systems-level description. Any presentation is described by a scene description, which consists of hierarchies of spatial locations and temporal relationships between media objects. The scene description is encoded in a Binary Format for Scenes (BIFS), which has embedded pointers to media objects. These pointers describe object descriptors, which are all encapsulated in an object descriptor stream. Each object descriptor, in turn, points to the individual compressed media elementary streams of video, audio, and images. 2D and 3D graphics are part of the BIFS stream.

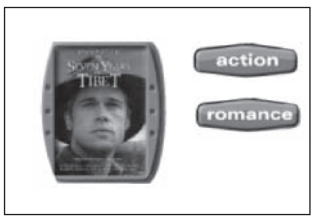
further down in the tree hierarchy describe whether the content represented by the node is audio or visual. Visual content is typically described by shape nodes, where the shape might be a rectangle or any 2D or 3D graphics. Furthermore, each visual object can have a texture, which is represented by another visual stream (for instance, a video or image). The reference to a video, image, or audio stream is described by a pointer called an object descriptor ID. Thus, a scene is composed of positional nodes, visual nodes, and audio nodes that ultimately point to an object descriptor. An example of a textual representation is described in Figure 14-5. This is a short example that shows

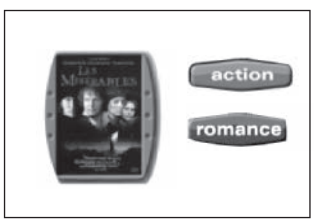
```
/* Part 1– scene layout */
OrderedGroup {
        Children [
        Sound2D {
            ODID 10
            StartTime -1
            StopTime 0
        }
    ]
    Transform2D {
        Translation 500 100
        DEF Button1 TouchSensor { }
        Children [
            Shape {
            Geometry Rectangle { size 100 80 }
            Appearance {
                DEF VideoCat1 ImageTexture {
                    ODID 11
                }
            }
        ]
    }
    Transform2D {
        Translation 500 200
        Children [
            DEF Button2 TouchSensor { }
            Shape {
            Geometry Rectangle { size 100 80 }
            Appearance  {
                DEF VideoCat2 ImageTexture  {
                    ODID 13
                }
            }
        ]
    }
    Transform2D {
        Translation 50 200
        Children [
            Shape {
            Geometry Rectangle { size 200 120 }
            Appearance {
                DEF MovieImage ImageTexture {
                    ODID 20
                }
            }
        ]
    }

/* Part 2 – conditional nodes */
    DEF Function1 Conditional
        {   Replace VideoCat1.ODID by 12
            Replace VideoCat2.ODID by 13
            Replace MovieImage.ODID by 30
        }

    DEF Function2 Conditional
        {   Replace VideoCat1.ODID by 11
            Replace VideoCat2.ODID by 14
            Replace MovieImage.ODID by 40
        }

/* Part 3 – routes */
    ROUTE Button1.isPressed TO Function1.activate
    ROUTE Button2.isPressed TO Function2.activate
}
```





ODID 10 Compressed audio data

ODID 11 

ODID 12 

ODID 13 

ODID 14 

ODID 20 

ODID 30 

ODID 40 

Figure 14-5 MPEG-4 systems scene description. The left image shows a node-based code snippet describing the layout and interactive functionality. Some nodes prefixed by DEF have defined node names, which are referenced later. The top part of the code corresponds to a layout description, as shown in the upper-right corner. On clicking the "action" button or "romance" button, appropriate object descriptor IDs are replaced to change the viewed content.

a part of the layout and functionality of the video-on-demand example. The code snippet in the left column consists of three logical parts:

- The top part consists of a scene layout in terms of audio and visual nodes. The visual nodes are specified with positional information and a textured appearance. Each visual or audio node ultimately contains an object descriptor ID pointer. This pointer refers to the appropriate object descriptor, which encapsulates the associated compressed AVO—an image, audio, or video. The code shows one sound node meant to signify background music and three visual objects, all of which point to image object descriptors. The scene layout as it stands is shown in the upper-right corner.
- The middle part consists of conditional nodes, which, when activated, change and replace the object descriptor IDs on predefined node names.
- The last part consists of routes. Routes are associated with touch sensor nodes, which activate conditionals. Touch sensors are defined in the scene layout and make a defined area sensitive to an input device, the implementation of which is platform dependent. For instance, if an MPEG-4 player is implemented on a PC, sensitive area is responsive to a device that might be a mouse pointer; if implemented on a set-top box, the device might be a remote control pointer; and so on.

Note the use of defined nodes in the scene description prefixed by a DEF. When this is used, the node can be referenced by its name later on. In the example shown, the scene layout names two TouchSensor nodes (*Button*1 and *Button*2) and three ImageTexture nodes (*VideoCat*1, *VideoCat*2, *MovieImage*). The named conditionals (*Function*1, *Function*2) manipulate the ImageTexture's ODID pointers. The conditionals in turn are activated by the routes on TouchSensors. In this example, on clicking the TouchSensor *Button*1, the conditional *Function*1 is activated, causing *VideoCat*1*'s* ODID to be replaced by 12, *VideoCat*2*'s* ODID to be replaced by 13, and the *MovieTexture's* ODID to be replaced by 30, changing the appearance of the scene to the image shown in the middle image on the right. Similarly, on clicking TouchSensor *Button*2, the scene changes as shown on the lower right.

Although this example does show a simple scene manipulation that changes on sensor events, you should easily see how this paradigm can be extended to make complex presentations. For instance, the button press event, besides changing the MovieTexture's URL, could also change its position, size, and even animate its position as it changes the texture. Correspondingly, you could also have routes trigger conditionals when an input device, such as a mouse, is over a TouchSensor node and is clicked, released, or the mouse pointer moves over the area corresponding to the node.

3.2 Encoding of AVOs

The media object streams or audio visual object streams consist of compressed image, video, and audio streams. These are called elementary streams and can be compressed using any of the MPEG-4 defined (or other MPEG defined) codecs. For instance, visual streams can be JPEG compressed, MPEG-2 compressed, or MPEG-4 compressed. Audio

streams can be compressed using speech compression (CELP, HVXC), high-band audio compressed (MPEG-4 AAC, MP3), or synthetic audio streams (using SASL, SAOL). When the stream needs to be decoded at the client, the client needs to know what decoder to instantiate to parse and decode the stream. Additionally, during transport, the Transport layer needs to know the maximum bit rate a stream requires to facilitate Quality of Service setups between sender and client. The client terminal needs to know if the stream is dependent on other streams either for synchronization information (for instance, video is dependent on audio as a driver) or for encryption-related information. Thus, additional qualifying information is needed by the decoder and by the rendering engine to instantiate appropriate decoders, to synchronize streams, and so on. This is encapsulated in a stream descriptor, as illustrated in Figure 14-6.

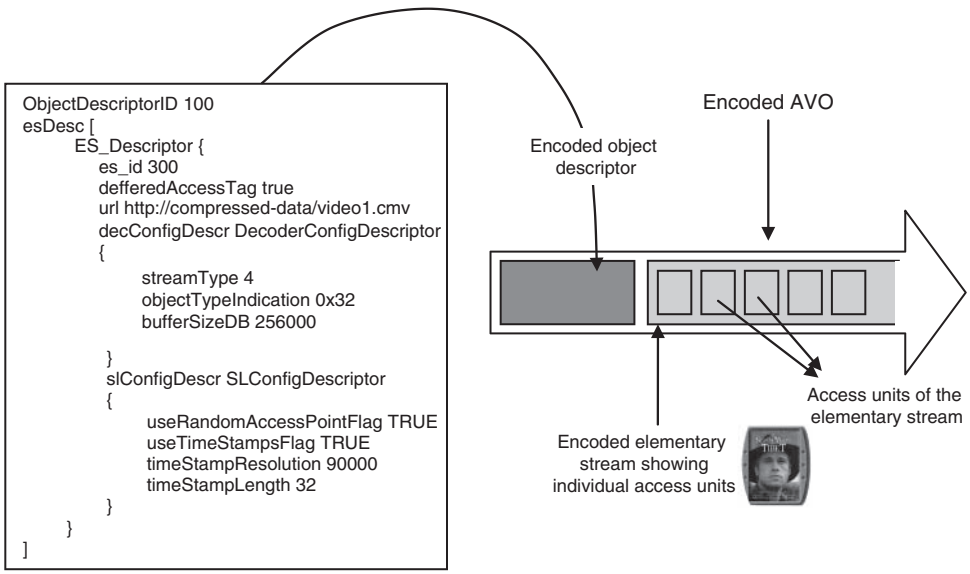


Figure 14-6 Structure of an encoded audiovisual object (AVO). Each AVO has an encoded object descriptor that qualifies the encapsulated elementary stream. The object descriptor contains a list of other descriptors that are used to instantiate and configure the decoder at the client, maintain Transport-level capabilities for the stream, and so on. Shown are two such descriptors—the DecoderConfigDescriptor and the SLConfigDescriptor.

Thus, each AVO that contains encoded audio and visual data also has an object descriptor that qualifies the type and identity of the data in the compressed elementary stream. In the example shown in Figure 14-6, the object descriptor is shown to consist of two other descriptors. The *DecoderConfigDescriptor* contains information used by the client MPEG-4 terminal to instantiate the appropriate decoder. Here, the streamType has a value of 4, which shows that it is a visual stream (as opposed to a BIFS stream, an audio-encoded stream, and so on). In addition, it has an objectTypeIndication of 0x32, which shows that this is an MPEG-4 compressed natural video stream (as opposed to an MPEG-2 compressed stream or an MPEG-1 compressed stream or a compressed image.). The *SLConfigDescriptor* provides information used by the client to synchronize data during decoding and presentation.

3.3 Synchronization and Delivery of AVO Streams

Every AVO stream consists of an object descriptor that encapsulates an elementary stream. The elementary stream consists of access units (AUs) that contain encoded information of the AVO object. For instance, in the case of a video elementary stream, the AUs contain encoded information that relates to an I frame (or a P frame or a B frame).

The scene description is encoded into its own elementary stream called a BIFS stream, and also has its own object descriptor. The object descriptors for elementary streams function as URLs containing pointers to elementary streams. A separate *synchronization layer* in MPEG-4 manages the issues that relate to the delivery and synchronization of elementary streams. It functions similarly at the encoder and in the MPEG-4 client. At the encoder, each elementary stream's AUs need timing information for delivery and display at the client. All the elementary streams need to be multiplexed and delivered, as illustrated in Figure 14-7. At the client, the decoders of the AVO objects request the synchronization layer to get the appropriate access units to be decoded and displayed. Figure 14-7 illustrates the synchronization and multiplexing of AVO streams. The major tasks of the synchronization layer are as follows:

- Identifying the AVO object's elementary stream based on its object descriptors
- Inserting appropriate timing information at the encoder
- Multiplexing and delivering the AUs of various AVO streams in the presentation
- Recovering the media object's (and all the AUs) time resolution and clock reference at the MPEG-4 client

It is best to understand the synchronization layer's internal operation by explaining the workflow at the MPEG-4 source and the MPEG-4 client. At the source, the MPEG-4 setup consists of various encoders that can encode all the media streams in a scene in real time for real-time presentations, or refer directly or indirectly to already compressed media elementary streams. Direct reference includes allocating data into the streams, whereas indirect reference involves pointers to URLs for the elementary streams' access units. The scene description is encoded in its own special stream—BIFS. All the streams have object descriptors. There is also a special object descriptor stream (OD stream) that refers to the ODs of all other elementary streams (see Figure 14-4). Each AU in all streams contains timing information. The timing information consists of two numbers used for synchronization at the client.

- *Decode Time Stamp (DTS)*—The DTS provides the decoder of the audio visual stream the time to start the decoding relative to the beginning of the presentation. If the AU is not available to the decoder before then—due to transport-level delays—it is probably not going to be decoded in time for a synchronized display and the decoder mechanism might exercise a choice to drop the AU.
- *Composition Time Stamp (CTS)*—The CTS provides the scene's presentation engine the precise moment at which the decoded AU has to be rendered on the client screen. Although the AU might be available to the decoder for decoding

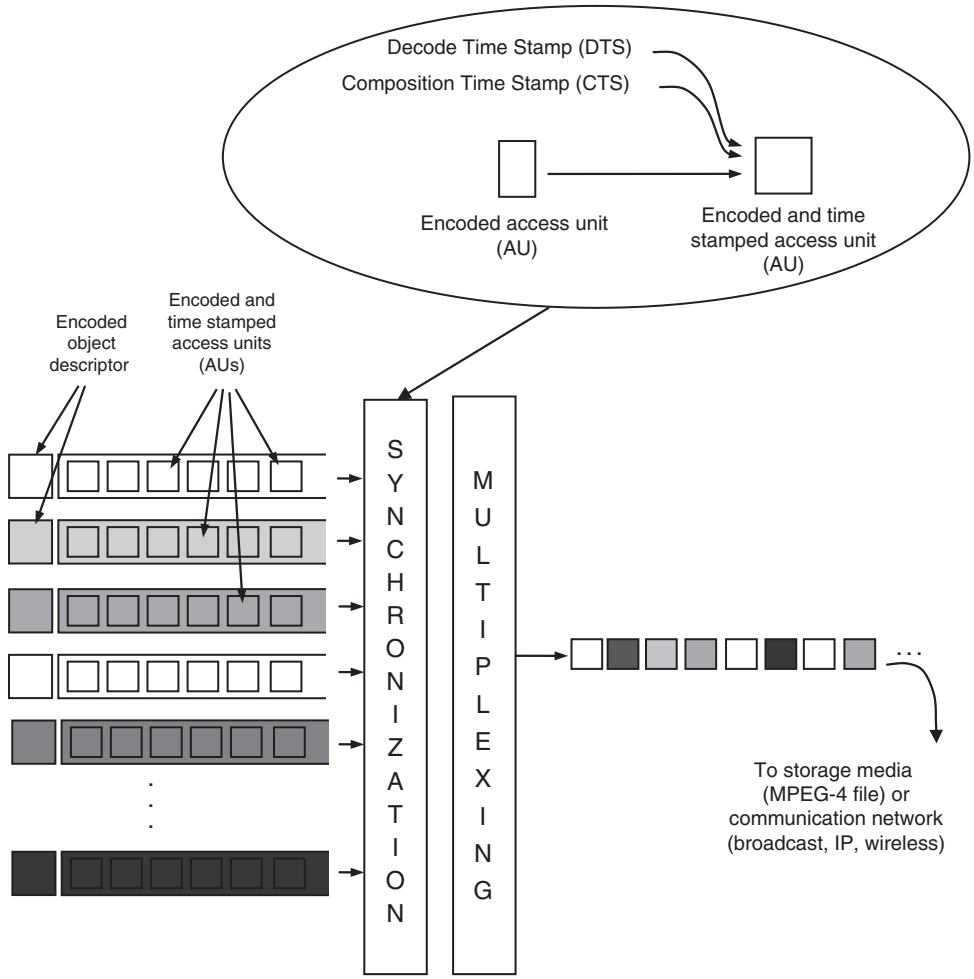


Figure 14-7 Synchronization and multiplexing of AVO streams. The access units (AUs) that make up an AVO stream are first time stamped with the Decode Time Stamp (DTS) and Composition Time Stamp (CTS). All the AUs of all streams are multiplexed for delivery. These multiplexed AUs can be generated in time for real-time streaming or stored in a multiplexed MPEG-4 file format.

at the right time as suggested by the DTS, its decoding might take longer than necessary because of processor resource problems or slow processing at the decoder. In such cases, to maintain synchronization, the Presentation layer might exercise a choice to drop displaying the decoded AU.

At the client MPEG-4 terminal, one of the tasks executed is decoding the BIFS stream to construct a scene layout. During the display of the scene at each time cycle, the client requests the decoders to deliver the specific data for display. This process uses the DTS at the decoder and the CTS at the display or composition time.

4 AUDIO OBJECTS

MPEG-4 audio compression was designed to succeed the earlier MPEG-1 and MPEG-2 formats by building on the functionalities and capabilities that they provide. Conceptually, unlike its MPEG predecessors or the ITU family of audio codecs, MPEG-4 does not define a single or set of efficient audio compression schemes, but a collective and complete toolbox of techniques used to compress low-band speech, high-band multichannel audio, and even music/speech synthesis. Structurally, MPEG-4 does not have a linear streaming architecture that was common to MPEG-1 and MPEG-2 audio, but every MPEG-4 presentation has an object-oriented structure, which can consist of none, one, or many audio objects that can be also be controlled by other scene objects. The interactive functionalities such as object-controlled audio streaming, 3D spatial sound control, and so on reside in the BIFS track. Because the MPEG-4 audio standard supports a variety of different applications, the audio compression support is divided into two groups: natural audio coding, which supports general wideband (music) and narrowband (speech), and synthetic audio coding, which includes structured audio descriptions, rendering speech directly from speech, and interactive audio programming. An overview of these two broad natural and synthetic areas is presented in the following sections.

4.1 Natural Sound

Natural audio objects in MPEG-4 include the classical MPEG audio compression for narrowband speech such as CELP as well as wideband compression such as MP3 and AAC, as illustrated in Figure 14-8. Because no single-coding algorithm can support all

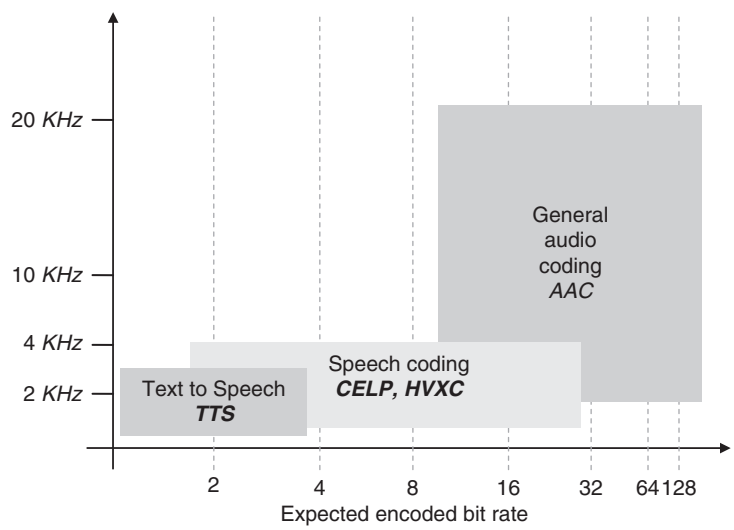


Figure 14-8 MPEG-4 natural audio coding requirements. MPEG-4 has been designed to support a variety of low- and high-bandwidth applications.

these narrowband requirements for speech signals along with wideband audio multichannel audio compression requirements, MPEG-4 supports many different types of compression algorithms, each of which has its own syntax encapsulated within the appropriate object indicators or object descriptors.

- The natural sound encoding algorithms provided by MPEG-4 include the following: *HVXC*—Low rate clean speech coder
- *CELP*—Telephone speech/wideband speech coder
- *GA*—General audio coding for medium and high qualities
- *Twin VQ*—Additional coding tools to increase the coding efficiency at very low bit rates

High-bandwidth support in MPEG-4 covers the bit range from 16 kbits/s per channel to more that 64 Mbits/s per channel. These are implemented by the Advanced Audio Codec (AAC) at higher bit rates and by Twin Vector Quantization (Twin VQ) at lower bit rates. Four object types supported at higher bit rates are derived from AAC: AAC Main profile, AAC LC, AAC SSR, and AAC LTP. MPEG-4 audio is defined to be backward compatible to MPEG-2 AAC and, hence, it supports all tools defined in MPEG-2. Twin VQ increases coding efficiency for coding musical signals at lower bit rates.

Natural speech coding in MPEG-4 is provided as a generic coding framework for a variety of speech signals. Unlike the other earlier standards where speech coding was designed to be for a specific bit rate using a specific algorithm (e.g., G.723 and G.729), the suite of tools and allowable compressed streams in MPEG-4 provide a variety of capabilities, from which an application can choose a certain capability for its purpose. The bit rate coverage spans from 2 Kbps to 24 Kbps. Two main algorithms are covered—HVXC (Harmonic Vector Excitation Coding), used at low bit rates of 2–4 Kbps, and CELP (Code Excited Linear Prediction) coding.

The new functionalities introduced in MPEG-4 are multi bit rate coding, bit rate scalable coding, and bandwidth scalable coding. Multi bit rate coding provides a variety of bit rate selection with the same coding algorithm. Here, a bit rate is selected from a variety of predefined or presupported bit rates for communication along variable bandwidths depending on the application. For instance, HVXC is supported at 2 or 4 Kbps. CELP is supported from 3.85 Kbps with increments of 0.2 Kbps. Bit rate and bandwidth scalabilities are used in multicast transmission, where different end clients can communicate at different or varying bandwidths. For instance, clients might be talking over an IP network to each other, or an IP network followed by a low bit rate cell phone network. The encoder in such cases generates a single base layer bit stream of lower quality with higher scalable bit streams that when decoded in connection to the base layer provides additional quality, as illustrated in Figure 14-9. This way, different voice clients can be sent a base layer stream and optional higher quality streams as needed, which results in a more scalable bit stream architecture compared with creating a different bit stream for each new client.

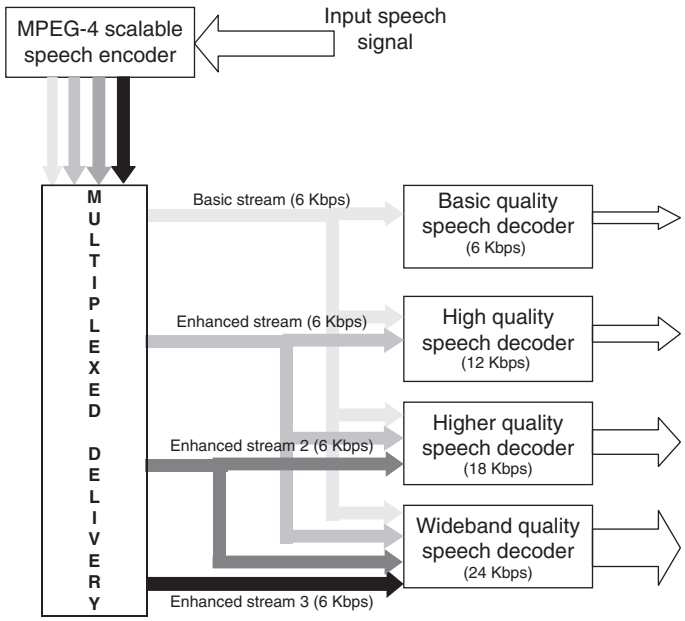


Figure 14-9 Scalable voice encoding in MPEG-4. Shown are one basic layer and three enhancement layers, each encoded at 6 Kbps. With available bandwidth, enhancement layers can be transmitted, increasing the quality of the decoded speech.

4.2 Synthetic Sound

In addition to well-defined scalable encoding for natural speech and sound, MPEG-4 also contains extensive support for synthetic coding and hybrid (synthetic and natural) audio coding. These include the following:

- Structured audio, which allows efficient representation, transmission, and client-side synthesis of music and sound effects
- A text-to-speech interface, which standardizes a single format for communication with speech synthesizers
- Audio BIFS, a portion of the binary format for scene description that makes it possible to describe hybrid sound tracks, 3D audio, and interactive audio programming setups

As with other parts of the standard, MPEG-4 provides tools for advanced audio functionality rather than specific algorithms for compressing synthetic audio. The structured audio compressed bit stream includes two parts. The first consists of a semantic description of event lists of sounds and the duration for which they are played. This is known as the Structured Audio Score Language (SASL). The second part consists of a sequence of synthesis algorithms that specify how to synthesize sound and their transitions. This synthesis model is not fixed and is provided as a framework, which

an end client can interpret and use its proprietary methods for digital sound synthesis. This part is known as the Structured Audio Orchestra Language (SAOL). The structured audio bit stream includes a decoder configuration header, which specifies the semantic SAOL instructions (exact sound implementations in software and/or hardware are left to the end client) used by the client to configure its synthesis engine. After the header is received and processed, the succeeding action units contain SASL time-sequenced instructions, causing the synthesis engine to schedule timely sound renderings, as illustrated in Figure 14-10.

The Text-to-Speech Interface (TTSI) is defined to transmit speech as text in the bit stream along with necessary timing information, enabling the text to be synthesized into speech at the decoder. Again, although the bit stream to communicate data is standardized, the algorithmic implementation to extract text from speech at the encoder and to synthesize speech at the decoder is left to proprietary technology. The audio BIFS are composed of BIFs nodes and BIFs system-level instructions. These together define a connection between the audio elementary streams and the scene audio sources, their 3D positions in space and time, channel-mixing specifications between different sources, and capabilities to resample, buffer, and synchronize the data provided in the audio elementary streams.

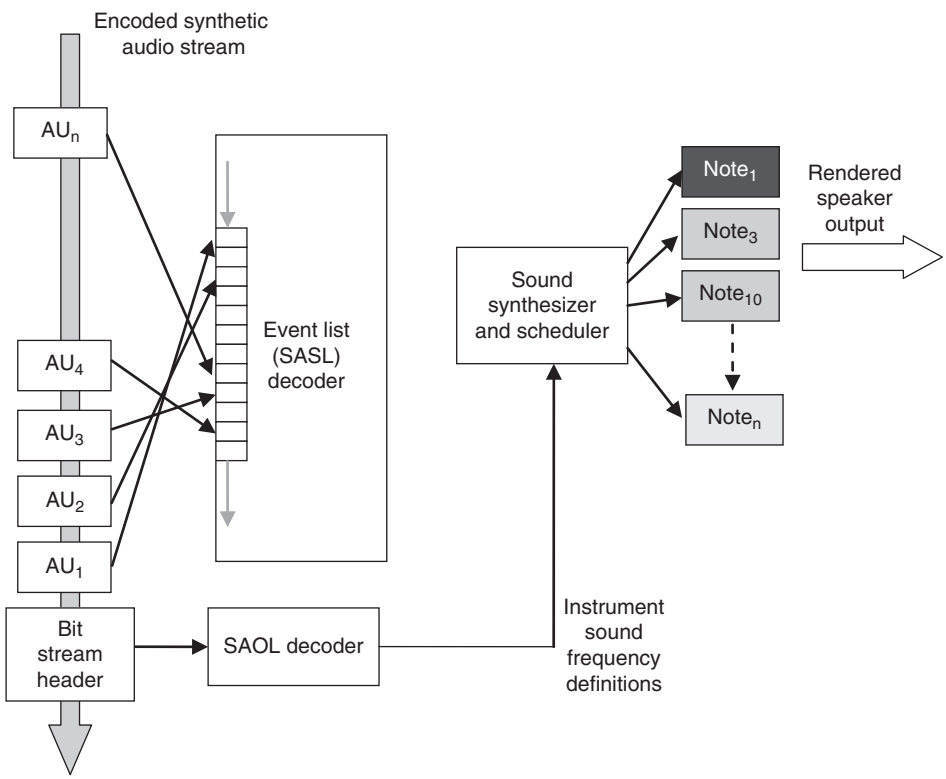


Figure 14-10 Structured audio decoding process in MPEG-4

5 VISUAL OBJECTS

When digital video replaced analog video, most applications provided only the same or similar functionality to analog video. Here, digital video content was viewed similar to analog video, albeit with better resolution and visual fidelity. Examples of such applications include digital television and replacement of analog VHS cassettes by VCDs and later by DVDs. MPEG-1, and especially MPEG-2, have been key standards in the deployment of these applications. However, once this content is in the digital domain, new viewing and interacting metaphors can now be added that allow users to view, access, and even manipulate video content in novel ways. The MPEG-4 video standard provides key technologies that enable such functionalities. Some typical applications are described in the following list:

- *Interactive digital TV*—Video and audio comprises the main content viewed by the end client in digital television. Apart from viewing video, other media types such as text, audio, and graphics can additionally enhance the entertainment value of viewing video. Examples include customized advertising, which display an image or graphical animation sent to a client. Multiscreen formats, which allow simultaneous display of various programming, such as stock quotes, weather details, traffic details are also examples of how other media types can enhance a viewing experience. Because these media types are individual digital objects not related to the main TV video, they can be interacted with individually. For instance, clicking or pointing your remote to the stock ticker can momentarily take you to stock news or current trading scenarios. Clicking on a weather image can take you to a local weather forecast. This interactive paradigm need not be limited to the secondary media types but can also be extended to the main video. Within a video, you can now think of providing the ability to link the content to other videos, home shopping network sites, and so on. If content is devised by having interest areas in the video tracked over time, then clicking on that interest area could result in an action such as going to a website, or printing related text information on the screen. Thus, coding not only video frames, but also other visual object representation in the scene can bring forth novel interactive television entertainment experiences.
- *Games*—One of the most compelling qualities of games is the level of interactivity it provides a user by immersing him/her in a virtual environment, which is mostly 3D. Enhancing this experience with video, live imagery from different real-life settings certainly makes the games more interesting. This brings the idea of games not being self-contained but making use of different live and static streams from different servers instead. Such broad media access of media objects needs to have a standardized means of representation, access, delivery, and rendering, which MPEG-4 provides.
- *News gathering and TV production*—Many television productions, live or otherwise, now involve a composition of two or more video feeds. For instance, a news reporter in the foreground is reporting about an incident whose video is

playing in the background. These two video signals are typically captured at different locations but need to be brought down to a production station, composited using chroma key techniques and then distributed. Rather than being composited and then transmitted as a single video stream, these can be composited at the end client terminal, which saves time and cost and also has the ability to add graphics/images for specific purposes that may be customized by a client. This allows televised content to be controlled from a user's point of view.

- *Multimedia on mobile devices*—Mobile devices, such as cell phones, PDAs, and pocket PCs, are being increasingly used by the public at large. With wide bandwidths provided by the 3 G networks, it is possible to use multimedia in mobile scenarios rather than just audio communication. For instance, you could think of using mobile devices to access financial information, see news segments, give geographic directions, view airline information, and, in general, access much of the information on the Internet. For these applications, improved coding efficiency of visual media with improved error resilience is a necessity for low-band multimedia services.

5.1 Natural 2D Video

MPEG-4's video compression methods are very similar to previous MPEG and ITU standards in that they use I frames, P frames, and B frames, where I frames are DCT coded and the P and B frames are predicted using macroblock-based motion compensation, leading to the compression of motion vectors and the error images. Please refer to the Chapter 8 on video compression for a detailed analysis of these methods. A few notable differences in MPEG-4 video encoding relate to the description or organization of the video, scalability in the encoding, and advanced global motion compensation techniques, which, if used, can bring down the final coded bit rate.

5.1.1 Hierarchical Video Description

MPEG-4 describes video hierarchically, as shown in Figure 14-11, and thereby supports hierarchical video encoding. At the top level, a video stream is encoded as an audio visual object (AVO), which we will call a video object here since it contains only video.. A scene can be composed of many video objects. Each video object is encoded in a scalable, multilayer form or a nonscalable single-layer form. The first video object layer (VOL_1) specifies the base layer bit stream, followed by one or more enhancement layers (VOL_2, VOL_3 . . . VOL_n). Each of the layers successively adds detail from a coarse to a fine resolution—explained more in the next section. If you recall, in earlier MPEG formats, we had mentioned the video being structured as time-sampled frames, which can be grouped to form groups of pictures (GOPs), containing I, P, and B encoded frames. In MPEG-4, video is structured as time-sampled video object planes (VOP). VOPs can be further grouped in the groups of VOPs (GOVs).

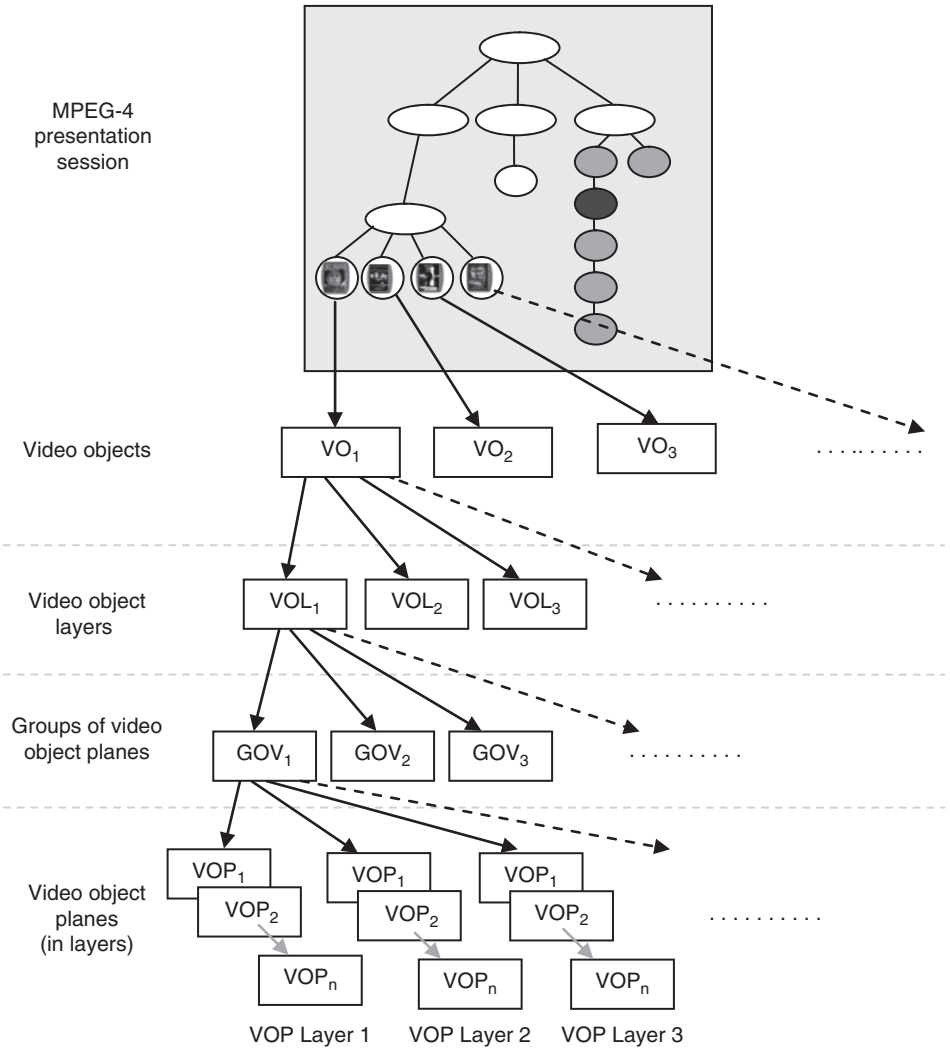


Figure 14-11 Logical hierarchical structure of the MPEG-4 video bit stream. Each AVO video object is described by video object layers, which, in turn, are composed of groups of video object planes, which are finally composed of layers of video object planes.

It is important to further understand the concept of a VOP and its usage in MPEG-4. A VOP can be of any shape, as illustrated in Figure 14-12. A conventional video frame can be thought of as a VOP with rectangular shape. VOPs can be used to describe arbitrary shaped video. In such cases, a complete description of the VOP includes shape information along with YUV texture information. The intraframe and interframe encoding of a VOP works similarly to a conventional frame, which contains motion parameters and error images.

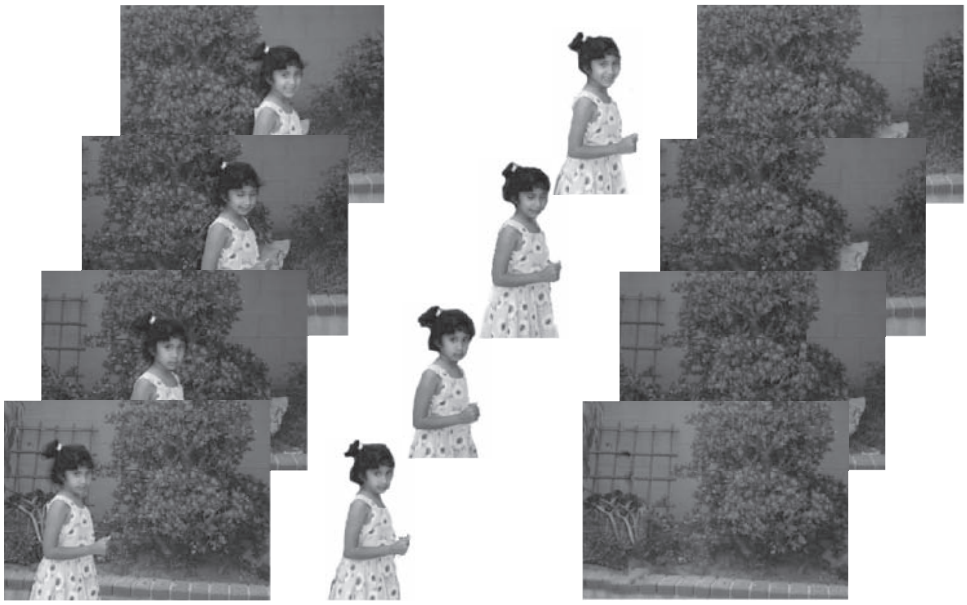


Figure 14-12 Video object layer composed of different VOP layers. The video has a foreground VOP layer consisting of the girl and a background VOP layer. Note that the foreground VOP has an arbitrary shape, whereas the background VOP has a rectangular shape.

In addition, VOPs also contain shape information, which is entropy encoded. The representation of shape in a VOP can be done explicitly or implicitly, as illustrated in Figure 14-13.

Figure 14-13 Representation of arbitrary shaped VOPs. The left image shows a VOP of arbitrary shape. The central set of images shows an explicit representation of the VOP shape, which is given by the alpha channel of the VOP. The image on the right shows an implicit representation, where green chroma suggests background.

- *Explicit representation*—In this case, the VOP texture contains one more channel along with the usual YUV (or RGB) color channels that specify the shape of the VOP. Similar to the alpha channel, this channel is represented as a matrix of values normally standardized to have a 1-bit binary value specifying areas in the image frame that correspond to the VOP shape. It can also be specified as an 8-bit channel to create a more accurate description at the shape boundaries.
- *Implicit representation*—In this case, the VOP is specified by a chroma or color, which keys the background. A constant color (typically blue or green) is chosen to specify the background, while the other colors represent the foreground. Here, the foreground element should not contain any background color pixels.

The approaches for motion estimation and compensation in MPEG-4 are similar to those used in prior MPEG and ITU standards, with the macroblock-based computation adapted to arbitrary shape VOP structures rather than rectangular frames. If a macroblock is entirely contained within the VOP shape, there is no change compared with the prior methods except that the precision of motion vector computation is allowed to go down to quarter-pixel accuracy. If the macroblock lies outside of the VOP's shape, there is no need to perform any motion estimation. If the macroblock lies on the shape boundary of the VOP, the motion vector for this block is computed using not the entire block but a modified block where the mean absolute difference (MAD) or sum of absolute difference (SAD) computation is performed only for those pixels of the macroblock that belong to the shape of the VOP. This improves the overall efficiency. The three modes for encoding a VOP are enumerated in the following list and illustrated in Figure 14-14.

- *Intra VOP (I-VOP)*—Where the VOP is encoded independently of any other VOP. This is similar to an I frame.
- *Predicted VOP (P-VOP)*—Where the VOP is predicted from a previously decoded VOP. The error VOP is coded along with the motion vectors (and shape).
- *Bidirectionally predicted VOP (B-VOP)*—Where the VOP is predicted based on past as well as future VOPs. The error VOP is coded along with motion vectors from both VOPs per macroblock.

5.1.2 Video Encoding Scalability

Scalability in MPEG-4 is implemented using video object layers (VOLs). These are typically defined by a base layer VOL_0, which encodes a coarse representation of all VOPs and optionally defined enhancement layers VOL_1, VOL_2 . . . VOL_n. Each enhancement layer adds more resolution onto the previous set of layers. The encoded information of each layer represents an elementary stream, which is further packetized and multiplexed together (along with other AVO elementary streams), as shown in Figure 14-15. This scalable encoding is effective to manage bandwidth changes in a single transmission session. It also helps manage data by encoding the content once but distributing it to multiple clients served on different bandwidths.

The scalability can be defined to be spatial or temporal. In spatial scalability, the enhancement layers improve upon the spatial resolution provided by a VOP. The base layer VOPs are computed by downsampling the original VOP and encoding it. The enhancement layers are computed by decoding the base layer VOP, upsampling, and finding the difference compared with the original layer. Enhancement layer VOPs are

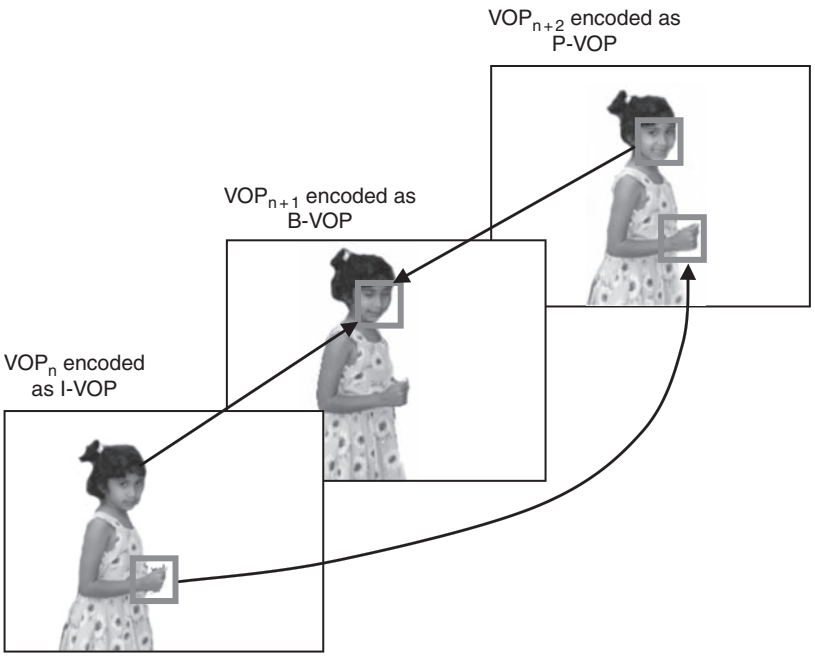


Figure 14-14 VOP encoding. There are three modes of VOP coding. I-VOPs are coded without any information from other VOPs. P-VOPs are predicted based on a past VOP. B-VOPs are predicted based on a past and future VOP.

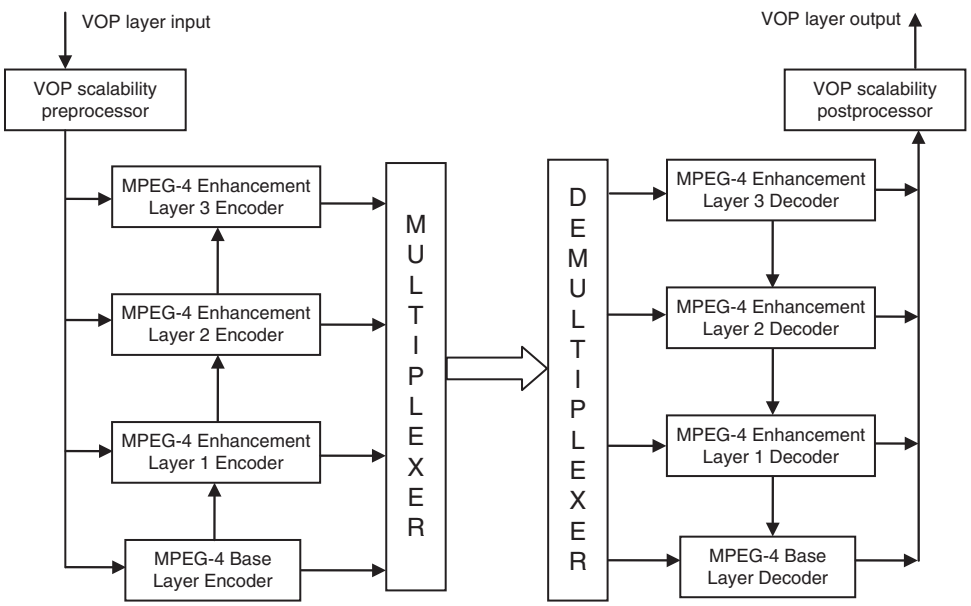


Figure 14-15 Block diagram of MPEG-4 scalability framework. The input VOP stream is encoded at different resolutions. The scalability preprocessor implements the desired spatial or temporal scalability.

encoded as P-VOPs or B-VOPs only. Spatial enhancement layers can be hierarchically defined for each VOP, as illustrated in Figure 14-15.

In temporal scalability, the enhancement layers add more information temporarily by inserting frame data left out in the base layer. Temporal scalability improves overall smoothness of motion in the sequence of VOPs. Because a VOL is composed of GOVs, the enhancement layers can improve temporal scalability for specific VOPs, the base layer, or for all the VOPs in the entire base layer. Both these cases are illustrated in Figure 14-16 and can be described as follows:

- *Enhancement Type I*—Where a few selected VOPs are enhanced giving better temporal resolution for specific parts of the frames
- *Enhancement Type II*—Where all the VOPs are enhanced

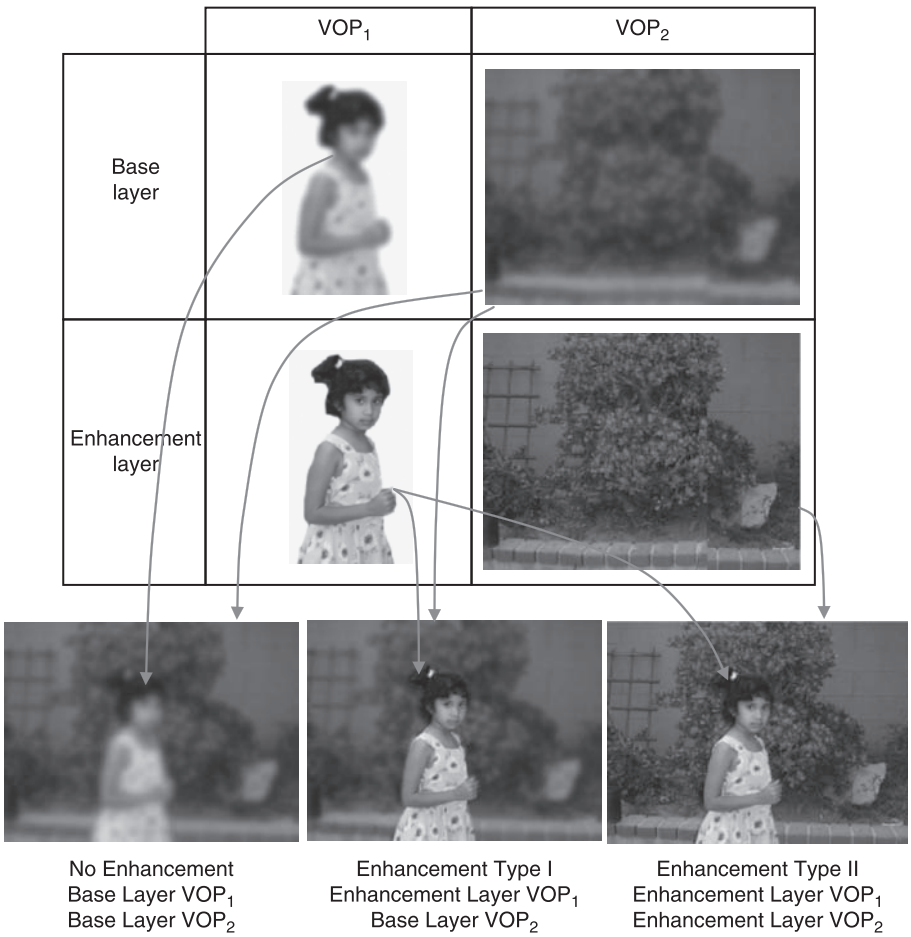


Figure 14-16 Temporal scalability in MPEG-4 video. The top table shows the two VOPs at their base layer and enhancement layer resolutions. These can be combined to produce enhancements on a frame. Illustrated are no enhancement (lower left), enhancement of Type I (lower middle), and enhancement of Type II (lower right).

5.1.3 Global Motion Compensation

The motion compensation techniques, discussed in Chapter 8 and used in prior MPEG/ITU standards, are known as local motion compensation techniques, where the movement of a local macroblock is predicted. Global motion compensation (GMC) attempts to predict larger areas with a single motion vector, as against one motion vector for small, single macroblocks. This is especially useful to predict motion of background caused by camera movements such as pan, tilt, and zoom. Here, a single motion vector is used to encode the motion of all the background pixels. To enable GMC, the foreground pixels or (macroblocks) need to be separated from the background. The foreground blocks are termed as sprites and can form a VOP by themselves. The background VOP has only one motion vector and, thus, helps save bits. An example of global motion compensation is shown in Figure 4-17. Several mathematical models have been proposed to estimate the global camera motion, but MPEG-4 has provided a standardized bit stream, which optionally contains global motion compensated VOPs. GMC is a valuable concept to improve video coding efficiency but has the disadvantage of large computation times, making it not useful for real-time encoding.

5.2 Synthetic Video Objects

Synthetic objects correspond to 2D or 3D graphical representations and animations. The representations of such entities in MPEG-4 are based on the prior VRML standard. By using nodes to describe vertex positions, interconnectivity, and nodes, such as the *Transform, IndexFaceSet2D, IndexFaceSet3D*, you can define where in the scene the 3D object or 2D object is placed. Over time, by updating these nodes, the polygons that define the object can be animated using routes and conditionals, as illustrated in Figure 14-5. The animation can change properties of the position, coordinates of any vertex, the texture maps coordinates, and so on. Apart from generalized 3D representations and animations, the MPEG-4 Version 2 standard defines specific representations for animating faces and bodies with a goal to use a synthetic human face and body models with animations in visual communication. In applications such as video teleconferencing and multiuser chat rooms, a 3D (2D) representation of the human face is used to portray visual manifestations of speech and emotion. Rather than compress the video of a talking person's head, higher compression can be achieved by representing the head as a 3D (or 2D) model at the decoder. Similar setups can be set up for the body in applications such as games and e-commerce.

The specification of the face or body model and the animation setup is distributed across two sections in MPEG-4: Systems and the Visual AVOs. The Systems section has standardized a way to represent and code the geometry of a face or body and methods to deform the representation. These are called the *Facial Definition Parameters* (FDPs) and *Body Definition Parameters* (BDPs). These are communicated in a *Face Body Animation* (FBA) node in the BIFS stream. The Visual section specifies the coding of animation parameters that are used to drive the FDPs and BDPs. The animation parameters are respectively called *Facial Animation Parameters* (FAPs) and *Body Animation Parameters* (BAPs). The FAPs and BAPs can be extracted automatically from a video stream and communicated to the decoder. They can also be created as a result of animations created on a face or body by an artist or animator. The FAPs can

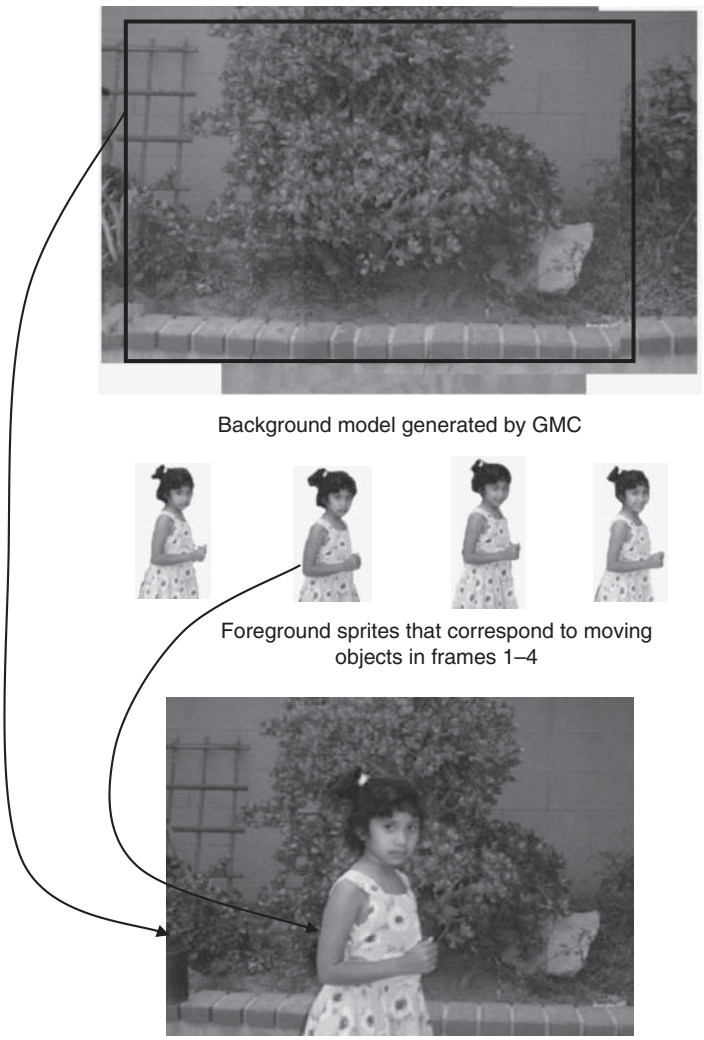


Figure 14-17 Global motion compensation. For the sequence of frames shown in Figure 14-12, global motion compensation computes the foreground elements in motion frame by frame (shown in center) and a model of a background panorama (shown at the top), which is built at the encoder and communicated to the decoder. On a frame-by-frame basis, the decoder needs to know what part of the panorama forms the background and can then composite the foreground VOP to create a frame (shown at the bottom).

also be extracted from a speech signal and used to specify mouth positions. Either way, the encoder embeds animation parameters and communicates only these parameters to the decoder. The decoder uses these parameters to animate the face (and body) model that is specified in the FDP or BDP. Some of the groups of FAPs are illustrated in Figure 14-18.

Group	Number of FAPs	Sample FAP values
Vowels (Visemes) and Expression	2	Phoneme “s”it, phoneme “l”ump, Expressions—joy, surprise
Inner & CornerLips, Jaw, Chin Positions	16	Top inner lip, bottom middle lip, left corner lip
Outer Lip Positions	10	Left upper outer lip, right outer bottom lip
Eyelid, Eyeball, Pupil Positions	12	Eye blink, eyes look left, right.
Brows	8	Inner brow, outer brow, furrow
Cheeks	4	Squint, cheek raiser
Tongue	5	Roll
Head Rotation	3	Tilt, look up, look down
Nose/NasioLabial	4	Nostril dilator
Ears	4	Ear left, ear right

Figure 14-18 Facial Animation Parameter (FAP) groups in MPEG-4

6 SYNCHRONIZATION AND TRANSPORT IN MPEG-4

Compared with its predecessor formats, MPEG-4 has been designed to adapt to a variety of operating scenarios that involve local and remote delivery for broadcast and multicast over various networks. The delivery layer in MPEG-4 has been abstracted to provide the functionality that can be molded to different digital distribution scenarios. MPEG-4 systems define how a presentation composed of media objects is organized using a scene description along with individual elementary streams. The elementary streams are encapsulated within an object descriptor with stream data consisting of access units. These multiple elementary streams have to be delivered in a synchronized manner, which is done by a separate synchronization layer.

The synchronization layer provides an extensible syntax that allows access units to be encoded with all relevant timing information, as defined by the systems compression. The individual entity of the synchronization layer is an SL packet. Access units from the systems layer for each elementary stream are, thus, converted into SL packets, which can then be exchanged with the delivery layer. All the SL packets of one elementary stream are converted into an SL packetized stream. The SL packets help in fragmentation of access units, so larger access units can be broken down with appropriate synchronization and passed onto the delivery layer. Also, each elementary stream can have its own synchronization definitions, which can be defined in a SLConfigDescriptor.

The delivery layer in MPEG-4 consists of a Delivery Multimedia Integration Framework (DMIF) that deals with the relevant details specific to different delivery technologies—IP networks, storage, broadcast, and so on. The delivery architecture of DMIF is linked to the systems layer (and synchronization layer) via an application interface called the DMIF Application Interface (DAI). This clean separation of the delivery

logic from the systems logic has allowed MPEG-4 to extend its delivery mechanism to different operating scenarios. The overall architecture is illustrated in Figure 14-19 followed by a brief discussion on typical transport setups to which the DMIF layer can be adapted.

6.1 MPEG-4 Transport over MPEG-2 TS

MPEG-2 has defined a standardized format that allows for transport and storage of multimedia data over MPEG-2 networks. MPEG-2 transport streams (MPEG-2 TS) provide a

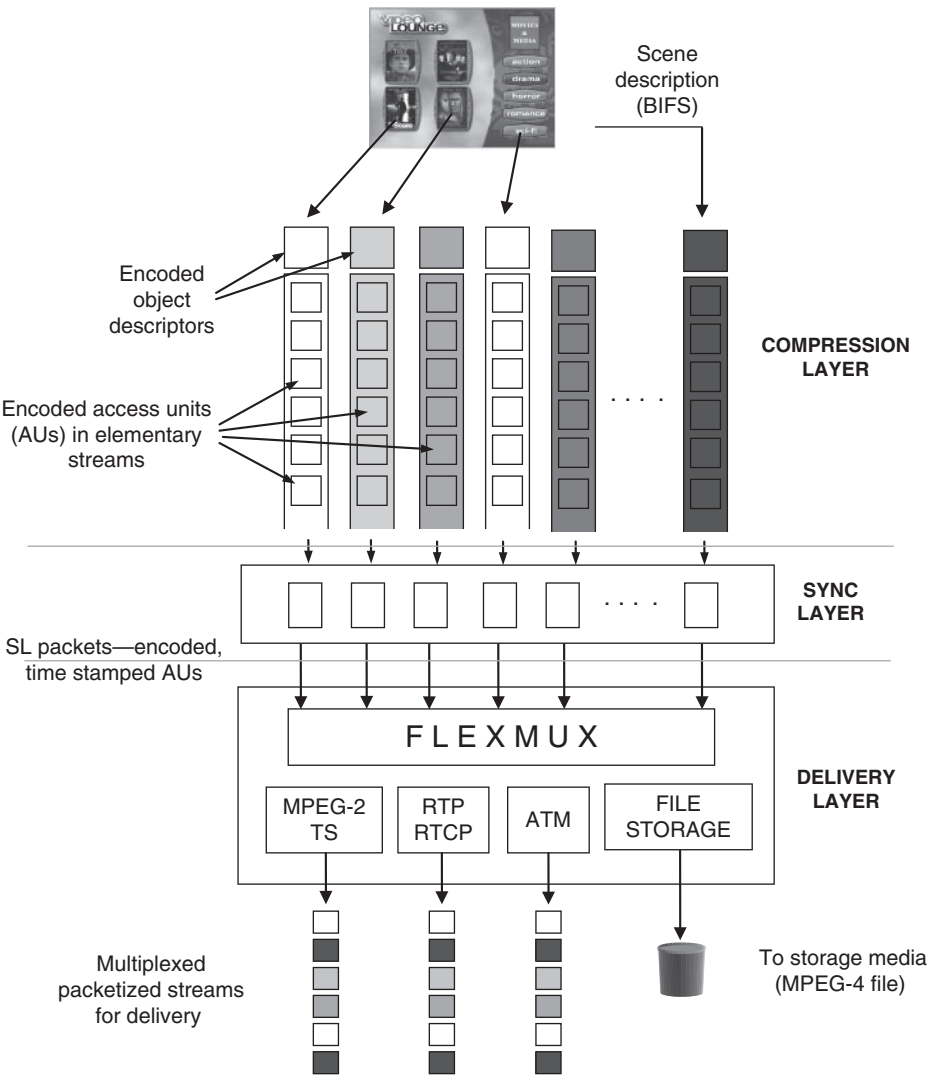


Figure 14-19 MPEG-4 systems overview, showing the compression, synchronization, and delivery architecture

means to transport MPEG-2 data for real-time digital broadcasts on cable/satellite networks. The program streams (MPEG-2 PS) allow representation of data for storage on DVDs. Delivery mechanisms using the MPEG-2 transport stream have been especially well set up with investments and infrastructure for digital television. It makes economical sense for MPEG-4 content delivery to make use of the same transport mechanism. MPEG-2 TS works by mapping elementary stream data into a sequence of transport packets, each of which is labeled by a specific packet identifier (PID). These TS packets have a fixed size (188 bytes). To transport the compressed BIFS and audiovisual elementary streams of MPEG-4, the data of the elementary streams has to be multiplexed into the MPEG-2 TS packets. Normally, for this procedure, the FlexMux is used to produce an intermediary stream similar to the MPEG-2 packetized elementary stream (PES), which is then mapped into the transport stream.

6.2 MPEG-4 Transport over the Internet

There are many options for transporting multimedia data over the Internet, the most popular being the Real-Time Protocol (RTP) or the Real-Time Streaming Protocol (RTSP). RTP has been designed for computational efficiency by respecting byte boundaries and 32-bit boundaries. It also expects that the payload carried by the RTP packets be identifiable. This semantic identification of the packet data (as being CELP compressed audio, or BIFS, or RTP) facilitates the client to perform stream-specific recovery in case of delays and loss during transport of that stream, while other streams can be decoded/rendered in a synchronized manner. Therefore, when using the RTP protocol to transport MPEG-4 data, the MPEG-4 elementary stream data can be encapsulated into RTP packets in a manner that also identifies the payload carried by the packet. Additionally, because an MPEG-4 presentation can consist of lots of elementary streams, there needs to be a mapping of each elementary stream to an RTP stream. This mapping is normally provided by the FlexMux layer.

7 APPLICATIONS CURRENTLY USING MPEG-4

It should be clear by now that MPEG-4 is a family of open international standards that provides tools to create and deliver multimedia content. These tools include excellent codecs for compressing conventional audio and video and also those that form a framework to create rich multimedia as a combination of images, audio, video, graphics, and interactive features. MPEG-4's latest video codec, the AVC, which is identical to the H.264, is now being employed in new video products and services because of its compression efficiency, such as HDTV satellite broadcasting, BluRay high-definition optical discs, the now defunct HD-DVD, video broadcasting on mobile handsets, and so on. The general-purpose audio codecs such as AAC are now used in low bit rate music services such as Apple's iTunes, XM satellite radio, and music download services such as for mobile carriers like KDDI Corporation. SONY's PlayStation Portable uses MPEG-4 AVC and AAC services. Although these examples illustrate the use of specific codecs in MPEG-4 used individually for audio or video, most of MPEG-4's richness comes from the tools that allow you to build and deliver interactive applications that contain

many media types. Building interactive content is possible because of the advanced constructs allowed by the systems layer. This has yet to be fully exploited, but the following sections enumerate a few such applications that have been deployed or are soon to be deployed that showcase the use of MPEG-4 technology.

7.1 Television Broadcasting

MPEG-2 is the current standard of choice for digital television content creation and distribution over cable, satellite, and so on, although it does not offer enough bandwidth savings compared with MPEG-4 AVC. The change to move to MPEG-4 AVC was not welcomed thus far, probably because of an already existing and well-working distribution infrastructure, which involved a lot of investment. But now as programming moves to HDTV, major TV operators such as DirectTV have either announced to use or have already started using MPEG-4 AVC.

7.2 IP-Based Television Distribution

With the rise in broadband IP networks, IP-based television distribution has become a viable distribution medium for digital television and is now an alternative to cable, satellite, or over-the-air broadcasting. Additionally, because IP networks are inherently two way, IP-based television provides improved and interactive video services. IP-based television systems use high-efficiency coding provided by MPEG-4 AVC to allow for multiple stream delivery over a single connection. Thousands of AVC encoders have already been deployed and a number of system operators have already launched services in the UK, Europe, and now also in the United States.

7.3 Mobile Communication and Entertainment

MPEG-4 addresses a broad delivery space from low bandwidth to high bandwidth using profiles. Profiled content enables networks and devices of limited capabilities to consume MPEG-4 content only if they support that profile. MPEG-4 Simple Profile has been a part of the 3G mobile standards for a few years now and has enabled video telephony over cell phone networks. NTT DoCoMo has deployed millions of MPEG-4 handsets to enable two-way video calls and even watch video programming streamed over 3G networks. In addition, MPEG-4 HE-AAC audio coding has been employed in several music download services. HE-AAC codec's next version along with the parametric stereo tool has been selected as a new standard for music streaming by 3GPP.

8 EXERCISES

1. [**R03**] The natural video encoding part of MPEG-4 has an option to use global motion compensation (GMC), which is supposed to further help compression by transmitting background and foreground blocks separately.

- Compare and contrast GMC with traditional local motion compensation. How does GMC differ? What does GMC effectively give you?
- Given that a frame is divided into 16 × 16 macroblocks, how would you compute GMC? How would you separate the background from the foreground objects?
- Does your computational strategy change if the camera looking at a scene is static or if it is moving?

2. [**R02**] Video object planes (VOPs) is a term introduced in this chapter and used in visual coding of pictures.
   - How is a VOP different from a frame? How do you represent VOPs?
   - How do you encode VOPs?
   - What use does a VOP have during encoding?
   - What does it mean to implicitly define a VOP—as compared with explicitly defining? Also, when explicitly defining the shape of a VOP, the shape information is compressed losslessly—why is this?
3. [**R03**] VOPs are traditionally encoded in the same manner as frames are encoded in the prior MPEG frames, except that motion estimation is now used in the context of a VOP rather than a frame.
   - Define the terms I-VOP, P-VOP, and B-VOP is.
   - What happens when a macroblock lies within a VOP, outside a VOP, or on the boundary?
   - What are the modes of VOP encoding? Can B-VOPs be predicted from previous B-VOPs? Give reasons for your answer.
4. [**R03**] One aspect of visual encoding that is advanced in MPEG-4 as compared with previous MPEG versions is the use of encoding scalability.
   - Explain what encoding scalability is and how it is handled in MPEG-4.
   - What is the difference between spatial and temporal scalability?
   - Spatial scalability does not use I-VOPs but only P-VOPs and B-VOPs. What is the reason for this?
   - Temporal scalability can be achieved for portions of the data. Explain how this is possible with MPEG-4. Can you give an example when such an encoding will be useful?
5. [**R03**] Most media players, for example Windows Media Player, Apple QuickTime Player, RealPlayer by Real Network, do support MPEG-4 encoded visual streams, but still only play a single A/V stream. MPEG-4 with its scene description capabilities can do more than this.
   - If you want to implement a simple A/V player, which plays only one audio and one video at a time, describe what the scene would look like in MPEG-4.
   - What additional advantage does an MPEG-4 based representation have over the other traditional A/V players for such single A/V stream viewing?
   - If you had to support a Picture in Picture (PIP) functionality, how would the scene layout and functionality look? Write pseudocode that does this, similar to the one described in the text. Note: A PIP setup has a smaller video of another stream displayed. If clicked on, the main A/V

stream switches back and forth between the main viewing area and the PIP window.

6. **[R03]** You want to create a "sound" application where the final rendering requires natural music, MIDI synthetic music, and the natural voice of a singer.
   - If this system was to work with the natural and synthetic audio on a hard disk, with a live performance of the singer, how would you design this application to work in MPEG-4?
   - Additionally, if you wanted to show synchronized text that a singer can read and sing, how would you do this in the preceding design? Note: This now becomes like karaoke.
   - If the natural instrument music was now played live by a music player, and also the singer sings live, how would your system design change? What are the requirements to make this system perform in a synchronized manner?
   - How do these requirements change if the singer and the player are in two distant countries and need to present a live performance at a third location?
7. **[R03]** Along these same lines, explain how you would implement a DVD player type of application using the systems-level scene description capabilities in MPEG-4. Your MPEG-4 based setup should have the standardized play, pause, and stop capabilities as you are viewing the stream. Additionally, you should be able to select chapters, select a language for audio, and have optional features such as viewing the "making of", "director's cut," and so on.
8. **[R03]** The previous few questions illustrated the need and importance of a scene description and its necessity when describing complex content. Another orthogonal way to describe complex scenes and interactivity is by using Web-based programming standards such as JavaScript, Dynamic HTML, and servlets, which have made complex and dynamic Web pages possible and provide a great deal of interactive Web-based setups and applications.
   - If an MPEG-4 terminal has to be implemented on a computer, can you use Java, JavaScripts, servlets, ActiveX control embedding and other tools to emulate what an MPEG-4 systems-level description can provide? Where do you see similarities, shortfalls, or even advantages?
   - If all the Web-based protocols work well on desktop platforms for the type of applications for which MPEG-4 systems were designed, you could easily emulate them on other embedded and terminal platforms, thus not needing the new MPEG-4 systems standard. Comment on this arguing for or against the need.
9. **[R03]** All streams in MPEG-4 are compressed and represented as elementary streams. This is true for all streams, including audio and visual streams.
   - What are elementary streams and what do they contain? Qualitatively describe what a visual and audio elementary stream contains. How are they organized?

- What other elementary streams can you think of besides audio and visual elementary streams?
- In any MPEG-4 presentation, which elementary stream(s) needs to be delivered first, and which ones can be delivered or streamed later?
- The 2D/3D graphics content and animations do not have their own elementary stream. In which elementary stream are they described? What is the reason for them to not have their own elementary stream?
- To follow up on the previous question, if 2D/3D graphics were designed to have their own elementary stream, give sample functionalities that are made simpler by having graphics in their own elementary stream. What becomes difficult in such cases?

10. [**R04**] All elementary streams in MPEG-4 are encapsulated logically within an object descriptor.
    - What does the object descriptor of a stream contain?
    - Why do you think this level or type of encapsulation based on an object descriptor is necessary? How does it help?
    - What do the streamType and objectTypeIndication provide? Where in the object descriptor do they reside?
    - Work out possible values streamType and objectTypeIndication can take for an AVO that is supposed to be an audio stream. You will not know the exact codes unless you refer to the MPEG-4 systems specification, but describe them qualitatively.

This page intentionally left blank

CHAPTER 15

Multimedia Databases and Querying

Digital media is used everywhere today—for communication, entertainment, advertising, and many other forms of information exchange. This includes video, audio, images, text, graphics, and various combinations of all. Additionally, inexpensive media recording and storage devices, good media authoring software, and an increase in accessible bandwidth over wired and wireless networks have enabled wide-scale pervasiveness of digital content. People record an enormous amount of personal media, such as photographs, home movies, and music. Commercial content houses such as studios, advertising agencies, and game companies constantly create digital content for entertainment purposes. A variety of multimedia applications communicate media, which is either stored or live, such as video/audio-on-demand, digital teleconferencing, digital television, distance learning, and so on. The communication is further facilitated by a variety of standards for compressing and distributing content, including MPEG (MPEG-1, MPEG-2, MPEG-4) and ITU (H.261, H.263, H.264, and so on). Furthermore, Internet and wireless media communication allow access to these media applications on desktops, laptops, and personal mobile devices, such as cell phones and PDAs.

With the explosion of multimedia digital information comes the need to browse, search, and catalog the media information effectively. Audiovisual multimedia data might be consumed immediately by a viewer, but there are applications now where the information is created, compressed, and saved for indexing and accessing later. For instance, you might want to produce a short presentation of a memorable occasion (such as a wedding, graduation, or bar mitzvah), which requires searching and sorting through thousands of pictures, movies, and audio clips. In a newsroom, the editors might need to quickly produce a news presentation from a very large set of stored digital media, following the death of a well-known personality. Other practical

applications might include the selection of specific programs based on semantic queries from a wide range of audiovisual entertainment content such as that provided by the numerous channels on cable television. This might be simple, such as listing all action movies or drama movies, or more complex, such as listing all documentaries that show the conservation of forests in South America.

The central difficulty stems from the dichotomy between data and knowledge. Although a movie clip might represent an interview with a movie star, it is stored as a set of frames, each one being a set of pixels, and, thus, very similar to any other movie clip. In the case of text only, the difficulty still exists, but is much smaller, as words are often a good approximation of semantic meaning, at least for search purposes. For instance, it is easy to search for text documents that discuss multimedia by searching for the word *multimedia* in all documents. Google is a prime example of success in text-based search. However, it is not so easy to search for a specific car in images (even though all images in the database might consist of cars), for a video that demonstrates the hunting behavior of lions from all videos available in a digital video library, or for a specific Christmas carol among many audio songs. The complexity of solving such queries stems from the need to extract semantics from the signals being queried, which is somewhat easier in the case of text only.

Many approaches in research literature make use of specific visual and audio properties. For example, image retrieval might rely on the color, texture, or shape of an object or even spatial interrelationships if they can be reliably extracted from an image. This is further complicated for video objects where, additionally, these properties might change over time. In video, object scene identification and object tracking are techniques used to extract meaningful semantics. Querying video content can also additionally make use of its accompanying audio track and analyze sound levels, frequency changes, and so on. However, developing formal methods for extracting semantics in a reliable manner is still a difficult, open problem.

Today, most media-based searching and indexing methods attempt to solve the problem by annotating the signal with added text information, which includes date of creation, topic, and other keywords that describe what the content is about. This information is normally termed as *metadata* and can be associated either automatically by a program or input by a user, group of users, or experts. Research to understand and extract metadata automatically from media signals is being constantly investigated by the academic and industrial research communities. Metadata can also be provided by users by inserting semantic, contextual text information by authors, content providers, subscribers, or consumers of the media signal. It is clear, then, that to make media searching and indexing a practical reality, we must define a standardized way by which metadata can be created and associated with media signals. The standardization of defining media metadata appropriately will help to create scalable applications to search, index, and make media databases useful in much of our day-to-day queries.

In this chapter, we hope to explain some of the problems, processes, and solutions for creating and inserting metadata with the media data signal. Section 1 explains the difference between multimedia content and multimedia context and the kinds of queries that need to be processed on multimedia data. Section 2 describes the requirements of a multimedia database capable of indexing media composed of video, audio, images, and so on. Section 3 explains the importance of metadata and its use in

indexing. Various standards have been proposed to deal with metadata. Section 4 enumerates some of the relevant ones by the MPEG community (*MPEG-7*), the Society of Motion Pictures and Television Engineers (*MXF with DMS-1*), as well as others in use by news agencies (IPTC's *NewsML*, *SportsML*, and so on) and for the Internet (*Dublin Core*). Multimedia queries often result in lots of data, which needs to be presented in efficient ways so that the user can appropriately browse through it. Section 5 deals with some of the issues involved in presentations and user interfaces. Finally, Section 6 briefly enumerates a few commercial multimedia projects now in use.

1 MULTIMEDIA DATA VERSUS MULTIMEDIA CONTENT

Any information retrieval system that indexes audiovisual information relies on key words, key expressions, or other advanced semantic concepts. Broadly, these can be termed as *semantics*. In the case of textual databases, text strings provide a direct, simple approximation of semantics that generally suffice for performing a variety of text-based operations, such as browsing, searching, summarization, linking, and so on. In the case of visual or audio media types, these semantics are much harder to infer automatically. Furthermore, the semantics could also depend on the context in which multimedia data is evaluated. Normally, two approaches are followed:

- Directly extract the semantic information from the media content. When attempting to do this automatically, most methods look for specific properties in the image that can be later used for querying. Examples of such properties might include dominant color and degree of frame-to-frame motion in the case of images and video. You can also do this manually by asking a user to describe the media object and/or manually fill in the required semantics.
- Indirectly extract information gathered from the context of the media object. This might include text information from the neighborhood of a media object on an HTML page or information from the header of a file providing details that relate to the content of the file, for example, the date and time of capture.

Processing queries on multimedia data becomes useful and effective only when the queries are content based, which requires extracting semantic tokens or objects and semantic interrelationships between these objects. This is perhaps easier done for text strings and text databases rather than multimedia data; hence, there are many text-based databases today. The reason for successful indexing and searching for textual data can be attributed to a few reasons, which are enumerated in the following sections.

1.1 Semantic Extraction

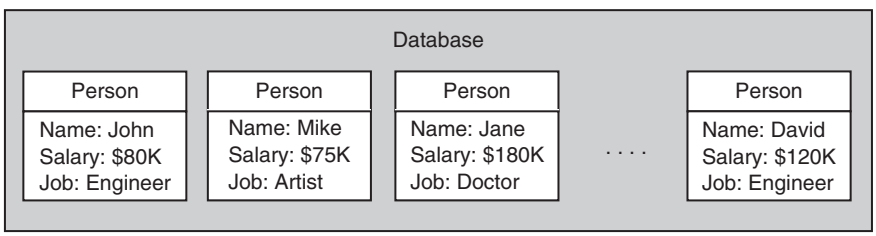
First, considerable progress has been made on formal language and grammar theory, allowing the extraction of useful semantics from text—words, sentences, and documents. Though complex semantic queries such as "Why did the author of this text intend to write a book on multimedia?" are still hard to process, parsing the sentence to provide semantics for useful search queries is definitely plausible: "Find all texts

that contain the words 'multimedia, video, compression, standards'." The equivalent easy query for video data might be "Find all videos that contain more than 1700 red pixels (255,0,0)." Although this query is easy to process, it is unlikely to occur in a real context. A more likely query might be "Find all videos that contain racing cars." This requires extraction of objects and getting into semantic understanding of the data, which is not yet practically possible for many of the media types, such as images, video, and audio. An even more complex query that has a higher embedded semantics and is very succinct and useful, but virtually impossible to address with today's state of the art in multimedia databases might be "Find all video clips of red racing cars in accidents where the driver was not hurt."

1.2 Query Processing

Once data is stored in a database, retrieval is performed by applying queries against it. Queries contain predicates and are normally expressed as "Find all records (or files) that contain the word multimedia," or "Find all books about multimedia and also discuss cinema technology." One reason why text databases work very well is because these queries can be abstracted easily using relational algebra that was proposed in the 1970s. Relational calculus can then be used to index and efficiently retrieve objects or documents that match the query. Some of the relational operations include *selection* (σ), *projection* (π), *product* (x), *join* ($\bowtie$), *union* (U), *intersection* ($\cap$), *difference* ($-$), and *rename* (ρ). A sample query using these is shown in Figure 15-1.

There are not such tools to work on multimedia data, partly because extracting semantics is difficult. Moreover, the semantics extracted from multimedia data change depending on the context and observer. Developing a formal representation for semantics in such cases is not straightforward because the media types are so varied. For instance, audio data can be queried using simple semantics such as words (lyrics),



Query 1: Find all persons who are engineers.
Relational Expression – $\pi_{job = engineer}$ (σ (Persons))

Query 2: Find all persons who are engineers and who earn more than 100K.
Relational Expression – $\pi_{job = engineer}$ ($\sigma_{salary > 100K}$ (Persons))

Figure 15-1 Relational databases. A simple database consisting of "Person" records is shown at the top. Also shown are two simple queries on the database expressed as a text query and also translated into relational expressions. Relational expressions and operations are used to get results for queries.

frequency levels, or complex semantics, such as specific sounds, patterns of sounds, and intonations. Images can be queried based on colors, shapes, textures, or objects, whose semantics are completely different from those of an audio signal. Video, besides being a sequence of images, also has a temporal dimension that complicates the semantics even further, for instance the person in the video might be walking, running, angry, tired, and so on. Querying multimedia data requires new interfaces different from those supporting text databases. For instance, one way to query is by providing an example of the content. You could provide examples such as a drawing, a photograph, or a melody. The query processor could then use that example to find others in the database that look like it or sound like it.

1.3 Nature of Multimedia

Other reasons why multimedia search and querying are complex relate to the very nature of multimedia. Multimedia data is heterogeneous, voluminous, and often compacted in proprietary formats. All this makes accessing, searching, storing, retrieving, and communicating data slow and cumbersome over digital networks. You might say that multimedia compression techniques for video, audio, and images have reached a mature level with standardized and interoperable bit streams as proposed by standards bodies such as JPEG, MPEG, ITU, VCEG, and so on. However, the adaptation of these digitally stored representations to different applications accessing different bandwidths and running on different end terminals is still not a fully solved problem. Compare and contrast two situations: running a multimedia query on your computer with fiber-optic access to a large database of video versus running the same query on the same database from your handheld device with limited rendering capabilities and much smaller bandwidth.

To conclude this section, it should be mentioned that while progress will go on to address the above-mentioned areas, the one practical approach that has come about is the use of metadata. Metadata can be considered as a descriptive placeholder of multimedia data that can be stored with or away from multimedia data, can be dealt with for querying, and can be transported or presented independently of the multimedia data. The next section addresses the issues associated with metadata.

2 MULTIMEDIA METADATA

"Meta" in Greek corresponds to "beyond" and relates to abstraction. Hence, metadata is a term used for data that describes data. Metadata is commonly used to describe multimedia content. As a result, it can be used to search, organize, or index multimedia information. A query occurs at the semantic level, which is not directly accessible in the data. Metadata provides the link between the semantic level of the query and the data. Metadata, especially in the form of text, makes it feasible to process complex queries at the metadata level, and to index and search in multimedia information. Media data may consist of images, video, audio, 2D/3D graphics, and combinations of them in either a native uncompressed format or various standardized compressed formats. Metadata is used to describe the data in either a frame of a video,

an image, a group of frames, or an audio clip. Metadata can describe content both spatially and temporally. Metadata that describes a multimedia object or resource can be viewed from various perspectives based on how the metadata is created or who has created it at the content creation, compression, distribution, and client consumption stages. Correspondingly, apart from general indexing and searching, metadata can also be tailored for specific use, such as the following examples:

- During the authoring stage, metadata that is inserted by a content author normally includes biographical and bibliographical information of the author and perhaps the content creation process itself, such as the author name, content title, recording devices or software used, creation data, resource format, and so on.
- During the distribution stage, metadata is added to describe information needed for retrieval, and even to authenticate various clients or markets. These can include various formats for encoding, various bandwidths for distribution, and semantic information about the content, such as actors and actresses in a movie or players in a game. Metadata can also be useful in a distributed system to describe any resource adaptation capabilities.
- During the client consumption stage, metadata can be used to enhance the viewing of the multimedia content and to browse through a presentation.

Metadata is anything that qualifies the media data. It can consist of keywords, expressions, and semantic information that can be used for applications that involve indexing, searching, and organizing multimedia data. It can consist of descriptions, end platforms it can be consumed on, ownership, and user information that can be used for tracking, distribution, authentication, and rights management in the content creation and distribution pipeline. Because the definition and usage of metadata is very broad and varied, a number of issues regarding its creation, storage, and management need to be addressed. Various efforts are being made to address the issues revolving around metadata, such as its automated and semiautomated creation, its representation and standardization, its storage, and its overall management throughout the life cycle of the media content.

2.1 Creation/Extraction of Metadata

Metadata information is either derived manually or automatically extracted. Examples of manually inputting metadata in commercial applications include making a table of contents for an e-book, closed caption insertion in broadcast news programs, or language text preferences while watching DVDs. Because manual annotation is expensive and time consuming, most sources of digital data today are still produced and delivered without any attached semantic information. Automated or semiautomated techniques for extracting the semantic structure of such raw data are clearly needed to address the organization, search, and retrieval of much of the media information available today. Designing these techniques has been the challenge of the research community. Much progress has been made in the fields of image processing, computer vision, audio processing, and artificial intelligence, and viable automated methods have been used in specific domains—such as surveillance in low-traffic areas, certain sports where it is

easier to track players moving against a green floor, and converting speech to text. The state of the art, however, is far from being able to fulfill the need of applications.

2.2 Storage of Metadata

Metadata can normally be stored in one of two ways: either internally in the same file as the media data or externally in a separate file. Both options have various advantages and pitfalls. Internal storage couples metadata with the media data and, thus, allows easily transferring the two together. The importance of this should be considered when metadata descriptions need to be synchronized with the media object. However, when stored together with media data, it increases transmission bandwidth. Also, there is always a chance of duplicating metadata or increasing the metadata footprint that goes with the signal. The redundancy issue, if not appropriately managed, can significantly increase the overall file size of the content when metadata is stored with it.

2.3 Metadata Management

The importance of metadata for applications that perform searching, indexing, and other organizational tasks should now be clear. Another aspect of metadata is that most scalable applications need a set of guidelines to manage the metadata, both for human operators and software processes. Some of the metadata management issues that typically come up in a large-scale system or application are as follows:

- *Creation/change/removal of metadata*—Metadata needs to be added to content when it is introduced into a database. The metadata might also change: A user might add more information, delete it, or change it. The metadata could also be time dependent and needs to be updated at specific times, for instance if the metadata has date-specific information. If the content itself changes, the metadata recorded needs to be updated to reflect this change. Metadata might also need to be changed to adapt the content for specific applications.
- *Access of metadata*—In any application or system, metadata needs to be accessed for purposeful queries. There should be a well-defined interface that allows access to read the metadata in a quick, transparent, and organized manner.
- *Proxy caching the metadata*—Most systems today work in a distributed environment in which both the underlying data along with its metadata should be cached at proxies to decrease the access time to the data/metadata.

3 MULTIMEDIA SYSTEMS AND DATABASES

Multimedia databases are very different from traditional text databases for a variety of reasons. First, describing media objects is very different from traditional text or numerical documents. Text is a description in itself that can be searched using specific keywords and phrases. Media object descriptions are not as straightforward or even standardized and can vary from person to person. Second, the disk storage requirements of media

objects are enormous compared with text, forcing a multimedia database to use scalable and distributed architectures as compared with monolithic setups common to text databases. Also, the viewing operations, browsing, and manipulation paradigms for media objects are different from textual documents. For instance, while searching text documents, a paragraph of text containing the search keywords can be displayed with the option to load the entire document if the user desires. This is different for images where an entire image has to be displayed and even more so for videos. Furthermore, assigning all searched candidates with relative importance is easier for text documents depending on the size of the substring matched or the number of times the keyword occurs in the document. For media objects, it is not clear how to do this. All these issues can be summarized into a list of requirements to make multimedia database usage viable:

- *Content-based information retrieval and browsing capability*—As mentioned earlier, extracting semantics is not easy for multimedia content. User descriptions and interfaces to place these descriptions are simultaneously used. These descriptions are stored as metadata and used for analysis and indexing.
- *Capability to handle a large number of media objects*—Media objects might not just represent images, video, and audio individually but can also be heterogeneous, containing combinations of one or more media types stored in standard formats or proprietary formats.
- *Cost-effective storage and data management*—A media database might store millions of video, audio, and other media clips each occupying anywhere from a few megabytes to terabytes, whose locations could be distributed over different networks. There should be low-cost storage solutions, transparent access, and retrieval of this media data and overall efficient media data management.
- *Database function support*—A media database should support common functions to maintain databases, such as create, insert, delete, search, and backup of media objects or metaobjects along with any metadata.
- *Media object composition within a database*—Because of the heterogeneous media–type nature of multimedia content, indexing and browsing capabilities should be able to address the media content in whole or in part. For instance, with text document search and browsing, hyperlinks link phrases in the current document to other documents or "sections" in other text documents. With the complex heterogeneous nature of multimedia, such a setup is not as straightforward.
- *Maintenance of media synchronization across queries and access*—Each dynamic multimedia content always has synchronization needs that should be maintained while viewing. In the distributed scenario, access to the media object, in whole or part, or any other nonlinear retrieval and access will need to preserve the synchronization while viewing the results.
- *Error recovery*—Distributed setups are common because of the voluminous nature of multimedia data. During communication, there is always a chance of data packets being lost or dropped, resulting in loss of data or synchronization. Error recovery and resiliency in such cases is needed.

- *Concurrent access, changes, and locking mechanisms*—These are common database issues and have been addressed in the case of text-based databases because text documents are small. However, with the large volume of multimedia, such operations can lead to larger wait times, leading synchronization problems, and others.

One common theme in all of the preceding requirements does stand out—the need for multimedia systems and databases to be distributed and dynamic. This is because of the nature of the multimedia content, its storage requirements, and its heterogeneous nature.

A snapshot of a distributed architecture is shown in Figure 15-2, where media is stored at different networked locations, and queried by users on different networks supporting a variety of end interfaces with different capabilities.

The problems that need to be addressed to make this architecture a practical reality are many, but can be grouped into content-related and distribution-related problems. Content-related problems include extraction of semantics, management of digital rights, authoring or maintaining scalable content that can be viewed on end terminals with different capabilities, linking content and metadata together, and so forth. Those that relate to the distribution aspects are management of QoS and real-time delivery and adaptation of the delivery across heterogeneous networks. Standards for content structure (multimedia and metadata) and standardized protocols for access, distribution, and internetwork exchange are imperative to aid interoperable functionality. We have discussed standards for multimedia content (MPEG-1, MPEG-2, MPEG-4, and so on) and standards for networking (Internet-based UDP, RTP, RTSP, wireless 1G, 2G, 3G, and so on) in the previous parts of the book. Next, we look at existing and proposed standards for exchange and use of metadata.

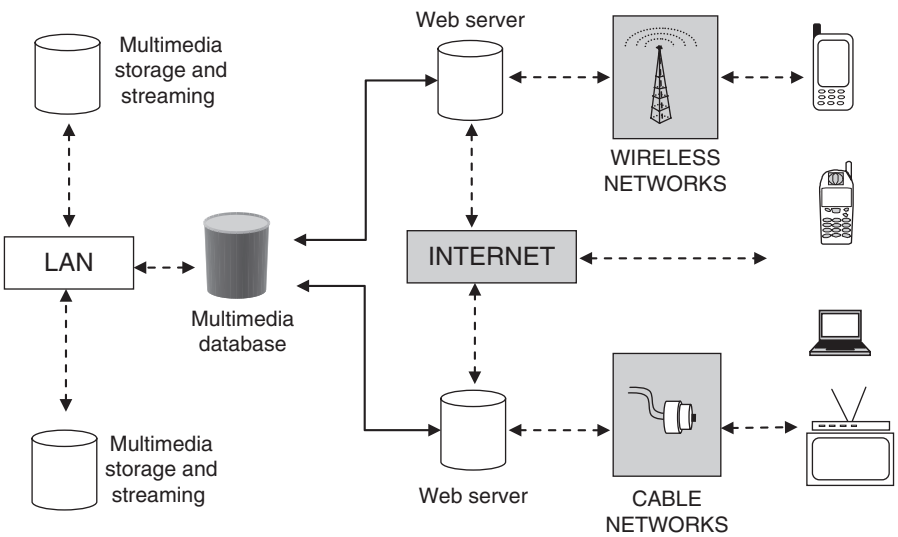


Figure 15-2 Distributed multimedia system. End terminals on different networks need to connect to servers to process multimedia queries on multimedia data that is stored at different locations.

4 STANDARDS FOR METADATA

Standards for metadata help ensure system interoperability for creating multimedia metadata, distributing it, interpreting it, and the overall management of metadata in media applications. Many standards have been proposed to address interoperability of metadata, some of which include the following:

- Descriptive Metadata Scheme-1 (DMS-1) proposed by the Society of Motion Pictures and Television Engineers
- TV-Anytime set forth by the TV-Anytime forum consisting of the BBC, France Telecom, Nokia, Phillips, Thomson, and so on
- MPEG-7 from the ISO MPEG community
- Dublin Core defined by the National Information Standards Organization (NISO)

However, the nonoverlapping purposes for which the standards were created have not harmonized a universal deployment, with MPEG-7 being by far the most diversified to suit multiple platforms.

4.1 MXF and Descriptive Metadata Scheme-1 (DMS-1)

The Material Exchange Format (MXF) was standardized by the Society of Motion Pictures and Television Engineers (SMPTE) to achieve interoperability in the broadcast industry for exchanging audio-visual material. MXF works by providing a standardized wrapper for audiovisual media that is independent of any specific codec. MXF also wraps metadata defined for the audio visual content using a default metadata scheme known as the Descriptive Metadata Scheme-1 (DMS-1). DMS-1 consists of a number of frameworks (DM framework) defined in the following paragraph, where every framework shares the same underlying data model and structure. A better insight of each metadata framework can be gained by understanding the organization of media content in the MXF representation, as shown in Figure 15-3. In the context of MXF, a large content is known as a *production*. For instance, the popular TV series *Seinfeld* is a production. Every production is made up of different episodes. An MXF file might represent an individual episode or a group of episodes. Each episode is represented by a number of clips. Each clip is further broken down into scenes, each of which is characterized by narrative or dramatic coherence. The division of the clip into scenes is an editorial decision, and scenes might overlap or relate to a specific point in the timeline rather than by duration. Each scene can be further broken down into one or more shots, where each shot is an individual atomic unit in the context of a scene.

The MXF file preserves the logical segmentation of content. Metadata input into the MXF file can be used for frames, shots, scenes, clips, episodes, or the entire production and, thus, the MXF file format can have a number of places where metadata needs to be associated. This is done by using *frameworks* that can be inserted into MXF files at appropriate places. A DM framework is a group of related metadata information details, which qualify the contents of an MXF file in a certain context. For instance, frameworks are locations in a separate track in the file if they pertain to the whole file, or at certain

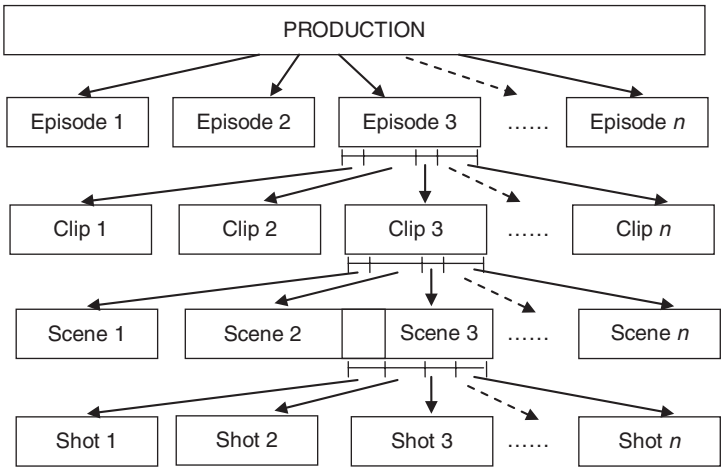


Figure 15-3 Logical demarcation of content. Every large content is known as a production, which is broken down into episodes, and then into clips, scenes, and shots.

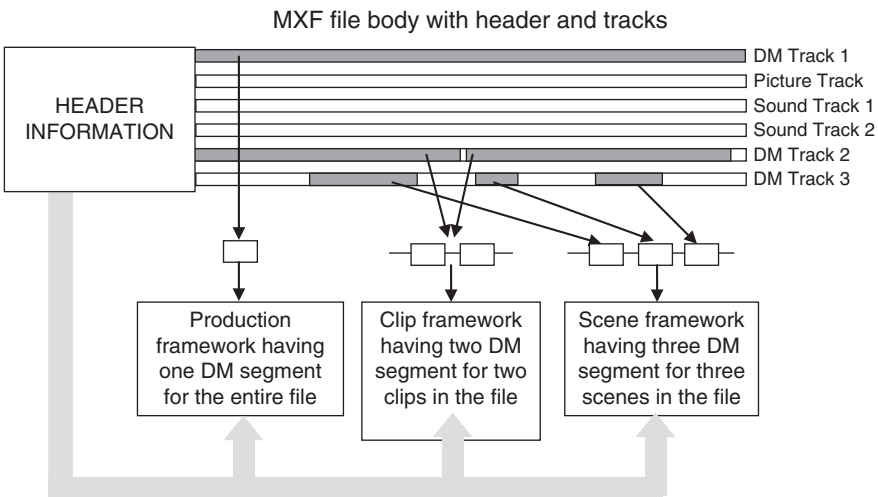


Figure 15-4 Descriptive metadata frameworks. The three types of frameworks (production, clip, and scene) containing segments are shown embedded in different tracks in the MXF file body.

places in the MXF file where they qualify the underlying data. An example is illustrated in Figure 15-4. The DMS-1 standard defines three types of frameworks:

- *Production framework*—The production framework contains descriptive metadata and properties that allow identification of the content and ownership details in the file body. An MXF file that contains wrapped content can only have one wrapped

production framework for the whole file, although the MXF file can contain one or many content programs pieced together. For instance, if an MXF file contains many episodes of a TV show, it will still have only one production framework.

- *Clip framework*—The clip framework contains descriptive metadata and properties that provide the capture and creation information about individual audio visual content in the MXF file.
- *Scene framework*—The scene framework contains descriptive metadata and properties that relate to actions and events within individual scenes of the audiovisual content in the MXF file.

Each framework is implemented by associating values to specific metadata properties, which are defined in a standardized SMPTE metadata dictionary. Some of these properties are titles, identification, group relationship, event, annotation, location, picture format, rights information, actors, and so on, and each is coded using 2-byte tags.

4.2 TV-Anytime

With the growing number of networks and channels that provide television content, it is becoming cumbersome for users to effectively sift through endless channel listings and catalogs of stored information from video-on-demand or pay-per-view services. The TV-Anytime standard has attempted to address this by setting forth specifications for controlled delivery of multimedia content to consumer devices such as digital video recorders (DVRs). The standard was formed in 1999 and aims to provide consumers with personalized TV experience by allowing consumers to store digital metadata tagged TV programs inexpensively on their local recording devices and then providing mechanisms to search and organize all the data to provide a better organized and/or personalized viewing experience. The objectives of the TV-Anytime standard can be summarized as follows:

- Develop specifications including metadata creation and insertion that enable applications to exploit local storage in consumer electronic platforms.
- Be agnostic of the networked delivery of content to the consumer electronics equipment. In other words, the mechanism should work irrespective of how content was delivered, for instance standard airways (ATSC), broadcasting using the ARIB (Association of Radio Industries and Businesses) standard, digital video broadcasting (DVB), digital broadcast satellite (DBS), Internet, and so on.
- Include metadata specifications of security structures to protect the rights of the content authors or distributors.

The TV-Anytime metadata contains descriptive information about TV content and consists of two main parts. The first part, *schemas*, has been adopted from the XML-based MPEG-7 Description Definition Language (DDL) and its metadata representation format. It has a unique structure that includes a program description and a user-based classification scheme (for example, action, drama, commercial). The second part, *systems*, defines a mechanism to encode and deliver schematic descriptions of

the content. The delivery mechanism must be efficient, allow for asynchronized delivery, and have modularized delivery so that any updates can be sent.

The TV-Anytime standard and its metadata specifications were adopted by the European Telecommunications Standards Institute (ETSI) in early 2006 and are currently under consideration to be interoperable with other delivery mechanisms.

4.3 MPEG-7

MPEG-7 was created by the ISO MPEG community to standardize the description of digital media content and required queries to search into these descriptions. Work on MPEG-7 first started in 1998 and the standard was ratified in 2001. It has been developed by experts from the consumer electronics industry, broadcasting and publishing industries, intellectual property rights management industry, telecommunication industry, entertainment industry, along with people from research and academia and, thus, aims to provide a comprehensive and interoperable standard to describe media content shared across the multitude of industries. Formally, it is also known as the Multimedia Content Description Interface. At the heart of MPEG-7 is a standardized XML-based description scheme with a description interface that provides a systematic way for users to describe media information using standardized *descriptors*. This allows for a standardized way to process audiovisual description information in applications that need to access, query, filter, and browse the content, as illustrated in Figure 15-5.

The MPEG-7 standard involves descriptions of content and the processing these descriptions. It consists of many parts that can be broadly classified into four groups:

- *Standardized descriptors*—These can be used to represent content's features. A descriptor defines the syntax as well as the semantics of that feature. A more detailed overview of the types of descriptors provided in MPEG-7 is summarized in the paragraphs that follow. For instance, an image can be

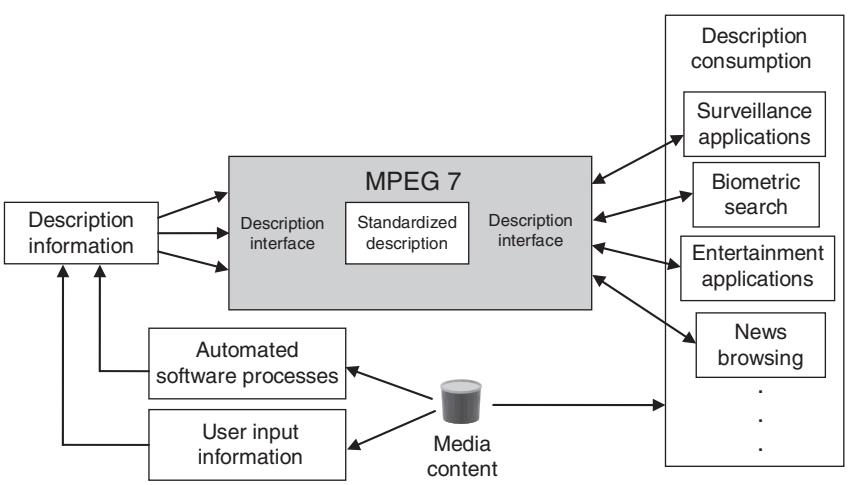


Figure 15-5 MPEG-7 processing architecture

segmented into different areas—each area can be represented by a descriptor specifying the region's shape, color, texture, and so on.

- *Multimedia Description Schemes (MDS)*—All the descriptors defined for the content can have semantic relationships between them, which is provided by a description scheme.
- *Description Definition Language (DDL)*—Although MPEG-7 does provide for many basic types of descriptors, there will always be a need for specific content, or specific contexts and applications, to have newer descriptors. The DDL allows for the specification of new descriptors and description schemes or allow for the modification and extensions of available descriptors.
- *Systems tools*—The systems part of MPEG-7 specifies systems-level protocols to encode MPEG-7 descriptions into binary representations for efficient transport or storage. It also allows multiplexing or descriptions and synchronization of descriptions with content.

The descriptors in MPEG-7 allow creating various kinds of descriptions, which can be formally grouped into three categories: *archival descriptors*, *perceptual descriptors*, and *organization or content access descriptors*. These descriptors are explained in the following list, and examples illustrating their use are shown in Figures 15-6 and 15-7.

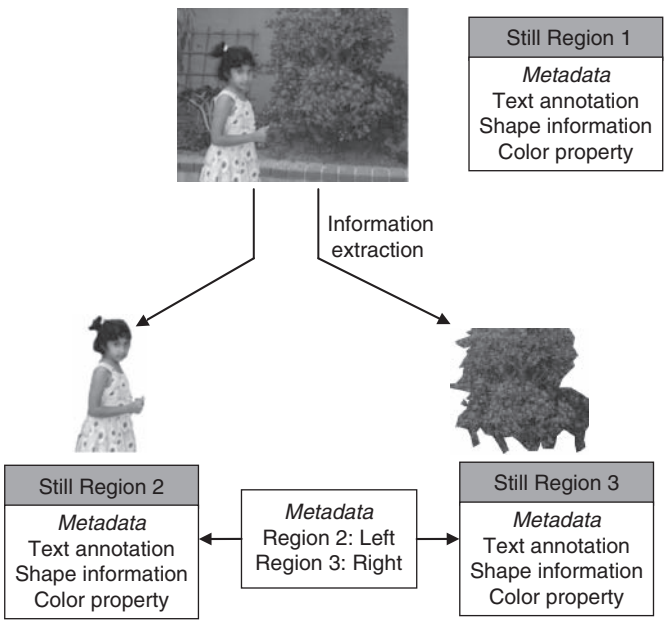


Figure 15-6 MPEG-7 example. The top image is shown broken into two regions. Each region has metadata that describes its properties. In addition, metadata information also depicts the relationships between different regions.

```
<MPEG7>
<ContentDescription xsi:type = "ContentEntityType">
<MultimediaContent xsi:type="ImageType">
<MediaLocator>
    <MediaUri> http://www.mysite.com/picnicPhoto1.jpg </MediaUri>
</MediaLocator>
<Region1 id="R1">
    <TextAnnotation>
        <FreeTextAnnotation> Photograph of girl walking in garden </FreeTextAnnotation>
    </TextAnnotation>
    <SpatialDecomposition overlap="false" gap="true">
    <Region2 id="R2">
        <TextAnnotation>
            <FreeTextAnnotation> Girl </FreeTextAnnotation>
        </TextAnnotation>
        <Relation xsi:type="SpatialSegmentRelationType" name="left">
        <VisualDescriptor xsi:type="ContourShapeType"> … </VisualDescriptor>
    </Region2>
    <Region3 id="R3">
        <TextAnnotation>
            <FreeTextAnnotation> Garden Bush </FreeTextAnnotation>
            <RelatedMaterial>
                <MediaLocator>
                    <MediaUri> http://www.californiaplants.com/gardenplant.html </MediaUri>
                </MediaLocator>
            </RelatedMaterial>
        </TextAnnotation>
        <VisualDescriptor xsi:type="ContourShapeType"> … </VisualDescriptor>
        <VisualDescriptor xsi:type="ColorType"> … </VisualDescriptor>
    </Region3>
    </SpatialDecomposition>
</Region1>
</MultimediaContent>
</Description>
</MPEG7>
```

Figure 15-7 A textual XML-based description in MPEG-7 for the region-based metadata information extracted in the previous example

- *Archival descriptors* include information that pertains to the content's author, date of production, date of expiry, and production process. It also includes descriptions of content usage, such as broadcast schedules, copyright management, and other information relating to the storage, encoding, and transport-related formats.

- *Perceptual descriptors* relate to the content's spatial and temporal organization (scene cuts, segmented region boundaries, tracked regions, and so on) as well as low-level information in the media signal (color, texture, sound levels, frequencies, and so on). It also includes information that relates to higher-level semantics of the content, such as interactions between objects, events, and so on.
- *Organization and content access descriptors* include information to support efficient browsing of content (summaries, translations, and so on) about how users should or have interacted with the content (preferences, history).

4.4 Dublin Core

The Dublin Core standard was started to standardize metadata descriptions for information resources on the Internet. The standard has defined the Dublin Core Metadata Element Set (DCMES), which forms the basis of Web metadata. DCMES consists of 15 elements that can be optionally used, repeatedly if necessary, to form larger descriptive blocks of metadata. Each element that describes a resource can further be specialized in a standardized manner with the help of qualifiers, which are often used to further define the semantics of the element describing the resource. The 15 elements are grouped into three categories, as described in the following list. In every element's case, a single-word label is specified as a tag.

- *Instantiation elements*—These elements include metadata used to qualify the origin of the resource and include identifier, date, format, and language.
- *Intellectual property elements*—These elements relate to information used to manage the digital rights of the resource and include contributor, creator, publisher, and rights.
- *Content elements*—These elements are used to better describe the content. They are composed of coverage, description, type, relation, source, subject, and title.

The metadata can be represented in many different syntaxes, including HTML, XML, and RDF (Resource Description Framework). HTML has two tags that can be used to embed metadata. These are *<meta>* and *<link>*. The position where these tags should be placed has also been standardized—being after the *<head>* and before the *</head>* tags in the HTML document. A sample usage is shown in Figure 15-8. Most Web browsers and search engines follow the standard to analyze the *<meta>* and *<link>* tags appropriately, that is to ignore them when the document is displayed in the browser, printed, or specifically look for it when performing a search.

4.5 IPTC Standards

The International Press Telecommunications Council (IPTC) was established in 1965 by a group of news organizations with the purpose of developing technologies and standards to improve news exchange and news content creation in various mediums—print, radio, television, and now the Internet and other wired and wireless

```
<html xmlns="http://www.thomson.com/solutions/learning/multimedia_book.html">
    <head>
        <meta name="DC.creator" content="Parag Havaldar" />
        <meta name="DC.creator" content="Gerard Medioni" />
        <meta name="DC.subject" content="Media" />
        <meta name="DC.description.abstract"
            content="Multimedia is a broad 'umbrella' that innovatively combines different fields of
research and industry to produce practical solutions that are used on a wide scale today. These fields
range from signal processing, imaging and color science, video and audio analysis, 2D/3D graphics,
information theory, networking, databases, watermarking, encryption, etc. Research in each field is
progressing, resulting in new applications that use the different media types in new ways. Although
books exist that deal with multimedia systems, most of them (to the best knowledge of the authors)
have been rather weighted toward either the networking or compression aspects and target only
specific areas and student audiences. There is no comprehensive textbook that puts all these
concepts coherently together explaining each area sufficiently enough and yet addressing the
problems, technologies, solutions, and standards of this ever-evolving field. This book intends to serve
that purpose for the advanced level undergraduate curriculum and entry-level graduate curriculum " />
        <title>Multimedia Systems – Algorithms, Standards and Industry Practices</title>
        <meta name="DC.Date.started"    content="2005-04-15" />
        <meta http-equiv="Content-Type"    content="text/html; charset=iso-8859-1" />
        <link rel="stylesheet" href="/css/default.css"    type="text/css" />
        <link rel="meta" href="index.shtml.rdf" />
    </head>
    <body>
        <H1> Chapter 1 </H1>
        …
        …
    </body>
</html>
```

Figure 15-8 Dublin Core metadata description example in HTML. HTML documents contain tags that have identities, data, and formatting specifications. The Dublin Core standard has introduced the "meta" tag with its "content" field to point to metadata.

network mediums. It consists of about 55 companies and has solely focused on developing and publishing industry standards for the interchange of news data. Over the last few years with the standardization of XML, IPTC has been putting forth versions of XML standards for news content creation, aggregation, sharing, advertising, and distribution to multiple media. Some of these IPTC standards include the following:

- *NewsML*—NewsML is intended to provide a media-independent structural framework for multimedia news.
- *SportsML*—SportsML is the only global extensible standard for interchanging sports data among a wide network of news studios. Sports data includes sports

scores, information about matches, sports schedules, individual/team rankings, and statistics for a wide variety of sports competitions.

- *ProgramGuideML*—ProgramGuideML is designed to standardize the interchange of radio/TV program information.
- *EventsML*—EventsML is another standard by the IPTC intended for exchanging information about event publishing, event planning, and event coverage.

5 USER INTERFACE AND BROWSING PARADIGMS

Semantic information is recovered from media and stored as metadata information along with the media file in a standard such as MPEG-7. This information is then used for indexing, sorting, searching, browsing, and other multimedia organizational processes. There are two issues here: that of efficiently creating, presenting, and editing (if needed) any semantics for a multimedia presentation and, second, presenting the results of these organizational processes in applications in a way that depicts the extracted semantics along with the accompanying media signals in a form that is concise, easy to understand and interact with, while at the same time aesthetically pleasing. A few browsing and presentation paradigms are discussed in this section.

5.1 Presentation of Semantics

Semantics include a variety of concepts that help understand the entities in a media signal and their relationships, such as color, shape, structure, spatial relationships, and temporal structure. A commonly used structure to present semantic information is in the form of a table of contents. Table of content paradigms may range from a simple sequence list in the case of a text document like an e-book, to complex ones that make use of a spatiotemporal list of events in a video. An example of such a table of contents for a soccer game is shown in Figure 15-9. Here, the video clip of a soccer game has been analyzed and semantic information has been inserted manually. A user can view the spatiotemporal map and edit or insert new information if necessary. All the information can be saved in a standard format such as MPEG-7.

5.2 Organization of Results

Any process used to organize data, such as indexing, searching, or browsing, produces results that need to be presented to the user in an appropriate manner, so that the most relevant results show up first. Processing queries and presenting results has already been effectively used on text documents. Commercial text databases as well as many search engines exist, which search and catalog text documents. You can easily see the complexity of information that needs to be presented when you perform a search on a popular Web-based search engine such as Google. Google ranks results based on its Page Rank method, where all results are ordered depending on the number of documents that reference it. Presenting other media type searches, such as

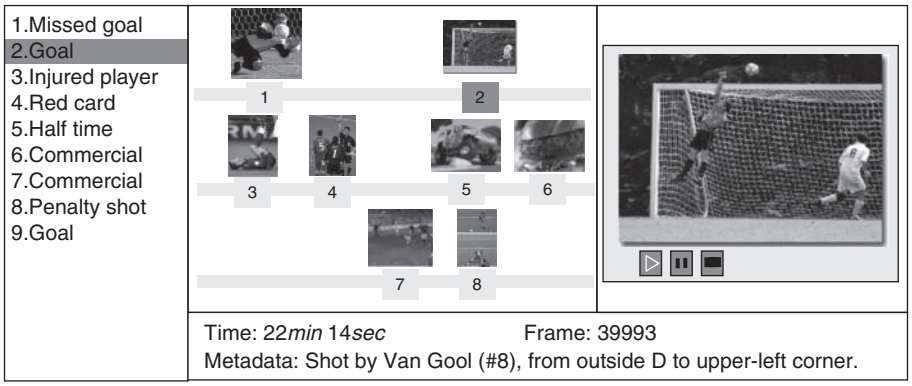


Figure 15-9 A sample interface for presenting and editing metadata for video. The left column shows keys used to describe a metadata insertion point. The middle band shows a visual timeline of the video with appropriate frames where metadata has been inserted. The bottom editing area lets the user input any descriptive string for that frame or group of frames.

images, video, and audio, in a useful way to sift through and analyze is not necessarily as straightforward as text primarily because of their large size and temporal nature. For instance, with commercially available megapixel cameras, producing large-sized images is easy. In such cases, displaying the results of an image query that might produce hundreds of images on limited screen space in a way that the user can take a quick look and sort through the desirable ones can be a challenge, more so for video. One common way to present searched results that involve images and video is to show smaller, "thumbnail" representations of the images and videos with specific temporal frames, as shown in Figure 15-10.

6 EXAMPLES OF MEDIA DATABASE PROJECTS

Any multimedia database systems must support accessing queries on media types in addition to traditional data management functions, such as data organization, creation, storage, retrieval, backup, and so on. Many media databases have been implemented at research institutions and even commercially though with limited success. These can be categorized into different classes depending on their invention time and functionality provided.

6.1 The Early 1990s

This time period saw the first class of implementations of multimedia databases. It could be said that these first attempts were not really database systems, but applications that relied on operating system interfaces and functionality to search for files or look for strings in files to process queries. A typical example of this is multimedia CD-ROM contents that represented catalogs of images, clip art, maps, and even

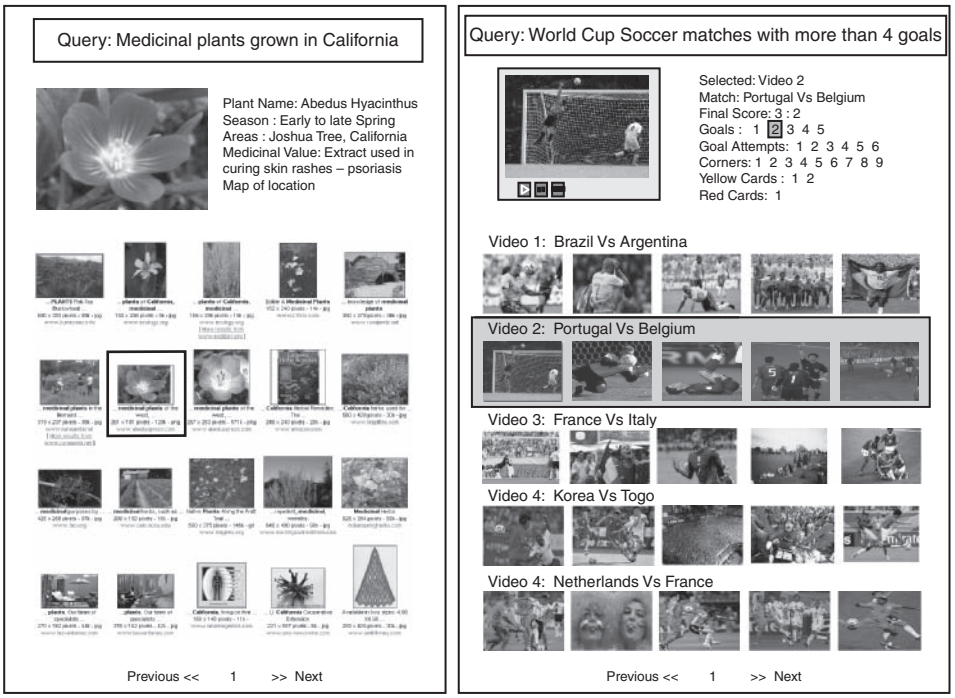


Figure 15-10 User interfaces to browse multimedia queries. The left image shows a query on an image database. The query results in thumbnail images, which, when selected, show a higher-resolution image with associated metadata. The right image shows a query to find all soccer world cup videos with more than 4 goals. Each video in the result takes one line with representative frames. Selecting a video displays the video with accompanying metadata, which is also hyperlinked to the video frames. For instance, selecting goal 2 sets the video to the appropriate frame.

encyclopedias. Being the first collections of media content, it was assumed that they represented a *database* in the form of collections of files or records with special indexing and searching mechanisms implemented by operation system functionalities. However, there was never a formal representational language or query processing language for these database representations. Examples of these include ImageAXS, which was primarily used for images. The first general-purpose multimedia database was MediaWay (or MediaDB), which provided very specific queries but on different media types and formats, such as GIF, QuickTime, BMP, PowerPoint, and so on. MediaWay was one of the more successful multimedia databases that provided a mix of standard database technology (concurrency, archiving, access, storage, retrieval) along with a specific set of multimedia-related functionality. Another successful database that falls into this category of early multimedia databases was Jasmine, developed by Computer Associates. Jasmine, though not purely a multimedia database, assembled text information along with any pertaining video, audio, and graphical

animations that could also be the result of a query. For instance, a manual on airplane manufacturing could be turned into a database so that you could easily access the text that pertains to assembling the wing of the airplane along with any accompanying images, video, or animations that showed how this was done. Jasmine, therefore, did deal with multimedia information, but did not process queries on multimedia. Other similar products of the time are ITASCA, POET, and Gemstone.

6.2 Turn of the 2000 Century

The multimedia database attempts in this group can be further classified into two: those that extend the working of a normal relational database to involve more media types and those that favor multimedia content itself, where queries work by extracting low-level semantic information either from the media object itself or from the text that surrounds the media object. For example, an HTML page might contain images with surrounding text that describes the image.

Around the late 1990s, database systems started managing complex objects rather than simple data. The object-oriented style of database management was extended to defining new data types and operators that were appropriate for the new kinds of media, such as images, video, and audio. The most useful multimedia databases that extended the object-oriented technology were Oracle 10g, IBM Informix, and IBM DB2. These databases include image, video, audio, or just about any object with definitions to access, store, manipulate, and index in a common relational framework. The multimedia features implemented include importing, exporting, access control, a browsing interface, and so forth.

Extracting information from media objects for database indexing purposes was first implemented in the Query by Image Content (QBIC) database developed at the IMB Almaden Research Center. QBIC used color histograms as a basis for comparison with index images from the database. Recall that a color histogram is a three-dimensional array that measures the frequency of each color in the image. Along with color histograms, QBIC also uses other attributes such as textual annotations, captions, and so on to help media indexing. Other similar image database attempts include the Berkeley Digital Library Project, Content Based Visual Query (CBVQ), and VisualSEEk at Columbia University.

6.3 Current Status

Current efforts at multimedia databases now address the desire for richer semantic content. Most of them rely on standards such as MPEG-7 or even MPEG-21 (see Chapter 16). A representative project based on MPEG-7 is the Multimedia Analysis and Retrieval System (MARS) carried out at the University of Illinois at Urbana Champaign and the Multimedia Data Cartridge (MDC).

MARS is an integrated multimedia information database management and retrieval system that considers multimedia information as the main objects that can be stored, retrieved, queried, sorted, and so on based on their semantic content. The multimedia data model used in MARS for content indexing, retrieval, and data

management influenced the development of MPEG-7. Some of the features used to extract semantic content are a table of contents extraction (ToC) for videos, a hybrid tree structure to support feature indexing across different media objects.

MDC is an extension to the Oracle database management system and provides a multimedia query language used to represent, process, and index media objects. The indexing capabilities rely on schemas derived from MPEG-7. The modular architecture of the Oracle databases allows for extending its services by implementing a Data Cartridge, which serves to extend the interface for user-defined data types by providing the normal operations of inserting, deleting, indexing, and so on. Thus, MDC extends database management to handle multimedia objects using Oracle's Data Cartridge technology. MDC consists of two parts: a Multimedia Data Model, which consists of metadata describing the multimedia content. In this case, the MPEG-7 MDS schema is mapped onto Oracle's database schema using the extensible type systems of the cartridge environment. The second part is the Multimedia Index Type, which provides an indexing environment for multimedia retrieval.

7 EXERCISES

1. **[R03]** The text has stressed the need for semantic queries on multimedia content. Assume that your multimedia content consists only of images, only of video, or only of audio. In each case, give examples of queries that meet the following criteria:
   - Useful and can be done automatically today.
   - Useful but cannot be done automatically and will need user input metadata. Give examples of the metadata descriptions in this case.
2. **[R03]** Although metadata is largely viewed as a much-wanted feature for indexing multimedia information and other tasks, it is not very easy to set up and use. Mention some of the problems that you see with practical applications that need to make use of metadata.
3. **[R04]** Metadata is used to tag media data so that it can used by applications for organizational needs of the media. This question tests your understanding of what *metadata* is.
   - Define metadata. How is it different from data?
   - Normally, most standards specify metadata by explicitly tagging it as metadata using keywords or tags. This definitely disambiguates the metadata from the actual raw data. Let us say you had no tags to do such a differentiation; how would you make out what the data is and what the metadata is?
   - Do you think you can always unambiguously differentiate between raw data and metadata? Give reasons and examples to qualify your answer.
   - Because metadata is some descriptive information about data, you could define meta-metadata as descriptive information about metadata, and so on. Would you want to standardize such recursive description models? Do you see any use for this?

4. **[R04]** Metadata is data that describes data. Going by that definition, one of the earliest standards associated with describing video content was closed captioning in digital television. Although we have not specifically mentioned closed captioning in the text, answer the following:
   - Research what closed captioning is and how it works. Can you term it as a metadata standard? Why or why not?
   - Can closed captioning be used for indexing or organizing media information? Do you know of any commercial programs that use closed captioning for indexing?
   - Explain some problems you see with the way closed captioning works today.
5. **[R05]** Metadata is normally used to describe media objects. Can it be used to compress media objects? If not, explain why not, giving examples, or mention how it can aid compression. Along the same lines, how (or if) can metadata be used to efficiently control delivery across variable bandwidths?
6. **[R04]** When you use search engines to look for information, it is presented to you in a rank-ordered manner, with the most relevant document occurring first. For instance, Google uses the Page Rank method to order documents.
   - How does Page Rank work?
   - Does it work on text documents only, or can it work on other media type documents, such as images, audio, or video?
   - The Page Rank method can be used when documents are hyperlinked. In the absence of hyperlinks, how can you solve this problem?
7. **[R05]** When comparing content such as images, video, and audio, it is important to understand what the content is about. However, extracting such semantics is difficult and generally an ill-posed problem.
   - Why is extracting semantics from media difficult?
   - One of the options mentioned in the chapter for querying media databases is "query by example." Define this term. Give specific examples of how this would work for images, audio, and video.
   - Query by example has proven to be a very useful paradigm in specific well-defined cases. Some of these include databases of images of faces, fingerprints, and cloth textures. Reason about how you might implement indexing in such cases.
8. **[R06]** When it comes to searching and indexing multimedia documents, using metadata to search a database of multimedia documents will result in a number of media documents to be sifted through. Although a standard paradigm has evolved for sifting through text documents, presenting all information in a relevant way is not always easy.
   - The text explains how to preview search result for images—by using thumbnail images, which are small and allow you to see hundreds of images at once. Each thumbnail is hyperlinked to its real-sized image. However, using thumbnails can also be cumbersome. Can you think of a better way?

- What about video? If a search query using metadata on a video database comes back with hundreds of videos, how will you rank them and show them effectively to the user?
- Can you show an effective user interface for browsing hundreds of video elements?

9. **[R05]** Dublin Core is primarily intended for metadata descriptions for Internet resources and is currently used by various Internet browsers and search engines.
   - Explain some of the advantages you see with Dublin Core technology.
   - Are there any disadvantages? Give some specific examples that will be hard to perform in applications that make use of the Dublin Core metadata.
   - Because metadata is defined in a textual format, the metadata information will be saved in a language, for example English. What potential problems do you see with the internationalization of the content?

10. **[R05]** Consider a large-scale distributed multimedia system where many resources are used to send multimedia information from a sender to the receiver. We have also seen networking protocols in the previous section that can be used to set up and reserve pathways. In any dynamic setup, changes to resource availability and its capability make it possible to affect QoS and, ultimately, the presentation experience at the receiver end.
    - First enumerate what problems you see in dynamic networks that provide content to end terminals with varying capabilities.
    - How can you use metadata effectively to solve this problem? Give examples arguing for the advantages/shortcomings.

11. **[R06]** Suppose you have an MPEG-4 presentation in the format of an MP4 file. The presentation contains a few video streams, audio streams, and images. Now you want to insert metadata information for each media stream in the MPEG-4 presentation. The metadata is to be in the MPEG-7 format.
    - Outline the broad steps that you would take to do this.
    - Will the final resulting file be an MPEG-4 file or an MPEG-7 file?
    - When streaming the content with the metadata, what problems could potentially come up? Does MPEG-7 or MPEG-4 address these?

12. **[R08]** This question is along the lines of browsing searched information. Suppose a query on a movie database returns many videos that you want to sift through to see which one is worth watching all the way through. One way to do this is to automatically create "trailers" of the movie to be shown to you.
    - Of course, every movie could have a premade trailer that could be shown to you. However, suppose there is no trailer, and you want to automatically create one. How would you accomplish this? What qualitative steps will you take to extract interesting sections from the video and show them?
    - Creating a trailer on the fly takes more time than creating it once and storing it; however, can you see any advantages to creating it on the fly?
    - Can you extend your technique to different domains for videos, for instance, when you are searching a video database of sports matches?

CHAPTER 16

Multimedia Frameworks

The chapters in the book so far have described various aspects of multimedia from representations and algorithms to standards. Each multimedia standard is designed for specific applications, and some have come to become an integral part of our everyday experience to access information and communication. The content designed for each standard also has different properties and uses different networks for distribution. As a result, a variety of networks and standards have come into operation. This chapter focuses on the way content can be shared across multiple networks in a unified manner. Section 1 explains the need for a unified multimedia framework. Section 2 introduces the objectives of MPEG-21 where content is encapsulated and expressed in individual units called Digital Items. Section 3 describes the Digital Items and Section 4 illustrates the identification of Digital Items that is necessary for further processing of Digital Items. Section 5 illustrates the issues involved in adapting Digital Items across different networks, end terminals and Section 6 shows how Digital Items are processed or consumed by the end terminal. Finally, Section 7 mentions the Rights Expression Language that is used to perform digital rights management on the Digital Items.

1 THE NEED FOR A UNIFIED FRAMEWORK

Many multimedia standards have been created to satisfy requirements of various markets or sectors. This has resulted in many different open standard and proprietary bit streams of information for exchange and distribution, each requiring specialized hardware and software. For instance, the MPEG-1 standard was specifically created for disc distribution of audiovisual content, whereas MPEG-2 was created for digital

television. Other standards such as 1G, 2G, and 3G address multimedia-related distribution explicitly for the wireless markets. The MP3 standard (MPEG-1 or 2, Layer 3) is used to compress and exchange digital musical audio over the Internet and is now also used in increasing capacity to store music and stream to PDAs and iPods. Recently, MPEG-4 has standardized audiovisual solutions, where users can interact with the content. MPEG-4 has created new opportunities for users to create, access, consume, and even repurpose media content. In addition, a variety of distribution protocols were developed specifically for Internet-based distribution (HTTP, RTP/RTCP, and so on) or MPEG-2 TS for transport over cable networks. The Digital Cinema pipeline, which is currently being standardized and setup, will allow for digitally scanned and rendered movie frames to be distributed to theaters, where they will be digitally projected, thus making it no longer necessary to print and physically distribute film. Additionally, this content will also be distributed securely in all cases by implementing Digital Rights Management solutions.

Thus, the software processes or hardware devices that create these content bit streams and the end terminal devices to consume these bit streams are all different—as are the distribution channels for that sector and the different consumer centric solutions and business models that have been developed for the manner in which media data is created, communicated, transacted, and consumed in that specific marketplace. This has led to the creation of mature multimedia technologies with numerous players in the content creation, delivery, and content consumption chain that provide an array of media information services and solutions. For instance, the digital television sector has broadcast networks, set-top boxes and televisions, and services to view (overview, recording, specific channel viewing, pay-per-view, and so on). Correspondingly, the wireless sector has a parallel set of infrastructure elements for content creation, distribution, and consumption for services such as telephone music, radio, Short Message Service (SMS), and so on. Similar services are offered on the Internet. This can be viewed as a variety of parallel standards and competitive solutions in an era where different markets and networks are crossing over in the desire to provide consumers with universal media access. No single solution exists that allows all the different communities in various markets—content creation, communication, consumer electronics, financial, and so on—to interact efficiently using the already-created infrastructures. For example, cable television and satellite television networks have been designed to provide audiovisual content directly to the home. However, as wireless bandwidth increases, a need arises to route cable television information to your mobile device and vice versa. Figure 16-1 illustrates an example of such use cases.

There is, therefore, a need to develop a common multimedia framework that would facilitate exchange of media content and consumer services across all these media sectors. It could be said that some sectors have already been trying to put solutions into place to achieve this. For instance, wireless users can use their mobile devices to access the Internet, although with reduced capabilities. A more efficient standard is needed that is aimed at the integration and exchange of the different needs of each media sector and its consumers, resulting in a possible *universal* enhanced user experience.

The problems that need to be overcome to accomplish the universal interchange mainly stem from two orthogonal but conflicting requirements—interoperability and Digital Rights Management. Each standard aims at interoperability, resulting in the

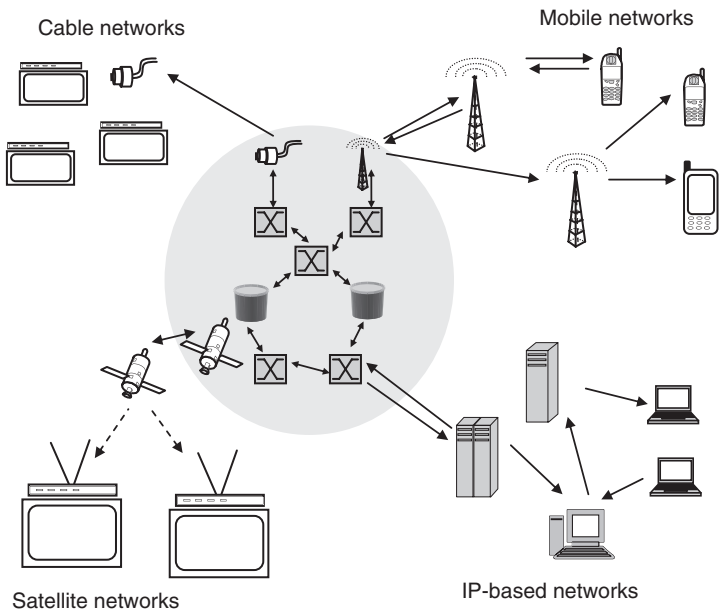


Figure 16-1 Distributed content communication across multiple networks. Each network has its own distribution protocols and content bit stream specifications. If such content streams are to be shared between multiple types of networks, a content distributed–based framework needs to be established.

design of a variety of software and hardware devices to create and consume a standardized bit stream, be it for MPEG-1 (VCD players), MPEG-2 (set-top boxes, cable TV boxes, DVD players), or just about any standard. Moving toward an open standard across sectors, the idea of interoperability has to be made even more flexible and accessible. However, the aim of interoperability, which makes the bit stream syntax freely available, can violate the requirement to protect the value of content and the interests of the digital rights holders. Each standard might have its own Digital Rights Management (DRM) process in place, which normally works by encrypting the content in a certain manner and associating a set of usage rules designed to control the distribution and use of the content. Examples of such specific DRM implementations can be seen in the current online music distribution services, such as iTunes from Apple, Rhapsody from Real Networks, Vodafone Live, and so on. However, songs purchased using the Rhapsody service cannot be played on an Apple's iPod. This is true of other markets as well, such as Digital Cinema, DVD distribution, and cable TV. The need to develop an open media exchange standard across different markets requires the definition and integration of a unified DRM system.

The MPEG-21 standard aims to achieve such universal exchange by describing any type of content in a standard manner, such that intermediary parties, devices, and software and hardware processes will know how to interpret it and, therefore, how to take action on it, if any, or if allowed. The next few sections describe the broad objectives and specific parts of MPEG-21 that allow such universal exchange of content.

2 MPEG-21 OBJECTIVES

MPEG-21 provides an open framework that can enable multimedia information exchange and consumption among various players in the content creation, delivery, and consumption chain. The open framework allows all players in all sectors equal grounds for information exchange and management, along with a schematic way to express Digital Rights Management issues. MPEG-21 achieves this universal exchange using a few key concepts:

- Describing all content in a standardized manner in way that is scalable via standardized descriptors. Such a description, termed a *Digital Item*, is the fundamental unit for exchange.
- Allowing a Digital Item to be *adapted* for interpretation on a terminal. This entails knowledge of the Digital Item's requirements and the capabilities of the terminal and/or network.
- Consuming or *processing* the Digital Item so users can interact with the terminal to consume the Digital Item.

The goal of MPEG-21 can, therefore, be described as formulating a technological framework needed to support users to exchange, trade, access, and view Digital Items. The representation of content in the form of Digital Items allows the exchange or manipulation of the description in an interoperable manner across different terminals. This is performed by adapting the description of the Digital Item to the needs of each terminal. Figure 16-2 shows an overview diagram of MPEG-21's pipeline.

The MPEG-21 standard is organized into different independent sections that provide mechanisms to support multimedia description, exchange, and delivery. The design of all these sections is based around the central MPEG-21 paradigm of defining the media content as a Digital Item. The MPEG-21 sections mainly include the following:

- *Digital Item Declaration (DID)*—The media content is described as a Digital Item. The Digital Item Declaration provides a uniform and extensible abstraction for the content, along with an interoperable schema for declaring Digital Items.
- *Digital Item Identification (DII)*—The Digital Item Identification (DII) defines the architecture for identifying uniquely any digital entity, regardless of its media nature, type, or granular composition.
- *Intellectual Property Management and Protection (IPMP)*—The Intellectual Property Management and Protection section provides the means to reliably protect and manage content across networks, sectors, and different devices along the way.
- *Rights Data Dictionary (RDD)*—The Rights Data Dictionary specifies a list of terms that are used to describe the media owner's rights as well as a user's access rights.
- *Rights Expression Language (REL)*—The Rights Expression Language consists of an expression language that makes use of the terms described in the Rights Data

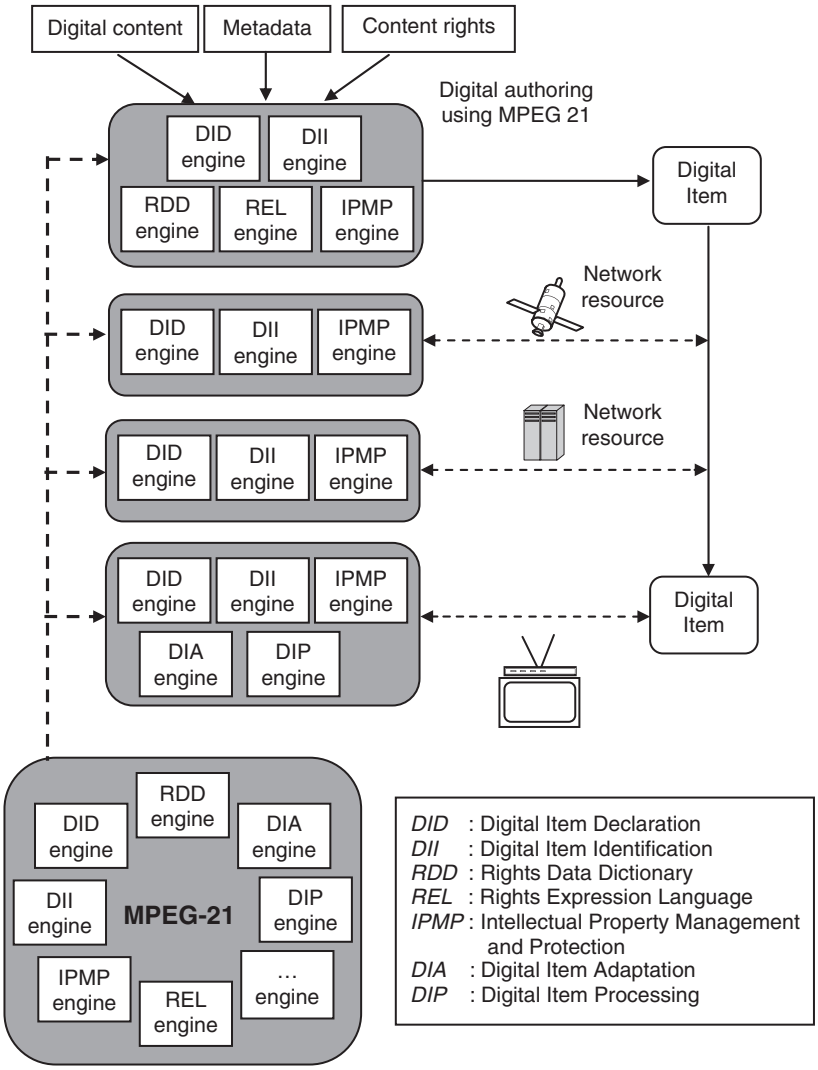


Figure 16-2 MPEG-21 elements shown consist of processes to describe Digital Items, identify them, process rights information regarding Digital Items, and, finally, adapt them for Digital Item Processing on various networks and end terminals. All or part of these elements are used in the creation and processing of content as it makes its way from a source to an end terminal.

Dictionary. The expressions are machine readable and convey the semantics of access usage in the media transaction chain.

- *Digital Item Adaptation (DIA)*—The Digital Item Adaptation (DIA) defines description tools that allow the multimedia content to be adapted for transparent access and interchange.

- *Digital Item Processing (DIP)*—Digital Item Processing defines mechanisms for interoperable processing of the media information in Digital Items in a standardized manner. Digital Items are processed on a platform after being adapted.
- *File format*—The file format describes a format for storing and distributing Digital Items.

Each section can be used independently, or in combination with different sections, as illustrated in Figure 16-2. This provides maximum flexibility in usage and lets intermediary media transaction processes implement them outside MPEG-21 as a whole in conjunction with their own proprietary technologies. For instance, although MPEG-2 was initially designed for digital television, only MPEG-2 video and not MPEG-2 audio is used for the U.S. digital TV representation, with MPEG-2 systems providing support for distribution. Though the various parts of MPEG-21 have been developed and can be used independently, optimal results are obtained when used together.

3 DIGITAL ITEMS

Because the Digital Item is the main structured digital object in MPEG-21 that describes digital content to support interoperable interchange, manipulation, and consumption in the media chain, it is necessary to describe precisely what a Digital Item entails and how it describes the underlying content. Digital content consists of a combination of one or more media types that have spatial and temporal properties, representation standards that hold the bit streams (such as MPEG-2 for video, JPEG for images, and so on), along with all the additional metadata needed in the description of the content. To allow uniform access and exchange of such media, there needs to be a unified way to describe the content. That is, any terminal accessing the content should know what it consists of, what has been used to encode the bit stream(s) in it, any metadata descriptions, rights-related information, and so on, so that the terminal can decide how to parse it for its own consumption, whether to disregard it, and even adapt it toward its own capabilities. In that regard, Digital Items can be defined as structured digital objects that have a standard representation and identification with the incorporated metadata, which overall represents the fundamental unit of transaction and distribution within the MPEG-21 standard.

There are clearly an infinite number of types and ways to create content and many ways to describe it, depending on what it is meant for, how it is meant to be interacted with, and other circumstances that reflect its context of use. Thus, outlining formal ways to describe content that can be flexibly modeled in a Digital Item can prove difficult. Also, the content is not necessarily static, and can morph into a different form; for instance, if it is interacted with, the content can change. Hence, the description has to additionally describe the content's behavior. Any description model that can be used to describe a Digital Item can be useful in a framework only when the description is unambiguous and can be understood uniquely, such as when it is interchanged and transacted in the content supply and consumption chain. To illustrate further, let us consider simple content first, such as an image with text that describes the image. This

content is uniquely described by all the pixels in the image and the text string that qualifies it. It is static and cannot be interacted with; hence, it can be wholly described as a unit of exchange. A more complex example is an HTML document that describes a news report. The document contains text, images, and possibly an embedded video, some of which might be included as hyperlinks. In the simplest of cases, this content can be described uniquely by the document itself, along with all the resources that it uses. If this document were to be communicated as a Digital Item, including the document with all the hyperlinked resources, images, video, text, and layout information would suffice. However, this example assumes that the hyperlinked resources are static and not changing. If new images became available at the hyperlinked location, the content is no longer the same. Additionally, if the document contains scripted logic that is used to change content depending on who is viewing it, then that complicates the description further. For instance, there might be an advertisement that shows up depending on a query executed in the script that takes the viewer's identity (for example, network ID or processor ID) into account. In such cases, a single snapshot including all the dependent links is not sufficient to unambiguously describe the content.

The example in Figure 16-3 highlights a few problems inherent to describing content. The first problem is that of determining all the dependencies, which in itself can be searched for with relative ease. The second problem stems from the inclusion of run-time logic that changes the content—from a static hyperlink to a link that can be determined only at run time. Methodically describing such content as a Digital Item can still be achieved, if instead of describing one document with a dynamic run-time link, you can create many Digital Items, each of which describes the content as one instance of the run-time link. Although the description has been unambiguously

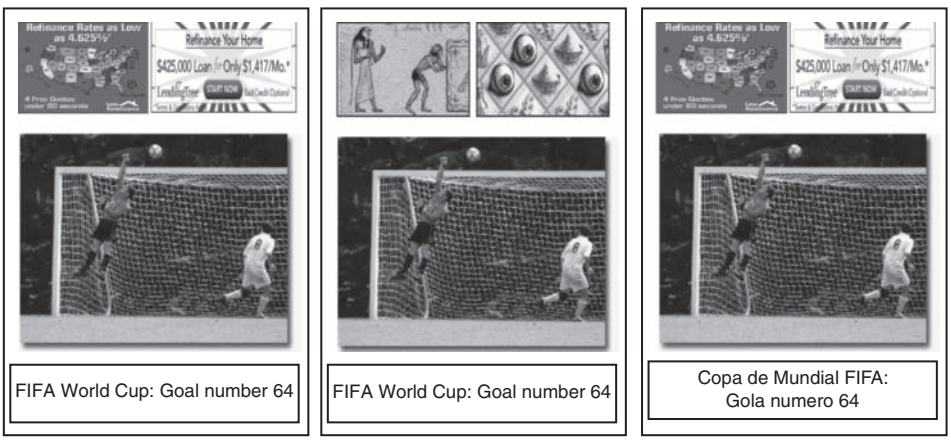


Figure 16-3 Digital Item description interpreted in different ways. Here, the Digital Item is a page with content that has video and images. Images represent advertisements that are indirectly referenced via hyperlinks and change depending on time and user identity. The left image displays one instance of the Digital Item. The middle image displays another instance where the indirect references point to different content. The right image shows that the language of the description has changed based on the user's identity.

achieved by creating multiple Digital Items, it is based on the assumption that creating multiple Digital Items was the author's intent. And this should illustrate yet another problem—whether the author's intent was to create multiple Digital Items or to create a single Digital Item with the advertisement unresolved. Not being able to decipher the author's intent makes it impossible to infer the exact resources that the Digital Item(s) should consist of. All these description-related issues are addressed by the Digital Item Declaration (DID). A Digital Item Declaration methodically describes the structure, organization, and overall architecture of the Digital Item.

3.1 Digital Item Declaration (DID)

The Digital Item Declaration specifies a set of standardized terms that are used to describe a Digital Item. Overall, MPEG-21 has put forth an abstract and extensible model to describe content in a document form. This model is realized by translating the model into a representation using interoperable schemas based on XML. These can be used to describe the content, query the description, and even change the description based on user actions. The abstract model and it schema-based representation is elaborated in the following sections.

3.1.1 DID Abstract Model

The Digital Item Declaration model describes a set of abstract terms along with the necessary concepts to describe a Digital Item. Within this model, the Digital Item can be considered as a digital representation of some "content" that is then acted upon or interacted with by other programmatic or semiprogrammatic processes. For instance, the representation needs to be managed, its definition needs to be extended, transacted, or exchanged, and so on. The model is designed to be a flexible description that can be molded and extended as desired, at the same time providing the necessary links to extend a higher level of functionality. This allows the abstract model to be typified to describe specific types of objects within the content, for instance interactive elements, dependencies between resources, Intellectual Property Management and Protection of a particular resource, and so on. As such, the model does not specify a language, but provides a set of concepts and terms along with the necessary grammar to describe relationships that can provide schemes for extensible descriptions. Examples of such terms and concepts used in the DID include containers, items, descriptors, components, anchors, resources, statements, predicates, and so forth. Some of the main terms are explained in the following list:

- *Resource*—A resource is a basic asset, such as video clip, image, or text. It could also be a physical object. All resources need to be properly defined by either having the data in place or a unique pointer to it.
- *Descriptor*—A descriptor qualifies an element in the Digital Item by associating information specific to that element. The elements are typically items or resources.
- *Component*—A component is a term used to describe a resource in the content. It contains a descriptor that qualifies the resource. For instance, the resource might be a compressed video stream. The component, then, is represented by

the resource along with a descriptor that contains structural and control information about the resource, such as bit rate, encryption information if any, and so forth. It should be noted that components are not items, but help build items.

- *Item*—An item is a group of components with a descriptor that qualifies the item. Items can contain choices that can be configured or customized using conditionals, predicates, and so on.
- *Container*—A container is a structure that logically organizes abstract representations, such as items or even containers. A container is a logically organized structure to be acted upon as a unit.
- *Anchor*—The descriptors used in components or other places can also be used to describe parts of a resource, in which case there needs to be an appropriate anchor to point it to the right position in the resource.

Figure 16-4 illustrates a hierarchical illustration of the Digital Item Declaration structure, which has been tailored to a specific example along with the Digital Item Declaration Language description.

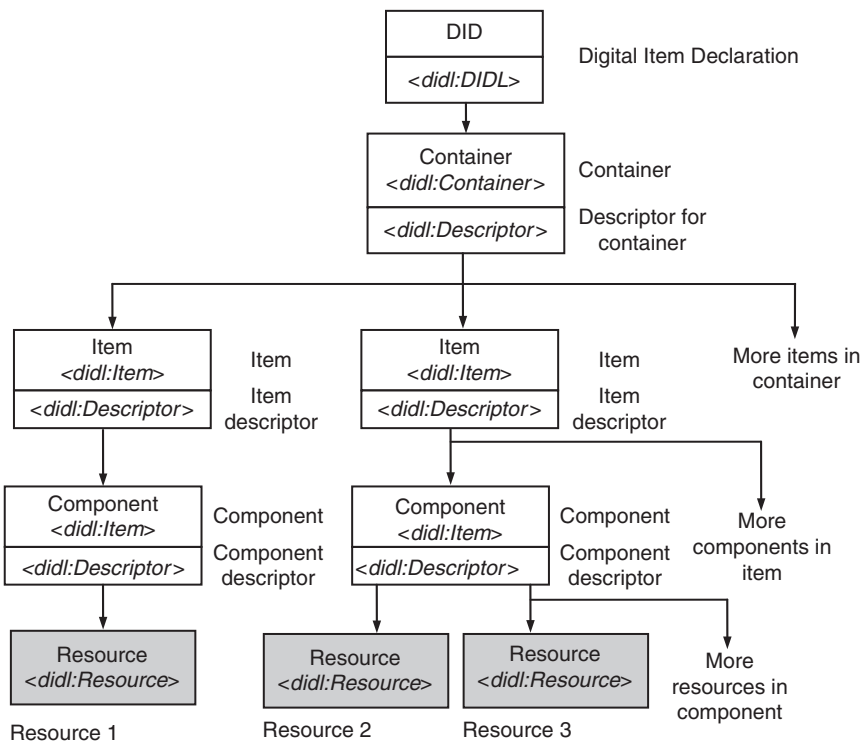


Figure 16-4 The Digital Item Declaration structure described hierarchically. Each declaration has a container with one or more items. Each item consists of one or more components and each component encapsulates one or more digital resources.

3.1.2 DID Representation

The abstract model described previously needs a standardized schema that can be used to express the syntax and semantics of the model in the context of describing a specific Digital Item. This is done by a Digital Item Declaration Language (DIDL), which is based on XML (Extensible Markup Language) and is used to describe the syntax and semantics of a Digital Item by using the model. For instance, the abstract term descriptor in the DID model is represented using DIDL by a descriptor element.


```
<didl:DIDL DIDLDocumentID="info:MPEG-21ID:media:ll28E67H5002"
        xmlns:didl="urn:mpeg:mpeg21:2005:02-DIDL-NS">
<didl:Container>
    <didl:Descriptor>
        <didl:Statement mimeType="text/plain"> World Cup Soccer Live Game
        </didl:Statement>
    </didl:Descriptor>
    <didl:Item>
        <didl:Descriptor>
            <didl:Statement mimeType="text/plain"> Advertisements
            </didl:Statement>
        </didl:Descriptor>
        <didl:Component>
            <didl:Descriptor>
            </didl:Descriptor>
            <didl:Resource mimetype="image/jpg" ref="http://www.loans.com/fooAd.jpg>
            </didl:Resource>
            <didl:Resource mimetype="image/jpg" ref="http://www.money.org/Ad.jpg>
            </didl:Resource>
        </didl:Component>
```

(Continued)

Figure 16-5 Digital Item Description. The top figure shows a media document with images and video. The DID specification for the same content is shown at the bottom using a DIDL-based XML schema.

```
    </didl:Item>
    <didl:Item>
        <didl:Descriptor>
            <didl:Statement mimeType="text/xml; charset=UTF-8">
            </didl:Statement>
        </didl:Descriptor>
        <didl:Component>
            <didl:Descriptor>
                <didl:Statement mimeType="text/plain"> Mpeg-4 Compressed Live Stream
                </didl:Statement>
            </didl:Descriptor>
            <didl:Resource mimetype="video/mpeg-4/simpleprofile"
                           ref="rtsp://www.worldcup.com/feed2411.mp4">
            </didl:Resource>
            <didl:Resource mimetype= "application/xml; charset=utf-8; size=12">
                FIFA World Cup: Goal number 64
            </didl:Resource>
        </didl:Component>
    </didl:Item>
</didl:Container>
</didl:DIDL>
```

4 DIGITAL ITEM IDENTIFICATION (DII)

Once content has been encapsulated in the framework of MPEG-21 as a Digital Item by making use of the Digital Item Declaration specification, there needs to be a way to identify Digital Items efficiently. The Digital Item Identification (DII) specification deals with the means to uniquely identify Digital Items. Although the DII specification does not provide new identification systems for the content elements for which identification and description schemes already exist, it does provide ways to use them to uniquely identify Digital Items and parts within them, which can then be used for proper digital rights processing, metadata processing, and so on. Overall, the goals of DII can be summarized as follows:

- A means to uniquely identify Digital Items, types of Digital Items, or parts in them, such as components and resources
- A means to use identifiers to link Digital Items with accompanying metadata information
- A means to uniquely identify description schemes
- A means to properly manage the intellectual property information

An illustration of how DII is specified for the Digital Item Description example in Figure 16-4 is shown in Figure 16-6. Parts of the description have been removed for brevity, but the *dii:Identifier* statement within the item's descriptor give the item its unique identity as shown in Figure 16-6

```
<didl:DIDL DIDLDocumentID="info:MPEG-21ID:media:ll28E67H5002"
           xmlns:didl= "urn:mpeg:mpeg21:2005:02-DIDL-NS">
<didl:Container>
    <didl:Descriptor>
    </didl:Descriptor>
    <didl:Item>
        <didl:Descriptor>
            <didl:Statement mimeType="text/xml; charset=UTF-8">
                <dii:Identifier xmlns:dii="urn:mpeg:mpeg21:2005:01-DII-NS">
                                urn:0-452445332-1
                </dii:Identifier>
            </didl:Statement>
        </didl:Descriptor>
        <didl:Component>
        . . .
        </didl:Component>
    </didl:Item>
    <didl:Item>
        . . .
    </didl:Item>
</didl:Container>
</didl:DIDL>
```

Figure 16-6 Digital Item identifier. The previous example is shown with an identifier embedded in the file. Identifiers are used to uniquely distinguish Digital Items and resources for interpretation, rights management, and so on.

5 DIGITAL ITEM ADAPTATION

The Digital Items that have been described in a standardized manner need to be adapted for exchange in various usage scenarios on different terminals connected across varying network characteristics. The adaptation also includes terminal capabilities, network conditions, and user characteristics.

The description based on DID schemas provides a fundamental structure that can be used by any adaptation engine. Some of the parameters that affect the adaptation are described in the following sections.

5.1 Adapting to the Terminal's Capabilities

A terminal is any device that receives and accepts a Digital Item in the media delivery chain. As such, the notion of a terminal is very general in terms of its adaptation. For example, obvious terminals include devices such as televisions, digital screens, mobile phones, PDAs, computers, and so on, but the definition could also extend to servers, proxies, routers, or any other intermediary nodes in the chain. Hence, terminal capabilities must be assessed in terms of receiving and forwarding Digital Items as well as interpreting Digital Items. For instance, a server might need to accept the Digital Item, interpret its digital rights issues, and decide to forward it to a receiver, whereas a

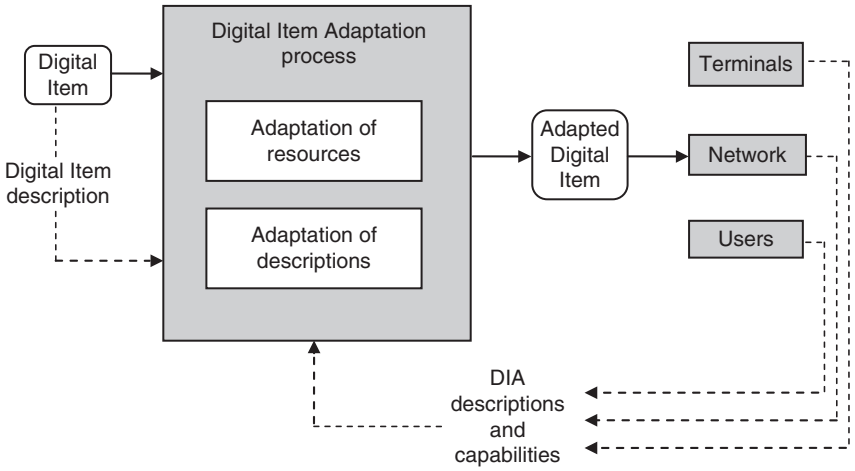


Figure 16-7 Digital Item Adaptation process. The content described in a standardized Digital Item needs to be interpreted and adapted on different platforms and networks and for different users. Each of these provides DIA descriptors that suggest their capabilities and properties, which are used by the DIA adaptation engine to appropriate and interpret the Digital Item for that respective platform.

receiver, such as a mobile phone on a low-bandwidth network, will need to interpret the Digital Item and accept only the parts that it is capable of consuming and rendering within any required real-time or non-real-time constraints. Correspondingly, terminal capabilities need to address the following different categories:

- *Codec capabilities*—Every terminal might need to decode parts of the content described in the Digital Item for access and even reencode or transcode it for further transmission, for example, a video coded in a certain MPEG standard using specific profiles and levels. Given the various coding formats available today, standardized or proprietary, it is necessary for the terminal to know which formats it is capable of handling as well as to know the controlling parameters and their effects when using codecs. Some standards, such as MPEG-2 and MPEG-4, have predefined levels and profiles that specify explicit constraints and limits required of the end terminal. In such cases, the profile or level should be used by the DIA engine, for example, MPEG-4 Video and Advanced Simple Profile. However, it is possible for some format encoding to not encapsulate such limits or even if defined, not use them in the encoding. In the absence of well-defined limits, the DIA engine should provide a means to describe such limits. It should know, for example, the maximum bit rate a decoder can handle.
- *Input/output characteristics*—Rendering and interaction capabilities are core to how terminals interpret and relate to Digital Items. For instance, understanding the capabilities of a display is needed for correctly rendering video content. Each display has resolution, color, and/or rendering format capabilities. Similarly, audio devices have mono, stereo, surround sound capabilities, power outputs, and signal to noise ratios. Users interact with the content using

different interaction modes. For instance, there might be a keyboard or a limited set of keys, a remote control, a microphone, and so on. If all this information is available to an adaptation engine, the adaptation engine can make appropriate choices that will lead to proper interactive consumption of the digital item on the end terminal, based on its description.

- *Device properties*—When the device is a physical terminal, it has certain limits for storage, capabilities of power, or processor speed and throughput. Storage characteristics are defined by the memory available, regardless of whether the memory is read only, the device can be written to, the buffer sizes used, or the input/output transfer rate. Such properties definitely affect the process by which a Digital Item is consumed by the terminal and the DIA adaptation engine can choose between streaming the data or storing it locally prior to rendering. Power properties provide information pertaining to battery capacity and time remaining. All these, when available, can be used to decide how to adapt the Digital Item to the device.

5.2 Adapting to Network Characteristics

Network characteristics deal with how the Digital Item can be effectively adapted so as to transport it efficiently across the network. The adaptation here can be described in terms of the capabilities of network resources and prevailing/changing network conditions. The former descriptions allow the Digital Item to adapt to static conditions, whereas the latter allows dynamic adaptations. The network capabilities suggest the maximum throughput of the network and the minimal guaranteed QoS, including throughput bit rate, error rate, and in-sequence/out-of-sequence delivery. Network conditions mostly suggest the current latency and error rates. The DIA adaptation engine can query network resources for the QoS properties and also dynamically monitor the network conditions so as to adapt the Digital Item descriptions for efficient transport.

6 DIGITAL ITEM PROCESSING

The previous few sections have shown how the standardized abstract model for Digital Item Declaration (DID) can be used by an author to describe digital content in the form of Digital Items using a Digital Item Declaration Language (DIDL). Using the information provided by the author in the DID/DIDL description, the item can be further adapted or adjusted depending on end terminal capabilities or network transmittal requirements. However, it does not provide an explicit format that can be used by the author to specify how an end user can interact with the Digital Item. This is the scope of Digital Item Processing (DIP)—to provide ways in which the content author's intentions can be realized, for example, how the item is to be viewed, interacted, or changed, if at all. For instance, DIP can be used for playing a resource adapted for a particular terminal described by metadata. DIP is implemented by supplying Digital Item Methods (DIMs), which are scripts that operate on Digital Items and their components. DIMs play the same role with DID documents that JavaScript plays with HTML documents.

DIMs are, thus, the key components that specify how a Digital Item is to be processed. It provides a mechanism that enables Digital Item authors to include

sequences of instructions that define the functionality of the item. As such, DIMs are not used for processing media resources but for interacting with them or even adapting them to the end terminal. A DIM is expressed in a Digital Item Method Language, much in the same way that that a DIDL is to express a Digital Item Declaration. A DIML specifies bindings for operations called Digital Item Base Operations (DIBOs). Each DIBOs implementation is supplied by the platform. DIBOs are functional building blocks utilized by a Digital Item Method.

An illustration of this process can be explained using our previous example in Figure 16-4. Here, the DID includes a video resource that the user can view. Let us say you want to embed the functionality that allows the end terminal to start playing the video as soon as it is connected, rather than waiting for the user to initiate the play (for example, by clicking a Play button). This might be conceptually simple, but will need different playback commands depending on the terminal displaying the video. For instance, the end terminal could be a computer, with the document adapted in HTML and JavaScript, or it could be on a set-top box with hardware-specific commands. A DIM must be specified in the DID to ensure how the terminal can play or interact with the resource. Figure 16-8 shows an example of this.

```
<didl:DIDL DIDLDocumentID= "info:MPEG-21ID:media:ll28E67H5002"
            xmlns:didl="urn:mpeg:mpeg21:2005:02-DIDL-NS">
<didl:Container>
    <didl:Descriptor>
    </didl:Descriptor>
    <didl:Item>
        <!--This is the actual digital item video content--!>
        . . .
    </didl:Item>
    <didl:Item>
        <!--This is a list of DIP Methods on the content--!>
        <didl:Resource mimeType="text/plain" ref="mpeg4Video#playTrack">
        </didl:Resource>
        <!--Implementation of DIP Methods on the content--!>
        <dip:Method mimeType="text/javaScript">
        function playTrack(mp4videoStream)
        {
            //Implementation to play Video Stream on Device
            <!--DIBO Bindings--!>
        }
        </dip:Method >
        . . .
    </didl:Item>
</didl:Container>
</didl:DIDL>
```

Figure 16-8 Digital Item Processing. The DID in the document should also include DIP methods that suggest how the end device should interact with the content.

7 DIGITAL RIGHTS MANAGEMENT IN DIGITAL ITEMS

Digital Rights Management has become crucial for controlling ownership and viewership of digital content. There are many specific solutions for specific markets. For instance, digital DVDs have their own authentication and encryption standards. Data shared among various digital devices, such as digital receivers, digital recorders, and digital displays, make use of the DTCP (and also HDCP) standards. MPEG-4 has developed the IPMP architecture for secure communication for MPEG-4 content. This has led to the development of many similar devices by different manufacturers supporting a variety of standards, but many of them are not compatible. When it comes to sharing digital content across different architectures, networks, and devices using the different in-place standards that are supposed to be interoperable, there is no overall solution to manage digital rights. Along with a standardized way to describe, exchange, and process Digital Items, one of the main objectives of MPEG-21 has also been to develop an interoperable framework for Intellectual Property Management and Protection (IPMP).

The requirements for rights management can be summarized from different viewpoints along the content distribution chain:

- From an authoring perspective, content authors have ownership rights over their content. The identification of this ownership, when included in a framework of Digital Items, is necessary to limit any unauthorized exploitation of content.
- From an end user's perspective, content should be available and accessible at any time and in a form that can be consumed by a variety of devices or in applications running on those devices.
- On the other hand, content providers want to distribute content in a manner that will help create rich and intuitive business models for specific end-user requirements. For instance, a content provider might want to distribute movie content in high or standard definition, with or without interrupted advertisements, with surround sound or stereo.
- From a hardware device and/or software manufacturer's perspective, cost is always an important factor especially in the light of increasing the value of the device for consuming various kinds of content. For instance, a central device(s) should have the capability of consuming content of various kinds by seamlessly interconnecting to different device(s)—and at the same time respecting as well as enforcing the digital rights imposed by the content owner.

In other words, the entire creation, distribution, and consumption framework must be capable of specifying and communicating rights information at each step of the way. That is, content owners and authors need to describe these rights and grant them as intended for an end user, content distributors and retailers need to understand the rights that pertain to content distribution, and end users need to know what rights are associated with the content and what has been granted to them. These rights can be simple or complex. In case of our live video stream example, a distributor might have bought the rights to unlimited playback and duplication of the video, an end user

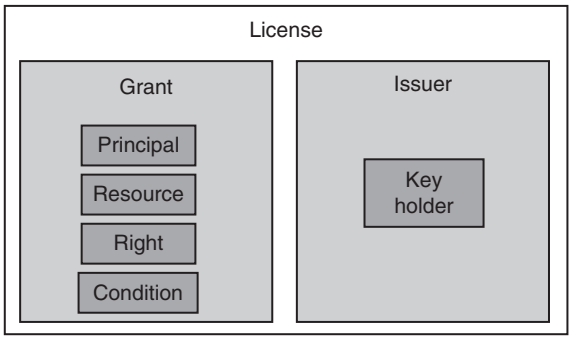
might have permissions to view its high-definition stream, another user might be allowed to view only a lower definition, with or without advertisements, and in either case, the user might or might not be allowed to record the stream for replay later. Therefore, the overall management of digital content rights proves to be a complex task requiring a more methodical model to standardize the management of digital rights. The need to have a standardized formal description here is not only for interoperability, but also because these rights might need to change over time. MPEG-21 addresses these needs by the use of a common Rights Data Dictionary (RDD), which describes possible keywords for rights management, and a Rights Expression Language (REL), which makes use of RDD to describe how these rights are supposed to be managed.

7.1 Rights Expression Language (REL)

The MPEG-21 REL is an XML-based declarative language that can specify rights and conditions for the approved distribution and use of any resource(s). Central to REL is the data model that makes the description effectively interoperable over different areas in the production and consumption of the content chain. The fundamental unit in the data model is a license. In a license, an issuer (content owner, distributor, and so on) grants one or more rights (actions to play, modify) to principals (end users, terminals along the chain) to access a resource (video stream, the main content) under certain conditions (time limits). These terms can be more formally defined as follows:

- *License*—A license is the main construct of REL and specifies a container structure that contains issuers, grants, and other related DRM information. Theoretically, a license is an abstraction of grants issued by one or more issuers.
- *Issuer*—An issuer is a term within the license that points to the party that issues the license. The issuer can also include digital signatures that can be used to authenticate the principals to whom the license is given.
- *Grant*—A grant is a term in the license that gives the specified right to a principal for a resource under possible conditions. Conditions include a one-time use, a time-length use, or unlimited use.
- *Principal*—A principal is a term within the grant that allows the identification of a party to whom a right is granted. Principals are normally identified by information unique to that party and that is used to authenticate the party. Such principals are described by the term keyHolder, suggesting that they hold a public/private key for authentication purposes.
- *Right*—A right is an action on a resource. It is granted by the license to the principal. REL provides rights, such as play, print, and adapt, as well as metarights, such as issue and revoke.
- *Resource*—A resource is the content object to which the principal is granted a right.
- *Condition*—A condition is a term used to describe any contingency or obligation that the principal needs to follow while exercising the right.

The preceding terms describe the basic elements in the REL model. Figure 16-9 illustrates the REL model implemented in XML for the video stream example.




```
<didl:DIDL DIDLDocumentID="info:MPEG-21ID:media:ll28E67H5002"
            xmlns:didl="urn:mpeg:mpeg21:2005:02-DIDL-NS">
<didl:Container>
    <didl:Descriptor>
    </didl:Descriptor>
    <didl:Item>
        <!--This is the actual digital item video content--!>
        . . .
    </didl:Item>
    <rel:License>
        <rel:Grant>
            <rel:KeyHolder licensePartId="John Doe" digitalSignature="AffD45sss38d">
            . . .
            </rel:KeyHolder>
            <rel:Right value = 'play'>
            </rel:Right>
            <rel:Resource>
                <dii:Identifier xmlns:dii="urn:mpeg:mpeg21:2005:01-DII-NS">
                </dii:Identifier>
            </rel:Resource>
            <rel:Condition>
                <notBefore> 2008-01-01 00:00:00 </notBefore>
                <notAfter> 2008-07-01 00:00:00 </notAfter>
            </rel:Condition>
        </rel:Grant>
        <rel:Issuer>
            <rel:keyHolder licensePartId="FIFA BroadCasting Authority">
        </rel:Issuer>
    </rel:License>
</didl:Container>
</didl:DIDL>
```

Figure 16-9 Rights Expression Language (REL) in MPEG-21. The top figure shows the abstract model of a license that is the fundamental construct of REL. The model is shown implemented as XML schema for the video stream example of the earlier figures.

7.2 Rights Data Dictionary (RDD)

The RDD consists of key terms required to describe all the rights of users for viewing or consuming content, including intellectual property rights. The primary purpose of RDD is to support REL by assigning semantic meanings to the terms used by REL. For instance, REL is used to grant the rights to use a resource to a principal. This right can take on many values, such as play, modify, and so forth. The semantic meaning of these values is cataloged in the RDD. The baseline dictionary provided by the MPEG-21 specification contains more than 2000 terms and also provides a basis for adding new terms if so desired. These terms are organized into families according to the roles they play in different contexts.

RDD and REL are intended to be used in conjunction to provide a flexible, interoperable mechanism to support definition of, operation of, and enforcement of digital rights for content passed along a publishing or distribution pipeline. In addition, they also support the access and control of digital content in cases.

Term	Definition
Adapt	To convert to a new resource as per the DIA statements—e.g., a video stream has to be adapted for a specific terminal
Delete	To delete a resource—e.g., removing a resource after its usage
Diminish	To convert a resource into a smaller resource—e.g., converting an image to a smaller image
Embed	To include a resource into another resource—e.g., including a watermark in the media that reaches the end client
Enhance	To convert a resource into a larger resource—e.g., enlarging an image
Enlarge	To modify a resource by including more information—e.g., adding a stereo stream to a mono audio stream
Execute	To run a digital resource—e.g., running a binary file, which is the digital resource in this case
Install	To follow instructions as outlined by an installer
Modify	To change a resource—e.g., performing decryption
Move	To move a resource from one place to another—e.g., moving where a file is available from one place to another
Play	To render or to make a visible/audible representation of the resource—e.g., listening to an MP3 file
Print	To duplicate the resource—e.g., making a copy of a video stream locally
Reduce	To modify a resource by shortening it (opposite of Enlarge)—e.g., converting a stereo stream to a mono stream
Uninstall	To follow instructions as outlined by an uninstaller (opposite of Install)

Figure 16-10 Table showing some of the Rights Data Dictionary terms. The terms are action types used to describe a right in the Rights Expression Language.

In conclusion, MPEG-21 is the latest standard ratified by the MPEG community. Unlike the previous standards aimed at audiovisual communication (MPEG-1, MPEG-2), content description (MPEG-4), or metadata standardization (MPEG-7), the MPEG-21 standard has much broader scope, aiming to facilitate a seamless exchange of digital content across different networks, devices, and services, each having different business models and contexts. As digital sectors proliferate, so does the content in each sector. It is imperative to standardize the process of exchanging data among sectors to empower end users with information and entertainment choices. This is not an easy task, but MPEG-21 is certainly a start to enabling this standardization.

8 EXERCISES

1. [**R02**] Digital Items are the fundamental units of exchange or transaction in MPEG-21.
   - Why is it necessary to describe every media element as a Digital Item? What does this achieve and how is it achieved?
   - Given a specific content, is the Digital Item description unique? Explain.
   - Mention some of the problems that make this description a challenging task.
2. [**R02**] The Digital Item Declaration (DID) model is used to describe any multimedia document as a Digital Item in terms of an XML schema specified by the Digital Item Declaration Language (DIDL).
   - Explain the abstract model in terms of its flexibility and extensibility.
   - Mention a few salient points of the hierarchy of a DIDL document.
   - How many identifiers can a DIDL document have? What are they used for?
3. [**R04**] A video is coded for a high-band network. Now you want to access the coded video on a low-bandwidth video. You decide to use the MPEG-21 framework.
   - Explain how you would describe the Digital Item and its adaptation (DIA) across the different networks.
   - Thereby, justify the need for Digital Item Adaptation.
4. [**R05**] The Rights Expression Language is based on an abstract model whose main construct is the license that is granted by an issuer to a principal. Explain the abstract model. How would you implement a license using REL in the following cases?
   - An online movie hosting house is providing consumers with movies to watch. Consumers can play any film in its library for a $1 fee. Licenses are granted and valid for 10 days before they expire. Once a consumer has viewed the movie, they can replay it any number of times during these 10 days, after which the license expires.
   - How will you modify the preceding case to a condition where there are many video online companies that charge different fees for viewing movies in their libraries?

5. **[R04]** Here is another interesting variant on the preceding problem. Suppose the online movie companies are allowed to grant licenses to paying end users, but only on the condition that the movie is viewed on a trusted device. For example, you want to implement a license issued by company X to a user Y that will allow Y to view a movie only on a specific type of video player that has been deemed as trusted by the content owner. This can be done because the content owner is certain that the video player has no mechanism to store or otherwise illegally allow recording of the content. How would you implement a license using REL in this case?

This page intentionally left blank

CHAPTER 17

Concluding Comments and Future Perspectives

We started this book with the goal to educate colleagues, students, and multimedia enthusiasts about the theory and practical implementations used to produce interactive multimedia content, compress/deliver it on various networks, and consume multimedia content on end devices with different capabilities. The primary lesson we can draw from the text in the book is that we are in the midst of a revolution that has deeply altered and still is altering the way we create, access, deliver, and consume information.

In this concluding chapter, we would like to provide a closure on the material that has been covered in the book. This is done by summarizing the concepts talked about thus far, by pointing out aspects of multimedia that were not covered in the chapters, and also lead you into an exploration of multimedia progress that is yet to come. The chapters of the textbook should have exposed you to technical elements that go with the field of multimedia. Section 1 summarizes the three parts of the multimedia pipeline that we have discussed in the book—content creation, compression, and distribution. However, the dynamic nature of the field itself makes it impossible to discuss everything. Consequently, there are topics that we alluded to but for which we did not provide in-depth coverage, topics that have matured into applications since we undertook this book project, and many more still being researched so as to shape the multimedia field of tomorrow. Section 2 illustrates such topics that we have not covered. Section 3 delineates an outlook in different areas where multimedia has had an impact and the problems that need to be solved to meet the requirements of the next generation of multimedia applications. Finally, Section 4 provides a few futuristic trends from the author's point of view. As such, it is not easy to provide a distant perspective while in the midst of all of the technological changes that are happening, but current trends can certainly be extrapolated.

1 WHAT HAS BEEN DISCUSSED IN THIS BOOK

The chapters in this book give an overall view of three aspects of multimedia systems—content creation, effective compression, and the distribution of multimedia content.

The first part of the book talks about content creation. Regarding content creation, we have discussed the main issues and requirements involved in the digitization systems that record and convert information into digital representations. Once the signal is in the digital domain, it needs to be represented in different formats, including open standards (such as MPEG and JPEG) as well as proprietary standards (such as QuickTime video, RealAudio, and so on). Chapter 3 enumerates many such representations that are now in common use for digital media types explaining the advantages of one over the other. For visual media types such as images and video, we have also explained the different color representations, the color matching theory, color spaces, and how it all relates to the color calibration process. This calibration and standardization process is needed to allow any manufacturers of capture devices such as digital cameras and camcorders to capture digital imagery such that the colors look the same when reproduced on any display device, such as televisions, cathode-ray tubes, LCD, plasma displays, and printers, which might, in turn, be manufactured by different companies. For audio, we have seen the different audio representation formats from mono, stereo, and 5.1 to the more commercial THX and Dolby systems. Also, selective techniques and algorithms widely used to combine various media types to create multimedia content are discussed. These include image processing effects for images, chroma-keying and compositing for video, simple audio filtering, creating graphical animations, and many others. We have also discussed authoring paradigms and the design implications that involve not just a single user assembling and creating multimedia content but also those that involve many users over a distributed architecture and a collaborative medium.

The compression aspects of multimedia information are dealt with in the second part of the book. Here, we explain the theoretical bounds of information as well as standard compression algorithms for each of the media types—images, video, audio, and graphics. Also discussed are compression issues that need to be addressed when designing real-time media distribution systems. These include time/complexity restrictions, practical constraints that relate to memory requirements, bandwidth requirements, and streaming while simultaneously preserving the delivery bit rate. In that regard, we have formally introduced what signifies the information content in signals and how to make effective use of this in compression techniques. Although a generic taxonomy of lossless and lossy techniques has been introduced in Chapter 6, it is followed by Chapters 7 to 10 that are devoted to the compression of each media type, explaining the techniques and theory involved as well as the standards that are used to represent, exchange, and distribute compressed media content. Some of these include the DCT-based JPEG and wavelet-based JPEG 2000 for images and the MPEG and ITU formats for video, such as MPEG-2, MPEG-4, H.261, H.363, and H.364. The prominent formats included for audio are the MPEG formats such as MP3, AAC, and CELP as well as the Dolby AC3 formats. Graphics compression formats are fairly new in comparison with other media formats, but we do discuss some of the main formats used in exchange of 3D graphics data, such as topological surgery in MPEG-4 and X3D.

The third part of the book is devoted to multimedia distribution and media distribution over a variety of wired and wireless networks. Also addressed are the distribution-related issues, such as medium access protocols, unicast versus multicast, constant bit rate traffic, media streaming, and so on. In particular, it also formalizes QoS requirements on both wired and wireless networks for distribution of multimedia traffic and explains how to control multimedia traffic using flow control, congestion control, and techniques to address latency issues. Also discussed are the main standards used for non-real-time and real-time media distribution, such as TCP/IP, UDP, HTTP, RTP, RTSP, and wireless WAP protocols on 2G and 3G networks. This section also addressed a parallel though related and necessary aspect in media distribution—that which relates to controlling the security and management of digital rights, which includes the current state of the art in digital watermarking and encryption techniques. Here, we also illustrate various commercial DRM solutions that have been deployed in the music industry (Apple Computer's iTunes and iPods), the consumer electronics industry (DVDs, Blu-ray), and the digital cinema industry, which is yet to be fully realized.

The last part of the book gives a discussion of recent trends and applications in multimedia that are now used in the industry. Among the topics discussed are the latest MPEG-4 standard, multimedia frameworks using the emerging MPEG-21 standard, multimedia-based querying, and indexing in multimedia databases. Although text-based databases are very common now, no commercial solutions give the same indexing capabilities for multimedia databases. This part explains the issues involved in multimedia querying and why it is not as easy as text databases. We also give examples of multimedia standards, such as Descriptive Metadata Scheme-1 and TV-Anytime used in the television industry as well as other standards such as the MPEG-7 and the Dublin Core.

2 WHAT HAS NOT BEEN COVERED

It should be clear by now that multimedia pipelines and multimedia applications touch many different concepts that include signal processing, color theory, computer graphics, image processing, video processing, information, compression and communication theory, computer networking, wireless technologies, digital rights management, databases, and so on, as well as well-suited software architectures that allow for processing, compressing, and communicating huge amounts of multimedia data. Although a lot has been covered to illustrate the multimedia pipeline, some aspects of the theory are not covered. Some of these topics were deliberately left out because of the defined scope of the material that we wanted the book to cover, some topics were not given much formal treatment when the topic was really vast and especially when other texts that singularly explain the subject matter are available (for example, signal processing, computer graphics), and other topics could not be covered because the topics are still in a research phase that will make an impact on the multimedia field in the future.

Although we have discussed capture processes, representations formats, and some selective methods to create multimedia content, we have by no means done any justice

to the area of content creation and processes involved in content creation. Content creation is a vast area that typically requires artistic viewpoints that make use of technology to create the needed content. Every industry today has its own mature applications that help assemble, composite, and create content. For instance, the advertising industry makes use of tools such as Dreamweaver, Macromedia Director, and a suite of Adobe tools (Photoshop, Premiere, AfterEffects) to create effective content that mainly involves images, video, text, and audio along with Web publishing tools and Web hosting tools to set up content on a Web site. The processes followed by the video game or the movie industry involve more emphasis on 3D graphics. Many use popular software tools such as Autodesk's Maya, 3D Studio, and LightWave to create and animate 3D content, which is then rendered in a real-time game engine that exploits the graphics hardware or offline software rendering setups such as Renderman. Additionally, matured research in the field of computer graphics and computer vision has allowed applications to create panoramas of images and videos. These now provide effective metaphors of interactively exploring environments and visualizing events. Another area of computer graphics is image-based rendering, modeling, and lighting. It is now routinely used to create synthetic, though realistic, imagery in the field of visual effects and animation in high-end movies. Improvements in capture technologies that provide high frame rates and high resolutions have created applications like motion capture, a standard process in animating digital characters in movies and games.

There has been considerable research in trying to extract semantics from media, which should make end terminals smarter by being able to gauge the consumer's desires and intent automatically using cameras and voice recordings. Just as consumer end terminals are getting smarter and more interactive, so are some of the distribution channels. Recent technological improvements have made the deployment of small, inexpensive, low-power, distributed devices, which are capable of local processing and wireless communication, a reality. Such nodes, called sensor nodes, are capable of only a limited amount of processing, but when coordinated with the information from a large number of other nodes, they have the ability to measure a given physical environment in great detail. Unlike traditional networks, sensor networks depend on dense deployment and coordination to carry out their tasks.

3 CURRENT ADVANCES AND IMPACTS

Technologies to capture, compile, organize, interact, and disseminate multimedia information continue to evolve. The evolution continues into the standardization of all required protocols, as well as the businesses and applications in these areas. Throughout this change, we can surmise a few trends:

- Multimedia information is becoming more mobile and easily accessible.
- Devices people use to create and access information are increasing in functionality and modalities, while at the same time decreasing in size.
- Rather than specific agencies or groups being held responsible for creating, organizing, and disseminating information, every user is slowly but steadily being equipped with the facilities to do the same.

- Media-based information dissemination is changing from a broadcast-like metaphor, with the user's participation being passive, to active interaction.
- Online collaboration and grouping of individuals to form collaborative/social networks is rising.

The following sections discuss impacts in different domains and the common problems that will need to be overcome to meet the next generation of multimedia applications to fulfill anticipated needs.

3.1 Impact on Information Organization

Until not so long ago, and it might still be so in some part today, the main medium of information storage was physical in nature. Examples of such mediums include paper for books, newspapers, catalogs, analog media storages such as cassette tapes and vinyl records for music, VHS video tapes for movies, movie reels for cinema, and so on. Some of these might have disappeared today and been replaced by their digital counterparts, such as CDs for music and DVDs for movies. And although the digital information evolution is still under way, an important aspect of our day-to-day handling, organization, and delivery of all this information is still tightly bound to the physical manifestation of its storage medium, whether analog or digital. For instance, we are still accustomed to thinking of a page in a book, or books organized on a shelf, a musical melody on a CD, CDs organized on a rack, purchasing a DVD movie from a store, albeit an online store, and so on. Also the unique physical location of each object necessitated the development of taxonomies to help index physically stored information. Examples of such taxonomies include a chapter index for a book, or alphabetical ordering on a shelf or CD rack, and even the Dewey Decimal system for storage of books in libraries. Although digital online commercial transactions are common using the Internet, we still get physical objects delivered to us as the end result of the transaction. For instance, we get books delivered to our homes and we buy CDs or DVDs from online stores. This tight coupling between digital information and its physical medium is undoubtedly going to change with high-bandwidth digital networked distribution and with the availability of smart, versatile digital devices to consume this information.

Consumers have started to expect digital delivery of information, whether a movie, a melody, a magazine, or a story, which will be stored on digital devices and consumed appropriately on digital devices. This is already true in the case of digital television, online news, video-on-demand, and also the *Kindle* popularized by Amazon for selling e-books. Information is being digitized and networked in virtually all areas and the distribution of this networked digital information is no longer limited by physical constraints or physical media. The advances in digital distribution technologies and the increase in conveniences provided by smarter digital devices to consume multimedia information are changing our habits and breaking our dependence on the need for a physical medium. With this change comes the need to organize, to access, and to index distributed digital information similar to the taxonomic abilities provided by the physical medium. Such digital taxonomies have already evolved. For instance, while browsing the Web, you can bookmark a sight and organize bookmarks so that it is easy to go

reaccess the information. We tag blogs and Internet postings so that we can read them later or send pointers to friends and other information seekers. In many ways, this is similar to the physical world of organizing information where you yourself are organizing information for your purpose or a group of people organize it for a portal. When you go to an information portal such as Yahoo! or YouTube or MySpace, you can either browse the information or search the site for information because a group of people have organized the information or have processes in place that allow easy information access. When reporters submit their digital news articles to newspapers or magazines, that information is properly organized and tagged, making it easier for visitors at the Web site to read, browse, and search.

One major difference from the paper world is that digital information is not static but dynamic, constantly and rapidly increasing with new sites, or new information posted on the same sites either by a group of people, an agency (such as *New York Times*), or by individual users like you and I in the form of individual blogs, posting your own music, or uploading your personalized photographs and videos. Organizing all this information, whether by hyperlinking or cross-referencing, is a staggering undertaking that cannot be accomplished by a few people who either own the site or maintain the hosting site. The dynamically changing and ubiquitously shared nature of the information needs an equally dynamic organization effort. Examples of such dynamic organization paradigms are already being used but need to be developed further so that users can make use of efficient information browsing. The following are some paradigms that are being developed today:

- Every user plays an active role in information organization where users create information, edit information, and organize information. Wikipedia is a prime example of this. Wikipedia is a digital encyclopedia and already has around two million articles, which are constantly updated, with newer ones listed daily. Each article is cross-referenced to other articles wherever necessary with all this done by all users of the site. Instead of being passive browsers, users are now active participants in shaping the information content and its organization.
- Additionally, there has been a lot of research in designing software programs, normally called "bots," which crawl the distributed information and automatically pull together articles either to search for and even compile new information.
- As information is changed and organized at a site, automated methods send out e-mails, text messages, and so on to users to encourage them to revisit the site.

3.2 Impact on Content Delivery

Most media distribution and delivery setups have now become service centric, where content providers have combined forces with the distribution industries to provide content as a service. Examples of these services include digital cable television, satellite television and radio, digital news, e-mail, Voice over IP, and so on. Many computational resources are being put into the Internet delivery infrastructure and provided as services to customers. The result is that there are now a variety of ways in which

content is delivered in a standardized manner on a variety of platforms. Three dominant content delivery systems exist today: the client/server-oriented World Wide Web, content delivery networks, and peer-to-peer file sharing systems. Although these systems serve the same purpose, that of distributing content to users, the architectures of these systems differ significantly, and the differences can affect their performance and their workloads.

The basic architecture of the client/server-based Internet is simple. Web clients run on end terminals and request content from Web servers. Content-delivery networks (CDNs) are dedicated collections of servers that are located strategically across the wide area Internet. Content providers work with commercial CDNs to host and distribute content. When content is hosted in a CDN, it is replicated across the wide area and, hence, is highly available, thus making it attractive to content providers. The largest CDNs have thousands of servers dispersed throughout the Internet and are capable of sustaining large workloads and traffic hot spots. Peer-to-peer (P2P) file sharing systems have surged in popularity in recent years. In such systems, peer nodes collaborate to form a distributed system for the purpose of exchanging content. Peers that connect to the system typically behave as servers as well as clients: A file that one peer downloads is often made available for upload to other peers. Unlike the Web and CDN systems, the primary mode of usage for P2P systems is a noninteractive, batch-style download of content.

3.3 Impact on Service Providers

Let us extrapolate this phenomenon further, with the example of the home. Just as in the past, even today a number of different providers give access to different communication channels into the home. The telephone company brings a twisted pair of wires into the home to deliver and transmit telephone conversations, connecting you with the POTS network. Note, however, that the infrastructure provider (who gives you the wires) and the service providers (who provide long-distance communication service) are separate and distinct. You can have different service providers for local and long distance calls, for instance. Similarly, television signals are delivered through an antenna for broadcast over the air channels, or through a coaxial cable for cable channels, or through a dish antenna for satellite service, or even through dedicated fiber-optic lines. In the last three cases, a set-top box allows interaction with the user to select channels, order special programs, and also validate your access to the requested content. Finally, Internet access, once predominantly through the phone line as a modem, can now be obtained through a dedicated phone line, satellite, or cable.

The result is that companies are now offering a set of services such as TV, telephone, and Internet, regardless of the underlying transport mechanism, which might be coaxial cable, optical fiber, and satellite. Buying TV and Internet service from a wireless phone provider such as Verizon would have been unthinkable a few years ago. Although the end result for a consumer desiring Internet, television, and telephone access is the same, the choice of service providers creates a marketplace that has continuously brought down access charges.

3.4 Impact on Source of News

The printing press enabled the emergence of newspapers as the main source of news. Other distribution means such as radio and then television overcame newspapers. Television still dominates all other sources today. In 2008, the Internet overcame newspapers as an outlet for national and international news. This evolution has thus far had drastic implications for the newspaper industry, as the revenue model has drastically changed. As the newspaper subscriptions dwindle, it has brought forth multiple issues for newspaper companies:

- A smaller subscriber base leads to reduced direct revenues but also to lower revenues for advertisers.
- Newspapers have established an online presence through their Internet edition, but the online version is free, leading to a further decline in the paying subscription base.
- The immediacy of news on the Internet changes the role of newspapers into delivering not only news, but also an analysis of news, with requires more resources.

4 FUTURE TRENDS

The advances in information technology, computational capability, and communication networks have resulted in large-scale multimedia content creation, distribution of vast amounts of multimedia data, and an ever-growing need for applications involving multimedia. Though numerous efforts have been devoted to these aspects and in large part have now become a practicality, it is still far from maturity and there exists many open issues that are currently being researched in universities, laboratories, and industries. In this section, we provide a few such challenges and requirements that will hopefully change the future multimedia landscape.

4.1 Need for Content Adaptation

The ever-growing variety of audiovisual coding formats and their delivery on a wide variety of networks will require a methodology for processing of multimedia content without considering the specific coding format used. The end consumer of multimedia content is not interested in the details of the coding format, how the content was delivered, or the device the end user is using to consume the content, but only in the actual content itself, which may be a movie, image, or music. For example, if a consumer pays to watch a video, he or she will desire to view it on his television, on his computer/laptop, or even on his iPod. On the other hand, the content and service providers also aim at avoiding maintaining multiple variations of the same content, each conforming to a different coding format, to fulfill the requirements of their customers. One way to solve this problem is to provide a Universal Multimedia Access (UMA) mechanism by allowing content to adapt itself to the distribution network and also to the end

terminal. That way, content bit streams are encoded once and, with additional information, can be adapted to delivery on different delivery platforms or for consumption on different end terminals. This will need a variety of capabilities provided by distribution nodes and end terminals, including transcoding, scalable encoding and decoding, adaptable digital rights management solutions, and so on. UMA is the main goal of MPEG-21 and will provide a technical framework for adapting content.

4.2 Intelligent Interfaces and Semantic Processing

Current multimedia applications and interfaces mostly never involve any semantic extraction and analysis. As capture devices get better (more resolution, better frame rates), as processing power and memory is no longer an issue, as research in semantic extraction matures, there will be more meaningful automated applications. For instance, interfaces for interacting with devices will make use of cameras and voice, not needing keys, pointers, touch screens, and so on. The places that we inhabit, work at, and live in will be more intelligent in sensing our desired conveniences and follow through automatically.

4.3 Generating Personalized Content

With the plethora of information that is now available, generating personalized content tailored to the needs of a user is gaining popularity. Generally, this is thought of in terms of generating textual content and there are few websites that generate personalized information for their users. For instance most websites can be tailored to suit a user's choice of information. Although most of the information organization is based on text with hyperlinks to other media, the paradigm can easily be extended to all different forms of media, video, and audio. Creating new meaningful content either completely or using existing content is still a nascent area in research today. However, useful tools are available today that help attain a certain degree of personalization. For example, Web feeds and Web channels are HTML or XML documents that contain pointers to the longer versions. There are two main Web feed formats: RSS (Really Simple Syndication) and Atom, which is an XML-based document format that describes lists of "related" information known as feeds. Such feeds allow software programs to check for updates published on Web sites according to user preferences, thus allowing users to automatically get updated content. Other attempts at creating personalized content are by the use of sophisticated programs called *Web robots*, *Internet robots*, or simply *bots*. These bots are programmed to perform different tasks depending on their level of sophistication, which can range from doing simple, repetitive tasks such as checking for updates on sites, to aggregating information from a large number of sites on a specific topic. Advanced bots with limited capabilities can also create reports for easy reading. Although the sophisticated operations need semantic analysis and do still fall short of human operations, they do show what we might expect of bots in the future. Current bots only attempt to organize textual content according to user preferences, but there will be a need to do the same for images, video, and audio content as more of these become available. Understanding the semantics of these multidimensional signals is still an unsolved problem at a general level.

4.4 Information Overload and Filtering

With the vast amount of digital information, which might be organized/disorganized, which might be hyperlinked/unlinked, which might be tagged/untagged, and so on, any search on a topic will return a tremendous amount of information resulting in an information overload state. The information will contain items written and compiled by groups of experts and knowledgeable people as well as novices, and nothing currently precludes a search from returning ill-informed and inaccurate information. Without any filtering, it is not possible to rate the correctness and the importance of the searched information. Current search engines do attempt to filter and rank information on Web pages so that it is presented in an organized manner for the user. For instance, search engines filter based on whether the search phrase appears in the title of the Web page or how many times a search phrase appears wholly or partially on a Web page. Others use more sophisticated techniques such as the PageRank algorithm utilized by Google. PageRank associates a numeric value that suggests the importance of a page on the Web. When a page is linked to by the information on another page, it effectively gets a vote. The more votes that are cast for a page, the more important the page must be. Also, the importance of the Web page casting the vote also factors into the value of the vote. Although the process involves more heuristics, this logically filters Web pages that have resulted for a search in a ranked manner depending on how many and which other pages point to it.

Although filters exist to organize information, they mostly depend on the spatial location of the matched string or hyperlinked information. This hyperlinking or any other form of organization must be preassigned by a user or another process. If Web pages that contain efficient information are not hyperlinked, or if they contain data not formatted relative to the expectations of filters, then it is unlikely that they will show up in a search. Correspondingly, incorrect information or partially correct information might also be ranked higher if the data organization on the page befits the filter-processing requirements. Current filters make no use or limited use of the information semantics to rank it. And although the problem (not solution) is straightforward to understand for text in Web pages, it is not so for images or video. Computational models for image understanding and video analysis remain active areas of research, which at a future date might become useful tools for information extraction and organization.

4.5 User Connectivity, Digital Communities, and Beyond

Digital connectivity among users today manifests itself in different ways. On one hand, users communicate and connect with each other via a multitude of ways, such as e-mail, instant messaging, mobile phones, and so on. This connectivity is predominantly between two users, can be active or passive, and the communication content is more personal to the two users. On the other hand, groups of users come together and exchange information in today's digital networks. Popularly called social networks, these online communities represent a major shift in personal connectivity and are starting to become major media carriers. Social networking is attributed to the abundance of Web-based applications, such as Facebook, MySpace, YouTube,

LinkedIn, and so on, with large online communities of people who share common interests. These sites usually allow members to build a profile, in the form of a Web page where you display information about yourself, your interests, and your activities. Most of the sites also offer different ways for users to communicate with each other. For example, you can allow others to write a message on your profile, send messages, join a live chat or forum, share and rate other Internet sites, or share and comment on files (audio, image, or video).

Just as technology has changed the way we work and communicate, social networking will also influence the way we communicate. Although there are a few examples where social networking sites have been dedicated to exchange purposeful information, most social networks are being used to exchange personal and mostly unimportant information. Therefore, they have in the past been viewed as frivolous. However, organizations will soon start to tap the benefits of digital social behavior to improve productivity in the workplace and support a culture of collaborative innovation. These might be termed as enterprise social networks and we do see the beginning of these in the form of wikis, but they have so far not been exploited to create a networking behavior of subnetworks and groups for specific projects and ideas. Also, in the current state of social networks, all friends are treated equally. Intelligent connectivity applications need to be developed to allow users to better define and maintain relationships. Today's social networks largely remain atomic in their operations and do not allow synchronization of activities, information across multiple networks, and so on. There is a need to develop advanced applications and services that will help easily manage the multiple social networks' profiles, help users manage information, and synchronize online activities with offline.

The semantic Web and the utilization of semantics in multimedia communication is the next evolution of the Internet connectivity experience. Web applications and experiences need to be *intelligent* and *sophisticated* by understanding the context in which the users access them. The first version and usage of the Internet manifested itself as applications in which people could talk to people. The second evolution, Web 2.0, changed this by allowing people to talk to applications as well. The next step in this evolution, Web 3.0, should allow applications to talk to applications, where the user's requests are automatically processed and presented based on the user's behavior, habits, and personal data. Much of this multimedia data resides in social networks right now. As such, anyone using the Internet and social networking applications is already *connected*. The next step, then, will be to help users convenience their lives by automating simple tasks of information exchange.

This page intentionally left blank

Answers to Selected Numerical Problems

Chapter 2

1. 0.01171875 Volts, 0.000732421875, −0.93 percent
2. 240 Kbps, 640 Kbps, 96 Mbytes
3. 1.512 Megbytes, 0.504 Megabytes, 0.504 Megabytes
4. 2.2222 Mbps, 71428.5 seconds, 85.47 mins
8. 15 Hz, 7.5 Hz, −4.5 Hz 0 Hz 1.5 Hz, 57.57 km/hr
10. 70.2 Mbps, 3.95 GBytes, 93.6 Mbps, 5.265 GBytes

Chapter 3

1. 12 bits, 16 bits, 12 bits
2. 70.2 Mbps, 3.95 GBytes
3. 277 images, 1944:1
4. 98.2:1, 73.66:1
6. 51.84 Mbps, 4:3, PAR changes to 2:1
8. 1:1, 1:1, 1.125:1
9. No, 149.15:1, Yes

Chapter 6

2. 146220 bits (approx)
4. 82.944 Mbps, Avg bit rate = 57.438 Mbps, Efficiency = 99.16%
5. 3 (without probabilities), 1.84 (with probabilities), 1.875 and 98.13%, H for pass no pass case = 0.5675
10. Average code length = 1.375, Four codes possible, not optimal
11. H = 4.131, normal binary needs 300 bits and Shannon Fano needs 266 bits (answer may vary), normal binary needs 430 bits and Shannon Fano needs 466 bits (answer may vary).

Chapter 7

1. 967680 bytes, 322560 bytes, 322560 bytes, 40320 bytes

Chapter 10

2. $192\,n + 6n\,log_2 n$

Chapter 11

6. 25 milliseconds
7. 4 megabytes, 252.5 milliseconds
8. 5 seconds, 1.4 seconds

Chapter 12

8. 400.185 MHz, 399.9814 MHz
9. 12.22 miles/second

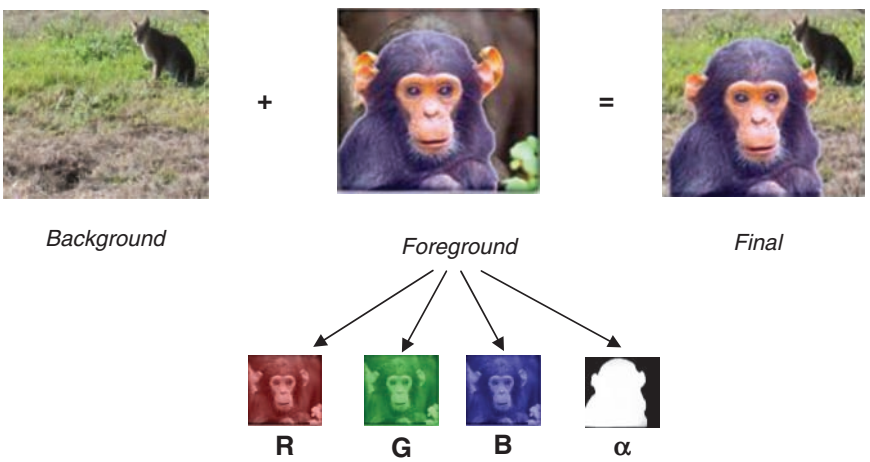
INDEX

Note: Page numbers in boldface indicate key terms.

J

N

Figure 3-1



Final[i][j] = Foreground[i][j] α [i][j] + Background[i][j]*(1–α[i][j])*

Figure 3-3

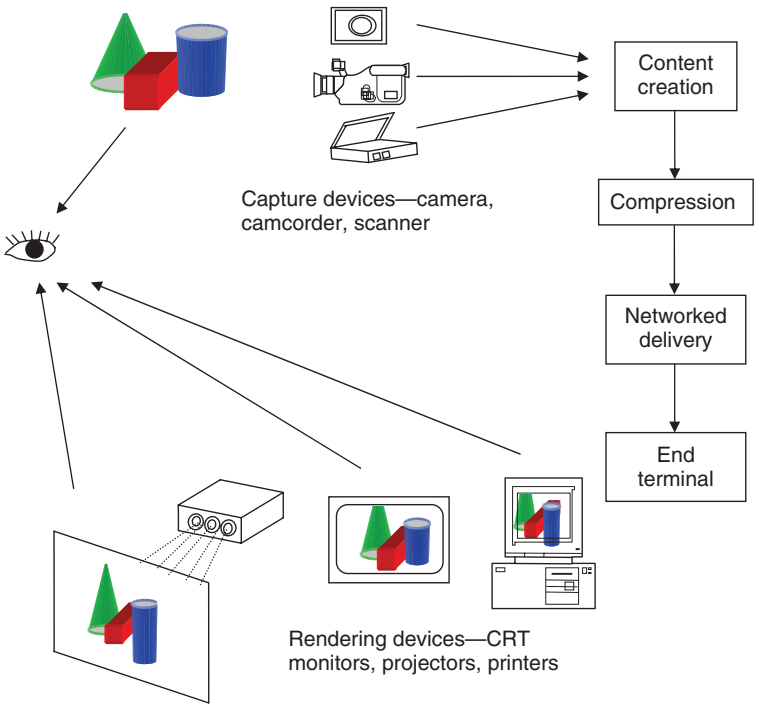


Figure 4-1

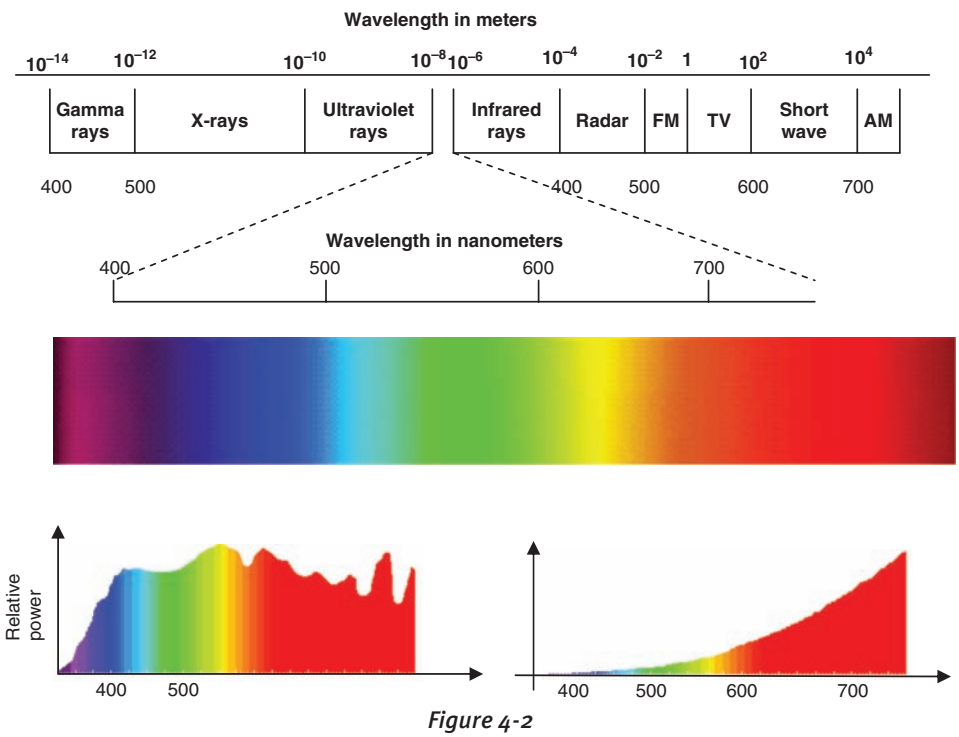


Figure 4-2

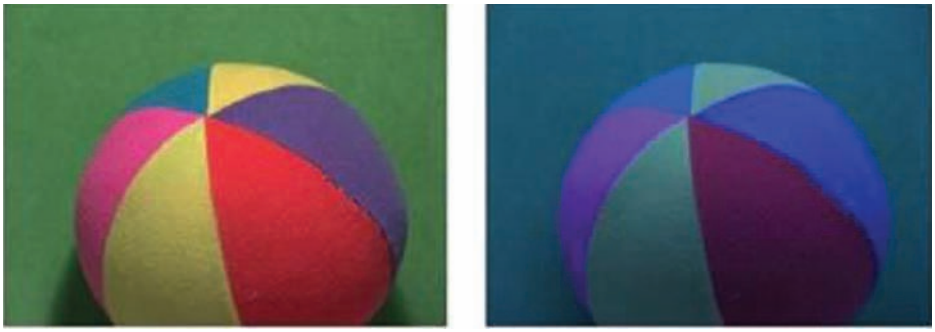
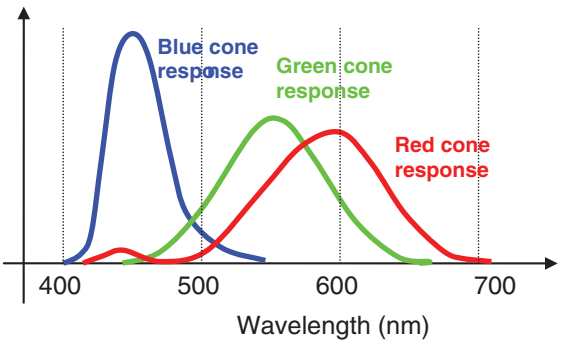


Image (a) Image (b)

Figure 4-4



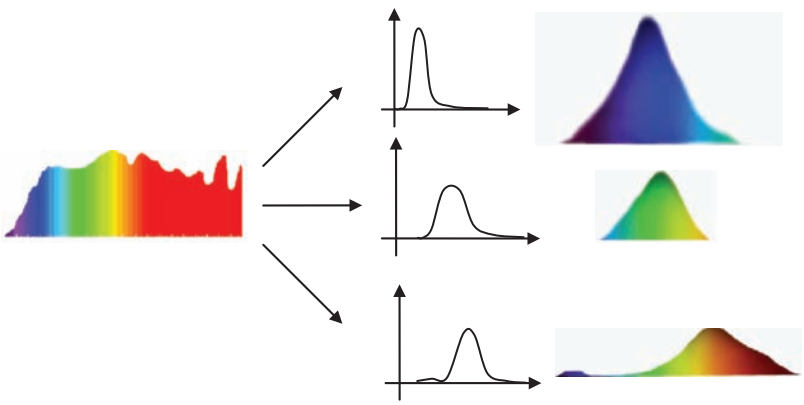
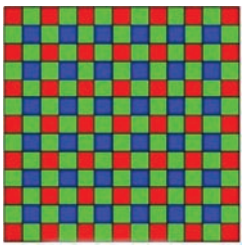
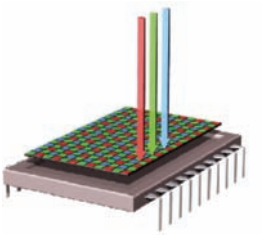


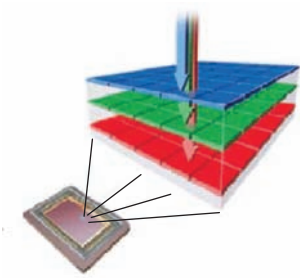
Figure 4-5



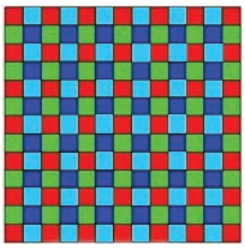
Bayer mosaic pattern



Bayer mosaic pattern
overlaid on a CCD chip



Fovean X3 technology



RGBE mosaic pattern

Figure 4-7

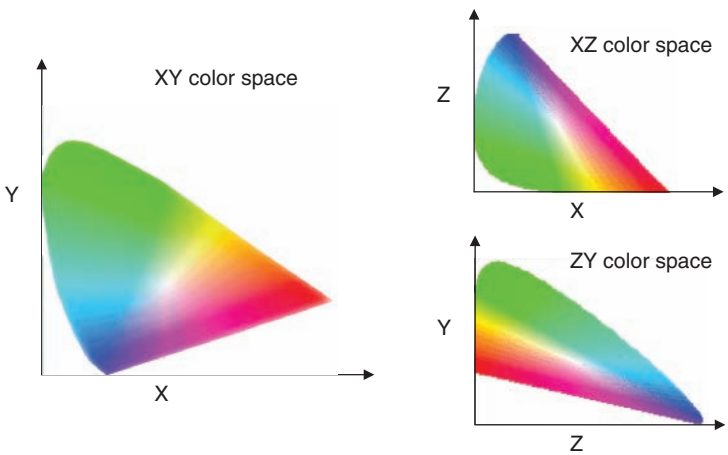


Figure 4-10

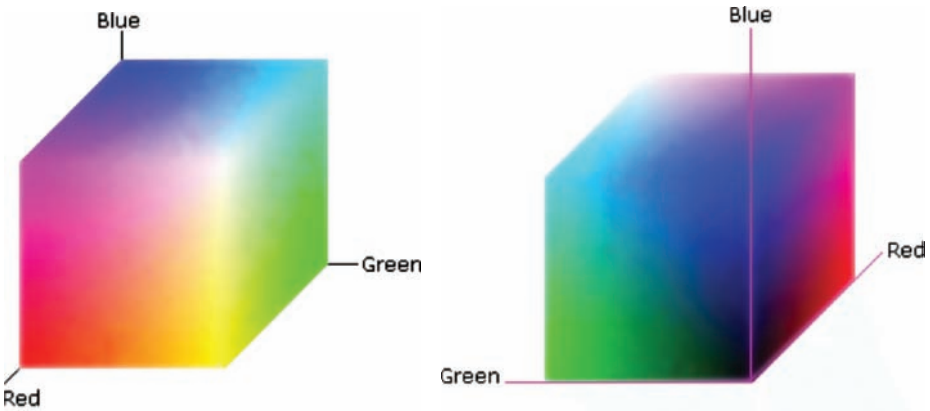


Figure 4-11

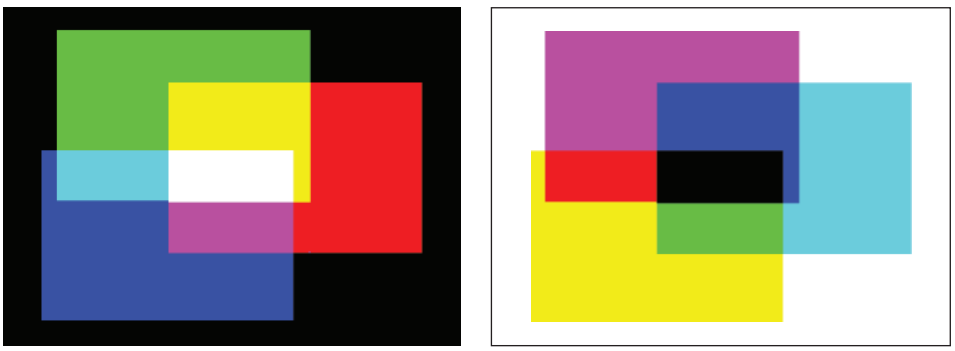


Figure 4-12

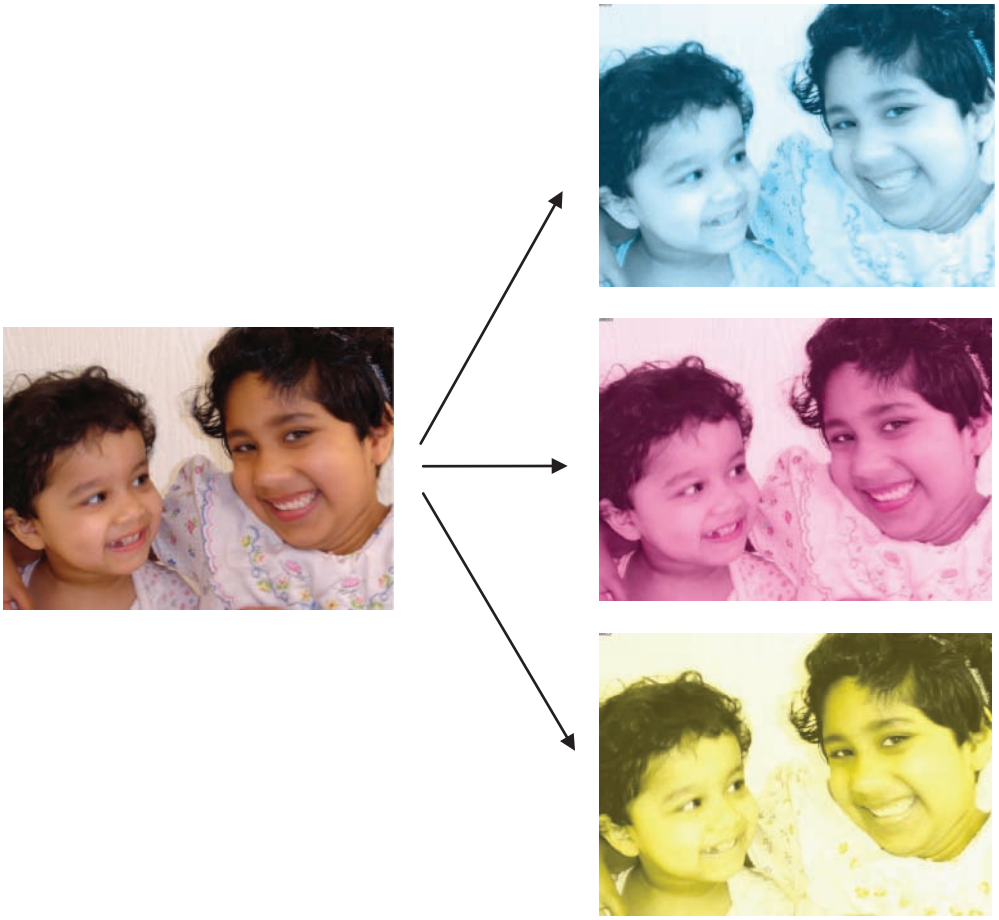
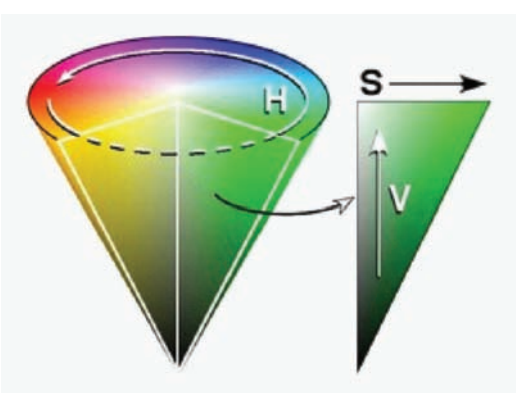


Figure 4-13



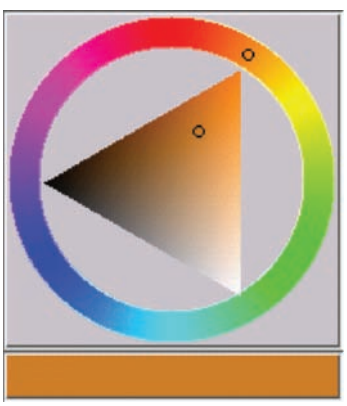


Figure 4-14

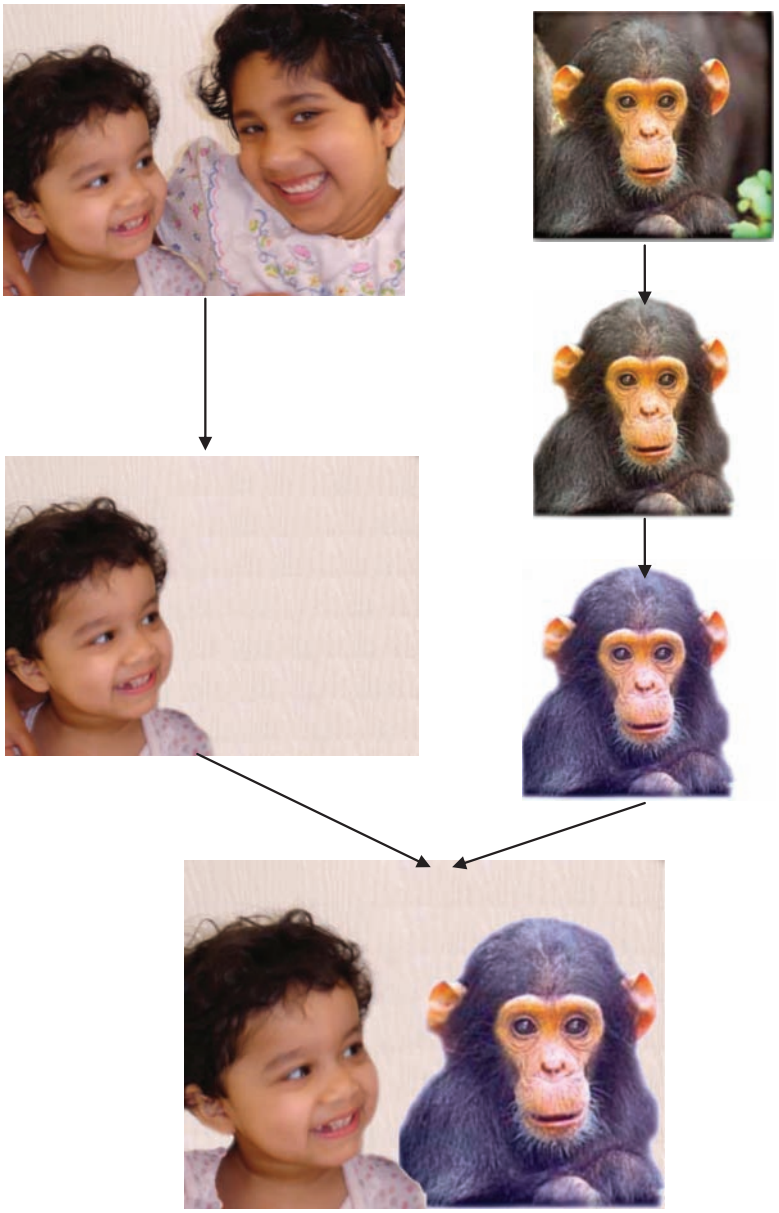


Figure 5-5

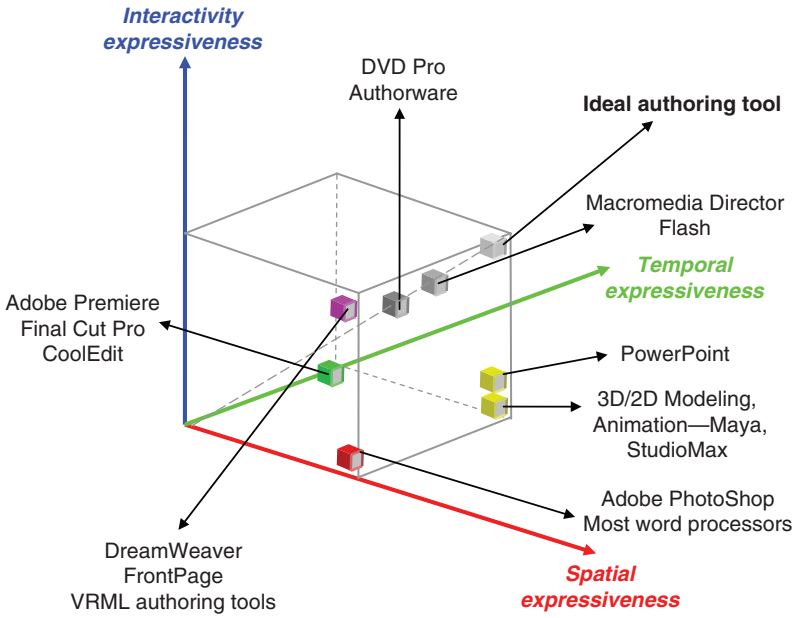

Interactivity expressiveness
DVD Pro Authorware
Ideal authoring tool
Macromedia Director Flash
Temporal expressiveness
Adobe Premiere
Final Cut Pro
CoolEdit
PowerPoint
3D/2D Modeling, Animation—Maya, StudioMax
Adobe PhotoShop
Most word processors
DreamWeaver
FrontPage
VRML authoring tools
Spatial expressiveness

Figure 5-8

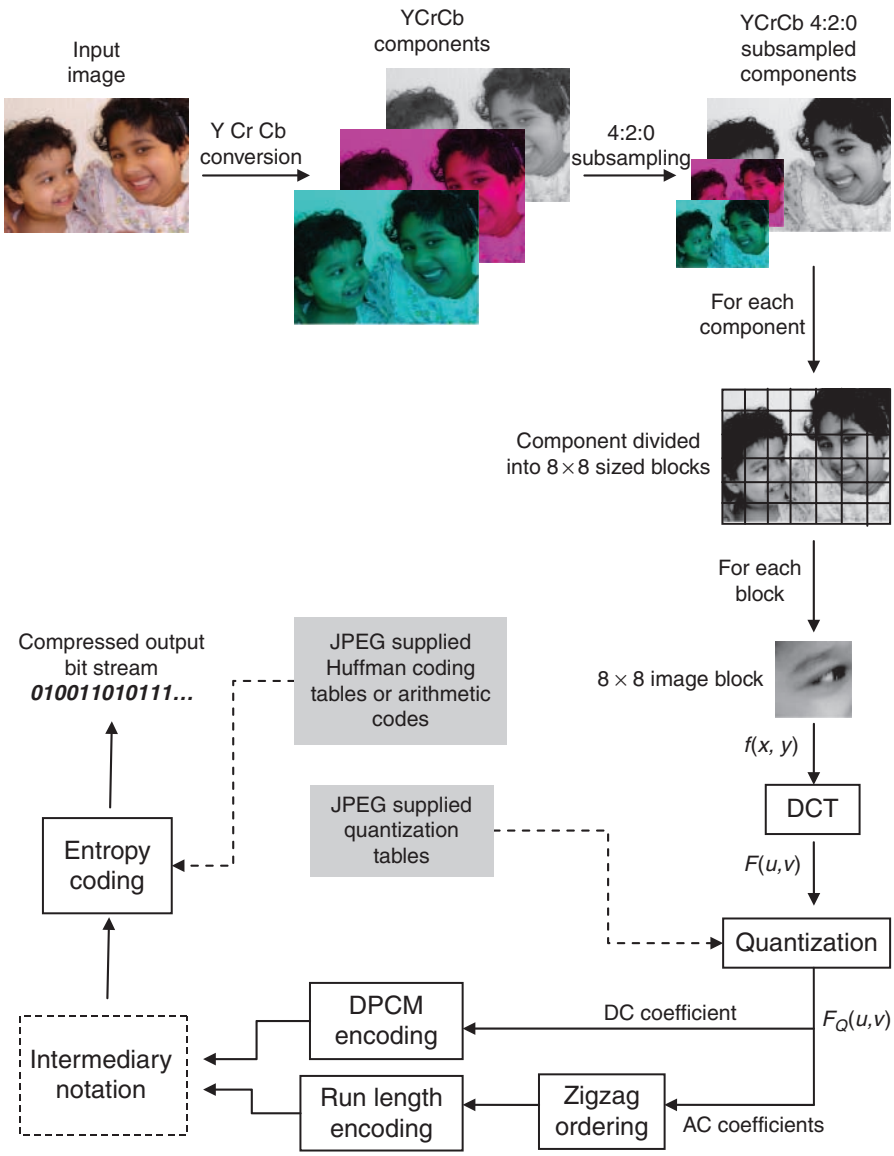

Input image
Y Cr Cb conversion
YCrCb components
4:2:0 subsampling
YCrCb 4:2:0 subsampled components
For each component
Component divided into 8×8 sized blocks
For each block
8 × 8 image block
f(x, y)
DCT
F(u,v)
Quantization
JPEG supplied quantization tables
JPEG supplied Huffman coding tables or arithmetic codes
Compressed output bit stream
010011010111...
Entropy coding
Intermediary notation
DPCM encoding
DC coefficient
$F_Q(u,v)$
Run length encoding
Zigzag ordering
AC coefficients

Figure 7-6

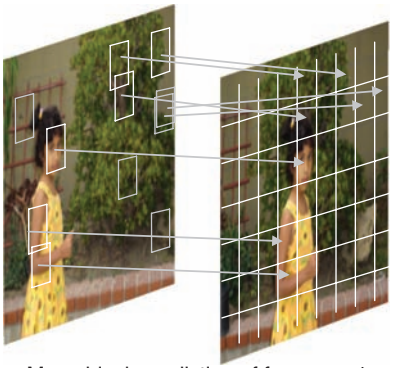
Frame n | Frame $n+1$ | Frame $(n+1)$ – Frame n

Frame n | Frame $n+1$ | Frame $(n+1)$ – Frame n

Figure 8-3

Frame n

Frame $n+1$



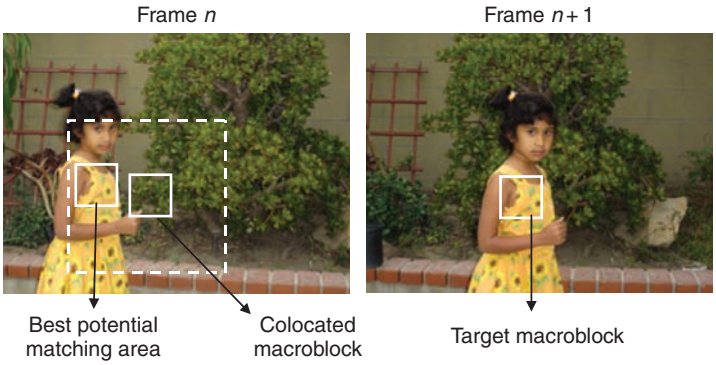
Macroblock prediction of frame $n+1$ from regions in frame n

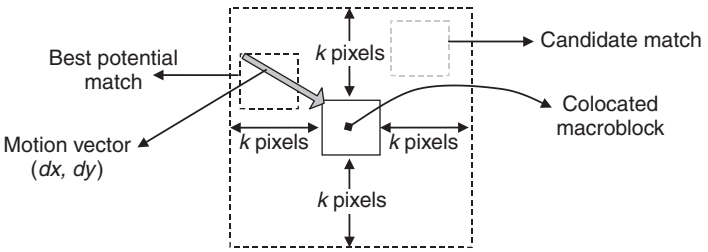
Reconstructed frame $n+1$ by macroblock prediction

Frame difference
frame $n+1$ - frame n

Frame difference
Reconstructed frame $n+1$ - frame $n+1$

Figure 8-5





Macroblock overlaid on search area in the colocated position

Macroblock overlaid at a candidate position

Macroblock overlaid at the best-matched position showing motion vector

Figure 8-7

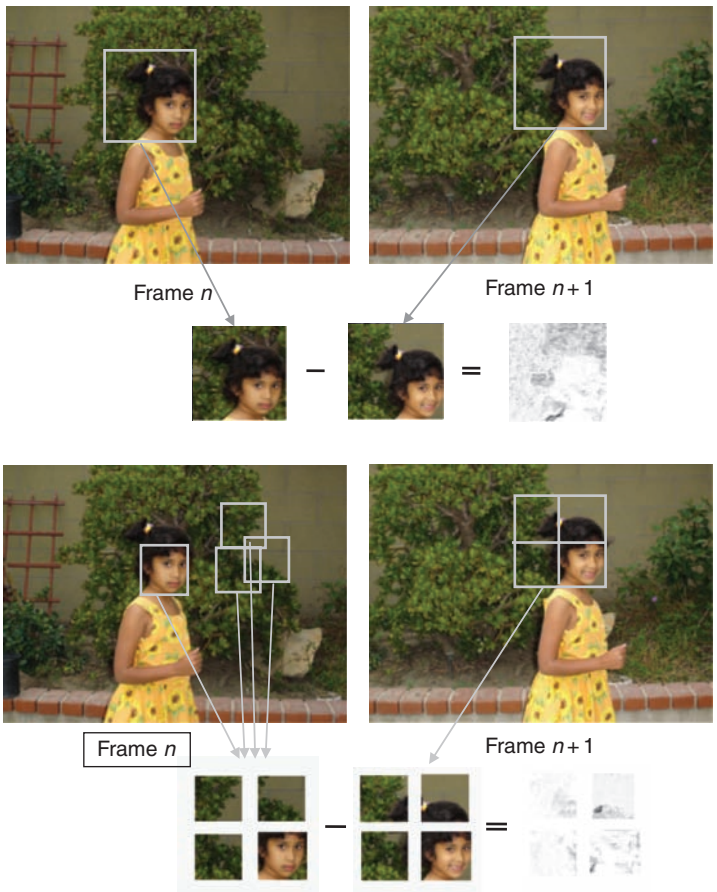
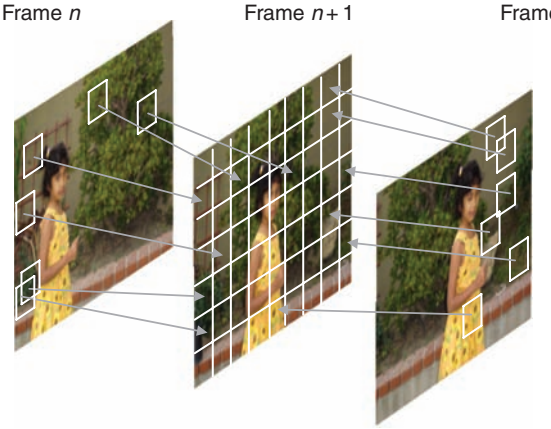


Figure 8-8

Frame n | Frame $n+1$ | Frame $n+2$



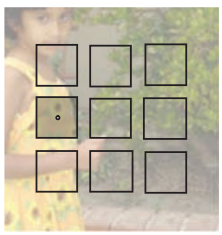
Reconstructed frame $n+1$ by macroblock prediction

Frame difference reconstructed frame $n+1$ - frame $n+1$

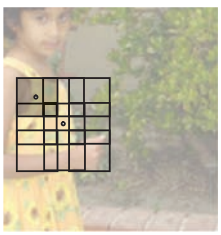
Figure 8-13

Search area around the colocation

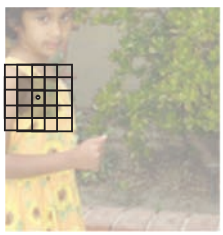
Macroblock of frame $n+1$



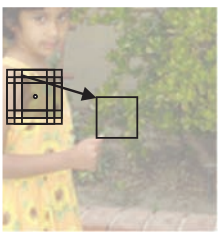
Iteration 1



Iteration 2



Iteration 3



Iteration 4 showing motion vector

Figure 8-19

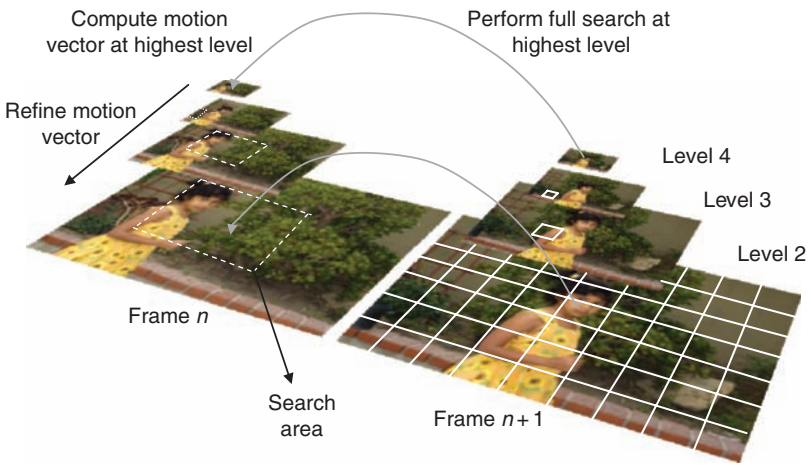


Figure 8-20

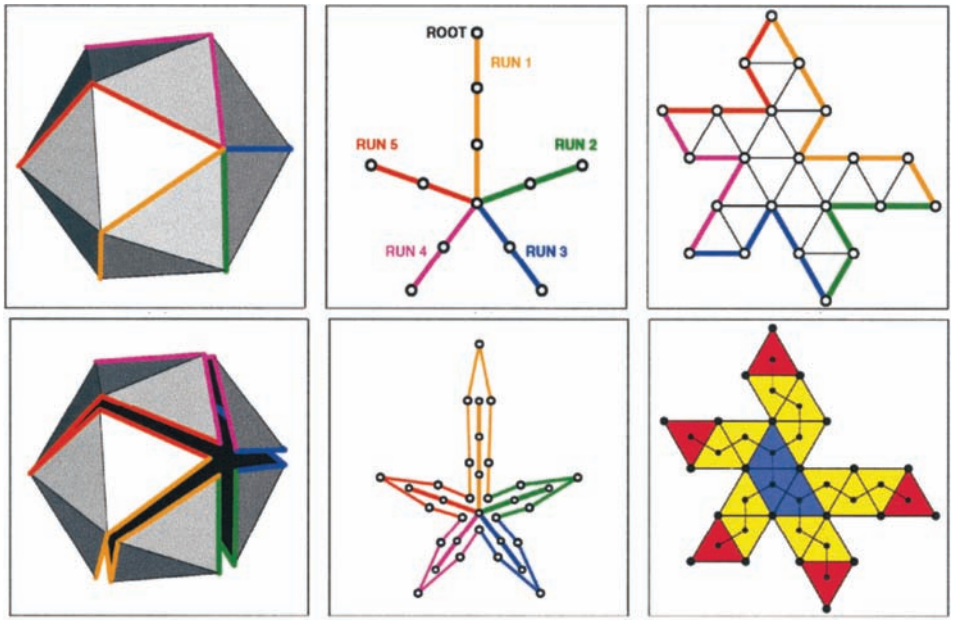


Figure 10-9

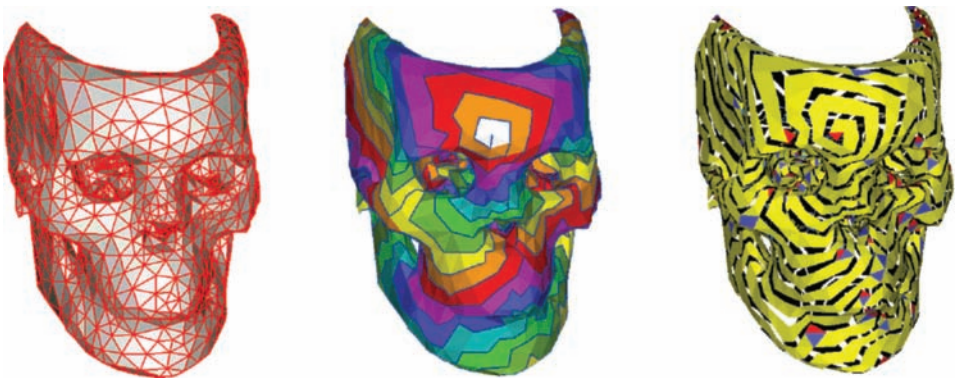


Figure 10-11