Video Coding for Mobile Communications

Efficiency, Complexity, and Resilience

Mohammed Ebrahim Al-Mualla
Etisalat College of Engineering,
Emirates Telecommunications Corporation (ETISALAT), U.A.E.

C. Nishan Canagarajah and David R. Bull
Image Communications Group,
Center for Communications Research,
University of Bristol, U.K.



ACADEMIC PRESS

An Elsevier Science Imprint

Amsterdam Boston London New York Oxford Paris San Diego
San Francisco Singapore Sydney Tokyo

SAMSUNG-1005

This book is printed on acid-free paper. ♾

Copyright © 2002, Elsevier Science (USA)
All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the publisher.

Requests for permission to make copies of any part of the work should be mailed to: Permissions Department, Harcourt, Inc., 6277 Sea Harbor Drive, Orlando, Florida 32887-6777.

Explicit permission from Academic Press in not required to reproduce a maximum of two figures or tables from an Academic Press chapter in another scientific or research publication provided that the material has not been credited to another source and that full credit to the Academic Press chapter is given.

Academic Press
An Elsevier Science Imprint
525 B Street, Suite 1900, San Diego, California 92101-4495, USA
http://www.academicpress.com

Academic Press
An Elsevier Science Imprint
Harcourt Place, 32 Jamestown Road, London NW1 7BY, UK
http://www.academicpress.com

Library of Congress Catalog Card Number: 2001098017

International Standard Book Number: 0-12-053079-1

PRINTED IN THE UNITED STATES OF AMERICA

02 03 04 05 06 EB 9 8 7 6 5 4 3 2 1

To my parents, brothers, and sisters
Mohammed E. Al-Mualla

To my family
C. Nishan Canagarajah

To Janice
David R. Bull

Preface
Scope and Purpose of the Book
Structure of the Bookce for the Book
Acknowledgments
About the Authors
List of Acronyms
1 Introduction to Mobile Video Communications
1.1 Motivations and Applications
1.2 Main Challenges
1.3Possible Solutions
Part I Introduction to Video Coding
2 Video Coding: Fundamentals
Overview
What Is Video?
Analog Video
Digital Video
Video Coding Basics
Intraframe Coding
Interframe Coding
3 Video Coding: Standards
Overview
The Need for Video Coding Standards
Chronological Development
The H.263 Standard
The MPEG-4 Standard
Part II Coding Efficiency
4 Basic Motion Estimation Techniques
Overview

Motion Estimation
Differential Methods
Pel-Recursive Methods
Frequency-Domain Methods
Block-Matching Methods
Efficiency of Block Matching at Very Low Bit Rates
Discussion

5 Warping-Based Motion Estimation Techniques

Overview
Warping-Based Methods: A Review
Efficiency of Warping-Based Methods at Very Low Bit Rates
Discussion

6 Multiple-Reference Motion Estimation Techniques

Overview
Multiple-Reference Motion Estimation: A Review
Long-Term Memory Motion-Compensated Prediction
Discussion

Part III Computational Complexity

7 Reduced-Complexity Motion Estimation Techniques

Overview
The Need for Reduced-Complexity Motion Estimation
Techniques Based on a Reduced Set of Motion Vector Candidates
Techniques Based on a Reduced-Complexity Block Distortion Measure
Techniques Based on a Subsampled Block-Motion Field
Hierarchical Search Techniques
Fast Full- Search Techniques
A Comparative Study
Discussion

8 The Simplex Minimization Search
Overview
Block Matching: An Optimization Problem
The Simplex Minimization (SM) Optimization Method
The Simplex Minimization Search (SMS)
Simulation Results
Simplex Minimization for Multiple-Reference Motion Estimation
Discussion

Part IV Error Resilience

9 Error-Resilience Video Coding Techniques
Overview
A Typical Video Communication System
Types of Errors
Effects of Errors
Error Detection
Forward Techniques
Postprocessing or Concealmentl Techniques
Interactive Techniques
Discussion

10 Error Concealment Using Motion Field Interpolation
Overview
Temporal Error Concealment Using Motion Field Interpolation (MFI)
Temporal Error Concealment Using a Combined BM-MFI Technique
Simulation Results
Temporal Error Concealment for Multiple-Reference Motion-Compensated Prediction
Discussion

Appendix Fast Block-Matching Algorithms
A.1 Notation and Assumptions

A.2 The Two-Dimensional Logarithmic (TDL) Search .....

A.3 The N -Steps Search (NSS)

A.4 The One-at-a-Time Search (OTS)

A.5 The Cross-Search Algorithm (CSA)

A.6 The Diamond Search (DS)

Bibliography .....

Signal Processing and its Applications

SERIES EDITORS

Dr Richard Green
Department of Technology, Metropolitan Police Service, London, UK

Professor Truong Nguyen
Electrical and Computer Engineering Department,
University of Wisconsin, Wisconsin, USA

EDITORIAL BOARD

Professor Manrice G. Bellanger
CNAM, Paris, France

Professor David Bull
Department of Electrical and Electronic Engineering,
University of Bristol, UK

Professor Gerry D. Cain
School of Electronic and Manufacturing System Engineering,
University of Westminster, London, UK

Professor Culin Cowan
Department of Electronics and Electrical Engineering,
Queen's University, Belfast, Northern Ireland

Professor Roy Davies
Machine Vision Group, Department of Physics,
Royal Holloway, University of London, Surrey, UK

Dr Paola Hobson
Motorola, Basingstoke, UK

Professor Mark Sandlar
Department of Electronics and Electrical Engineering,
King's College London, University of London, UK

Dr Henry Stark
Electrical and Computer Engineering Department,
Illinois Institute of Technology, Chicago, USA

Dr Manecchl Trivedi
Horndean, Waterlooville, UK

Preface

Scope and Purpose of the Book

Motivated by the vision of being able to communicate from anywhere at any time with any type of information, a natural convergence of mobile and multimedia is under way. This new area, called *mobile multimedia communications*, is expected to achieve unprecedented growth and worldwide commercial success.

Current second-generation mobile communication systems support a number of basic multimedia communication services. However, many technologically demanding problems need to be solved before *real-time mobile video communications* can be achieved. When such challenges are resolved, a wealth of advanced services and applications will be available to the mobile user. This book concentrates on three main challenges:

1. Higher coding efficiency
2. Reduced computational complexity
3. Improved error resilience

Mobile video communications is an interdisciplinary subject. Complete systems are likely to draw together solutions from different areas, such as video source coding, channel coding, network design, and semiconductor design, among others. This book concentrates on solutions based on *video source coding*. In this context, the book adopts a *motion-based approach*, where advanced motion estimation techniques, reduced-complexity motion estimation techniques, and motion-compensated error concealment techniques are used as possible solutions to the three challenges, respectively.

The idea of this book originated in 1997, when the first author was in the early stages of his Ph.D. studies. As a newcomer to the field, he started consulting a number of books to introduce himself to the fundamentals and standards of video source coding. He realized, however, that, for a beginner, most of these books seemed too long and too theoretical, with no treatment of some important practical and implementation issues. As he progressed further

in his studies, the first author also realized that the areas of coding efficiency, computational complexity, and error resilience are usually treated separately. Thus, he always wished there was a book that provided a quick, easy, and practical introduction to the fundamentals and standards of video source coding and that brought together the areas of coding efficiency, computational complexity, and error resilience in a single volume. This is exactly the purpose of this book.

Structure of the Book

The book consists of 10 chapters. Chapter 1 gives a brief introduction to mobile video communications. It starts by discussing the main motivations and applications of mobile video communications. It then briefly introduces the challenges of higher coding efficiency, reduced computational complexity, and error resilience. The chapter then discusses some possible motion-based solutions. The remaining chapters of the book are organized into four parts. The first part introduces the reader to video coding, whereas the remaining three parts are devoted to the three challenges of coding efficiency, computational complexity, and error resilience.

Part I gives an introduction to video coding. It contains two chapters. Chapter 2 introduces some of the fundamentals of video source coding. It starts by giving some basic definitions and then covers both analog and digital video along with some basic video coding techniques. It also presents the performance measures and the test sequences that will be used throughout the book. It then reviews both intraframe and interframe video coding methods.

Chapter 3 provides a brief introduction to video coding standards. Particular emphasis is given to the most recent standards, such as H.263 (and its extensions H.263+ and H.263++) and MPEG-4.

Part II concentrates on coding efficiency. It contains three chapters. Chapter 4 covers some basic motion estimation methods. It starts by introducing some of the fundamentals of motion estimation. It then reviews some basic motion estimation methods, with particular emphasis on the widely used block-matching methods. The chapter then presents the results of a comparative study between the different methods. The chapter also investigates the efficiency of motion estimation at very low bit rates, typical of mobile video communications. The aim is to decide if the added complexity of this process is justifiable, in terms of an improved coding efficiency, at such bit rates.

Chapter 5 investigates the performance of the more advanced warping-based motion estimation methods. The chapter starts by describing a general warping-based motion estimation method. It then considers some important parameters, such as the shape of the patches, the spatial transformation used, and the node-

tracking algorithm. The chapter then assesses the suitability of warping-based methods for mobile video communications. In particular, the chapter compares the efficiency and complexity of such methods to those of block-matching methods.

Chapter 6 investigates the performance of another advanced motion-estimation method, called multiple-reference motion-compensated prediction. The chapter starts by briefly reviewing multiple-reference motion estimation methods. It then concentrates on the long-term memory motion-compensated prediction technique. The chapter investigates the prediction gains and the coding efficiency of this technique at very low bit rates. The primary aim is to decide if the added complexity, increased motion overhead, and increased memory requirements of this technique are justifiable at such bit rates. The chapter also investigates the properties of multiple-reference block-motion fields and compares them to those of single-reference fields.

Part III of the book considers the challenge of reduced computational complexity. It contains two chapters. Chapter 7 reviews reduced-complexity motion estimation techniques. The chapter uses implementation examples and profiling results to highlight the need for reduced-complexity motion estimation. It then reviews some of the main reduced-complexity block-matching motion estimation techniques. The chapter then presents the results of a study comparing the different techniques.

Chapter 8 gives an example of the development of a novel reduced-complexity motion estimation technique. The technique is called the *simplex minimization search*. The development process is described in detail, and the technique is then tested within an isolated test environment, a block-based H.263-like codec, and an object-based MPEG-4 codec. In an attempt to reduce the complexity of multiple-reference motion estimation (investigated in Chapter 6), the chapter extends the simplex minimization search technique to the multiple-reference case. The chapter presents three different extensions (or algorithms) representing different degrees of compromise between prediction quality and computational complexity.

Part IV concentrates on error resilience. It contains two chapters. Chapter 9 reviews error resilience video coding techniques. The chapter considers the types of errors that can affect a video bitstream and examines their impact on decoded video. It then describes a number of error detection and error control techniques. Particular emphasis is given to standard error resilience techniques included in the recent H.263+, H.263++, and MPEG-4 standards.

Chapter 10 gives examples of the development of error-resilience techniques. The chapter presents two temporal error concealment techniques. The first technique is based on motion field interpolation, whereas the second technique uses multihypothesis motion compensation to combine motion field interpolation with a boundary-matching technique. The techniques are then

tested-within-both-an-isolated-test-environment-and-an-H.263-codec.-The-chapter-also-investigates-the-performance-of-different-temporal-error-concealment-techniques-when-incorporated-within-a-multiple-reference-video-codec.-In-particular,-the-chapter-finds-a-combination-of-techniques-that-best-recovers-the-spatial-temporal-components-of-a-damaged-multiple-reference-motion-vector.-In-addition,-the-chapter-develops-a-multihypothesis-temporal-concealment-technique-to-be-used-with-multiple-reference-systems.-

Audience for the Book

In-recent-years,-mobile-video-communications-has-become-an-active-and-important-research-and-development-topic-in-both-industry-and-academia.-It-is,-therefore,-hoped-that-this-book-will-appeal-to-a-broad-audience,-including-students,-instructors,-researchers,-engineers,-and-managers.-

Chapter-1-can-serve-as-a-quick-introduction-for-managers.-Chapters-2-and-3-can-be-used-in-an-introductory-course-on-the-fundamentals-and-the-standards-of-video-coding.-The-two-chapters-can-also-be-used-as-a-quick-introduction-for-researchers-and-engineers-working-on-video-coding-for-the-first-time.-More-advanced-courses-on-video-coding-can-also-utilize-Chapters-4,-7,-and-9-to-introduce-the-students-to-issues-in-coding-efficiency,-computational-complexity,-and-error-resilience.-The-three-chapters-can-also-be-used-by-researchers-and-engineers-as-an-introduction-and-a-guide-to-the-relevant-literature-in-the-respective-areas.-Researchers-and-engineers-will-also-find-Chapters-5,-6,-8,-and-10-useful-as-examples-of-the-design,-implementation,-and-testing-of-novel-video-coding-techniques.-

Acknowledgments

We-are-greatly-indebted-to-past-and-present-members-of-the-Image-Communications-Group-in-the-Center-for-Communications-Research,-University-of-Bristol,-for-creating-an-environment-from-which-a-book-such-as-this-could-emerge.-In-particular,-we-would-like-to-thank-Dr.-Przemysław-Czerepiński-for-his-generous-help-in-all-aspects-of-video-research,-Dr.-Greg-Cain-for-interesting-discussions-on-implementation-and-complexity-issues-of-motion-estimation,-Mr.-Chiew-Tuan-Kiang-for-fruitful-discussions-on-the-efficiency-of-motion-estimation-at-very-low-bit-rates,-and-Mr.-Oliver-Sohm-for-providing-the-MPEG-4-results-of-Chapter-8.-

We-also-owe-a-debt-to-Joel-ClayPool-and-Angela-Dooley-from-Academic-press-who-have-shown-a-great-deal-of-patience-with-us-while-we-pulled-this-project-together.-

Mohammed Al-Mualla is deeply appreciative of the Emirates Telecommunications Corporation (Etisalat), United Arab Emirates, for providing financial support. In particular, he would like to thank Mr. Ali Al-Owais, CEO and President of Etisalat, and Mr. Salim Al-Owais, Manager of Etisalat College of Engineering.

He would also like to give his special thanks to his friends: Saif Bin Haider, Khalid Almidfa, Humaid Gharib, Abdul-Rahim Al-Maimani, Dr. Naim Dahnoun, Dr. M. F. Tariq, and Doreen Castañeda.

He would also like to give his most heartfelt thanks to his family and especially to his parents and his dear brother, Majid, for their unwavering love, support, and encouragement.

Mohammed Ebrahim Al-Mualla
Umm Al-Quwain, U.A.E.

C. Nishan Canagarajah,
David R. Bull
Bristol, U.K.

About the Authors

Mohammed E. Al-Mualla received the B. Eng. (Honors) degree in communications from Etisalat College of Engineering, Sharjah, United Arab Emirates (U.A.E.), in 1995, and the M.Sc. degree in communications and signal processing from the University of Bristol, U.K., in 1997. In 2000, he received the Ph.D. degree from the University of Bristol with a thesis titled "Video Coding for Mobile Communications: A Motion-Based Approach." Dr. Al-Mualla is currently an Assistant Professor at Etisalat College of Engineering. His research is focused on mobile video communications, with a particular emphasis on the problems of higher coding efficiency, reduced computational complexity, and improved error resilience.

C. Nishan Canagarajah received the B.A. (Honors) degree and the Ph.D. degree in digital signal processing techniques for speech enhancement, both from the University of Cambridge, Cambridge, U.K. He is currently a reader in signal processing at the University of Bristol, Bristol, U.K. He is also a coeditor of the book *Insights into Mobile Multimedia Communications* in the Signal Processing and Its Applications series from Academic Press. His research interests include image and video coding, nonlinear filtering techniques, and the application of signal processing to audio and medical electronics. Dr. Canagarajah is a committee member of the IEE Professional Group E5, and an Associate Editor of the IEE Electronics and Communication Journal.

David R. Bull is currently a professor of signal processing at the University of Bristol, Bristol, U.K., where he is the head of the Electrical and Electronics Engineering Department. He also leads the Image Communications Group in the Center for Communications Research, University of Bristol. Professor Bull has worked widely in the fields of 1- and 2-D signal processing. He has published over 150 papers and is a coeditor of the book *Insights into Mobile Multimedia Communications* in the Signal Processing and Its Applications series from Academic Press. Previously, he was an electronic systems engineer at Rolls Royce and a lecturer at the University of Wales, Cardiff, U.K. His recent research has focused on the problems of image and video communications, in particular, error-resilient source coding, linear and nonlinear filterbanks, scalable methods, content-based coding, and architectural

optimization. Professor Bull is a member of the EPSRC Communications College and the Program Management Committee for the DTI/EPSRC LINK program in broadcast technology. Additionally, he is a member of the U.K. Foresight ITEC panel.

List of Acronyms

525/60	a television system with 525 vertical lines and a refresh rate of 60 Hz
625/50	a television system with 625 vertical lines and a refresh rate of 50 Hz
ACK	acknowledgment
ARQ	automatic repeat request
ASO	arbitrary slice ordering
ATM	asynchronous transfer mode
AV	average motion vector
AVO	audio visual object
BAB	binary alpha block
BCH	Bose-Chaudhuri-Hocquenghem
BDM	block distortion measure
BER	bit error rate
BM	boundary matching
BMA	block-matching algorithm
BMME	block-matching motion estimation
CAE	context-based arithmetic encoding
CCIR	international consultative committee for radio
CCITT	international telegraph and telephone consultative committee
CD	compact disc; conjugate directions

CDS	conjugate-directions search
CGI	control grid interpolation
CIF	common intermediate format
CMY	cyan, magenta, and yellow
CR	conditional replenishment
CSA	cross-search algorithm
DCT	discrete cosine transform
DF	displaced frame
DFA	differential frame algorithm
DFD	displaced-frame difference
DFT	discrete Fourier transform
DM	delta modulation
DMS	discrete memoryless source
DPCM	differential pulse code modulation
DS	dynamic sprites; diamond search
DSCQS	double stimulus continuous quality scale
DSIS	double stimulus impairment scale
DSP	digital signal processor
DWT	discrete wavelet transform
ECVQ	entropy-constrained vector quantization
EDGE	enhanced data rates for GSM evolution
EREC	error-resilience entropy code
ERPS	enhanced reference picture selection
EZW	embedded zero-tree wavelet
FD	frame difference
FEC	forward error correction
FFT	fast Fourier transform
FLC	fixed-length coding

FS	full search
FT	Fourier transform
GA	genetic algorithm
GMC	global motion compensation
GMS	genetic motion search
GOB	group of blocks
GOV	group of video object planes
GPRS	general packet radio service
GSM	group spécial mobile or global system for mobile
HD	horizontal difference
HDTV	high-definition television
HEC	header extension code
HMA	hexagonal matching algorithm
HVS	human visual system
IDCT	inverse discrete cosine transform
IEC	international electrotechnical commission
ISD	independent segment decoding
ISDN	integrated services digital network
ISO	international organization for standardization
ITU-R	international telecommunications union — radio sector
ITU-T	international telecommunications union — telecommunication standardization sector
JTC	joint technical committee
KLT	Krahunen-Loève transform
LBG	Linde-Buzo-Gray
LMS	least mean square
LOT	lapped orthogonal transform
LPE	low-pass extrapolation
LTM-MCP	long-term memory motion-compensated prediction

MAE	mean absolute error
MAP	maximum *a posteriori* probability
MB	macroblock
MC	motion compensation
MCP	motion-compensated prediction
ME	motion estimation
MFI	motion field interpolation
MHMC	multihypothesis motion compensation
MIPS	million instructions per second
ML	maximum likelihood
MPEG	moving picture experts group
MRF	Markov random field
MR-MCP	multiple-reference motion-compensated prediction
MSE	mean squared error
NACK	negative acknowledgment
NCCF	normalized cross-correlation function
NSS	*n*-steps search
NTSC	national television system committee
OMC	overlapped motion compensation
OTS	one-at-a-time search
PAL	phase alternation line
PCA	phase correlation algorithm
PDC	pel difference classification
PDE	partial distortion elimination
PRA	pel-recursive algorithm
PSNR	peak signal-to-noise ratio
PSTN	public switched telephone network

QAM	quadrature amplitude modulation
QCIF	quarter common intermediate format
QMF	quadrature mirror filter
QP	quantization parameter
QSIF	quarter source input format
RBMAD	reduced-bits mean absolute difference
R-D	rate-distortion
RGB	red, green, and blue
RLE	run-length encoding
RPS	reference picture selection
RS	rectangular slice
RVLC	reversible variable-length coding
SAD	sum of absolute differences
SC	subcommittee
SEA	successive elimination algorithm
SECAM	sequential couleur avec memoire
SG	study group
SIF	source input format
SM	simplex minimization
SMD	side-match distortion
SMS	simplex minimization search
SNR	signal-to-noise ratio
SPIHT	set partitioning in hierarchical trees
SQCIF	sub-QCIF
SSCQS	single stimulus continuous quality scale
SSD	sum of squared differences
STFM/LTFM	short-term frame memory/long-term frame memory

TDL	two-dimensional logarithmic
TMN	test model near-term
TR	temporal replacement
TSS	three-steps search
TV	television
UMTS	universal mobile telecommunication system
VD	vertical difference
VLC	variable-length coding
VO	video object
VOL	video object layer
VOP	video object plane; visual object plane
VQ	vector quantization
VS	video session
WBA	warping-based algorithm
WG	working group
WHT	Walsh-Hadamard transform

Chapter 1

Introduction to Mobile Video Communications

1.1 Motivations and Applications

In recent years, two distinct technologies have experienced massive growth and commercial success: *multimedia* and *mobile communications*. With the increasing reliance on the availability of multimedia information and the increasing mobility of individuals, there is a great need for providing multimedia information on the move. Motivated by this vision of being able to communicate from anywhere at any time with any type of information, a natural convergence of mobile and multimedia is under way. This new area is called *mobile multimedia communications*.

Mobile multimedia communications is expected to achieve unprecedented growth and worldwide success. For example, in Western Europe alone, it is estimated that by the year 2005 about 32 million people will use mobile multimedia services, representing a market segment worth 24 billion Euros per year and generating 3,800 million Mbytes of traffic per month. This will correspond, respectively, to 16% of all mobile users, 23% of the total revenues, and 60% of the overall traffic. Usage is expected to increase at even higher rates, with 35% of all mobile users having mobile multimedia services by the year 2010 [1]. The estimates become even more impressive when put in the context of a worldwide mobile market that reached 331.5 million users by the end of June 2000 [2] and is expected to grow to 1.7 billion users by 2010 [1]. It is not surprising, therefore, that this area has become an active and important research and development topic for both industry and academia, with groups across the world working to develop future mobile multimedia systems.

The definition of the term *multimedia* has always been a source of great debate and confusion. In this book, it refers to the presentation of information through multiple forms of media. This includes textual media (text, style,

layout, etc.), numerical media (spreadsheets, databases, etc.), audio media (voice, music, etc.), visual media (images, graphics, video, etc.) and any other form of information representation.

Current second-generation mobile communication systems, like the *Global System for Mobile* (GSM),[1] already support a number of basic multimedia communication services. Examples are voice, basic fax/data, short message services, information-on-demand (e.g., sports results, news, weather), e-mail, still-image communication, and basic internet access. However, many technologically demanding problems need to be solved before real-time *mobile video communication* can be achieved. When such challenges are resolved, a wealth of advanced services and applications will be available to the mobile user. Examples are:

- Video-on-demand.
- Distance learning and training.
- Interactive gaming.
- Remote shopping.
- Online media services, such as news reports.
- Videotelephony.
- Videoconferencing.
- Telemedicine for remote consultation and diagnosis.
- Telesurveillance.
- Remote consultation or scene-of-crime work.
- Collaborative working and telepresence.

1.2 Main Challenges

The primary focus of this book is *mobile video communication*. In particular, the book focuses on three main challenges:

1. **Higher coding efficiency**. The radio spectrum is a very limited and scarce resource. This puts very stringent limits on the bandwidth available for a mobile channel. Given the enormous amount of data generated

[1] Originally, GSM was an acronym for *Group Spécial Mobile*.

by video, the use of efficient coding techniques is vital. For example, real-time transmission of a CIF[2] video at 15 frames/s over a 9.6 kbits/s GSM channel requires a compression ratio of about 1900:1. Although current coding techniques are capable of providing such compression ratios, there is a need for even higher coding efficiency to improve the quality (i.e., larger formats, higher frame rates, and better visual quality) of video at such very low bit rates. This continues to be the case even with the introduction of enhancements to second-generation systems, like the *General Packet Radio Service* (GPRS) [3] and the *Enhanced Data Rates for GSM Evolution* (EDGE), and also with the deployment of future higher-capacity, third-generation systems, like the *Universal Mobile Telecommunication System* (UMTS) [4].

2. **Reduced computational complexity**. In mobile terminals, processing power and battery life are very limited and scarce resources. Given the significant amount of computational power required to process video, the use of reduced-complexity techniques is essential. For example, recent implementations of video codecs [5,6] indicate that even state-of-the-art *digital signal processors* (DSPs) cannot, yet, achieve real-time video encoding. Typical results quoted in Refs. 5 and 6 are 1–5 frames/s using small video formats like SQCIF and QCIF.[3]

3. **Improved error resilience**. The mobile channel is a hostile environment with high bit error rates caused by a number of loss mechanisms, like multipath fading, shadowing, and co-channel interference. In the case of video, the effects of such errors are magnified due to the fact that the video bitstream is highly compressed to meet the stringent bandwidth limitations. In fact, the higher the compression is, the more sensitive the bitstream is to errors, since in this case each bit represents a larger amount of decoded video. The effects of errors on video are also magnified by the use of predictive coding and variable-length coding (VLC). The use of such coding methods can lead to temporal and spatial error propagation. It is, therefore, not difficult to realize that when transmitted over a mobile channel, compressed video can suffer severe degradation and the use of error-resilience techniques is vital.

[2]CIF stands for *Common Intermediate Format*. It is a digital video format in which the luminance component is represented by 352 pels $\times$ 288 lines and the two chrominance components are each of dimensions 176 $\times$ 144, where each pel is usually represented by 8 bits. Digital video formats are discussed in more detail in Chapter 2.

[3]*Quarter-CIF* (QCIF) has a luminance component of 176 $\times$ 144, whereas *sub-QCIF* (SQCIF) has a luminance component of 128 $\times$ 96.

It should be emphasized that those are not the only requirements of a mobile video communication system. Requirements like low delay, interactivity, scalability, and security are equally important.

1.3 Possible Solutions

Mobile video communication is a truly interdisciplinary subject [7]. Complete systems are likely to draw together solutions from different areas, like video source coding, channel coding, network design, semiconductor design, and others. This book will concentrate on solutions based on the *video source coding* part of the area. Thus, before being able to present the adopted approach, a closer look at video source coding is in order. Figure 1.1 shows a typical video codec. Changes between consecutive frames of a video sequence are mainly due to the movement of objects. Thus, the *motion estimation* (ME) block uses a motion model to estimate the movement that occurred between the current frame and a reference frame (usually a previously decoded frame that was stored in a frame buffer). This motion information is then utilized by the *motion compensation* (MC) block to move the contents of the reference frame to provide a prediction of the current frame. This *motion-compensated prediction* (MCP) is also known as the *displaced frame* (DF). The prediction is then subtracted from the current frame to produce an error signal known as

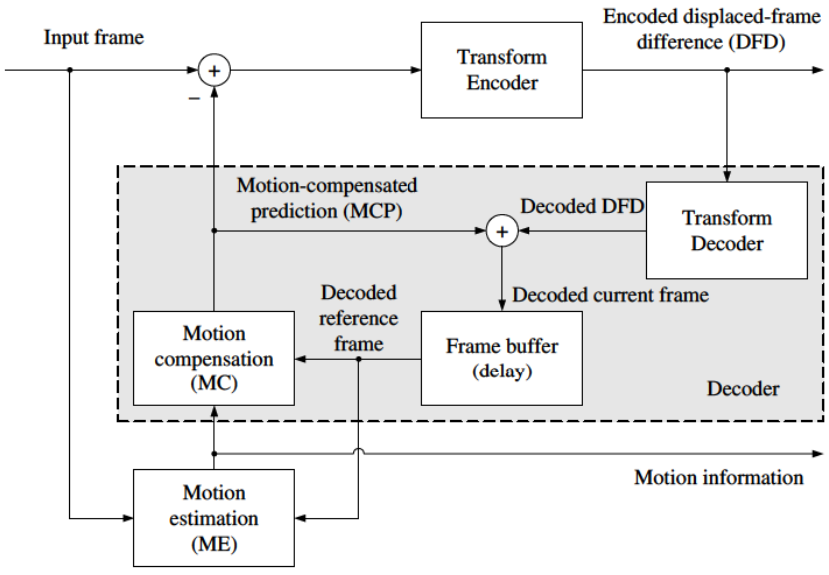


Figure 1.1: Typical video codec

the *displaced-frame difference* (DFD). Instead of encoding the current frame itself, this error signal is encoded, since it has a much reduced entropy. At the decoder, the same reference frame is used along with the received motion information to produce the same prediction. This prediction is then added to the received error signal to reconstruct the current frame.

Careful examination of this codec (as will be detailed in subsequent chapters) reveals that a *motion-based approach* can be adopted to provide suitable solutions for the three challenges of higher coding efficiency, reduced complexity, and error resilience. This motion-based approach can be summarized as follows:

1. **Advanced motion estimation techniques**. One way to achieve higher coding efficiency is to improve the performance of the motion estimation and compensation processes. The aim is to produce a better motion-compensated prediction and consequently reduce the entropy of the DFD signal. This should be achieved at the same or, preferably, a reduced motion overhead.[4]

2. **Reduced-complexity motion estimation techniques**. Motion estimation is the most computationally intensive process in a typical video codec. In fact, profiling results (as will be shown in Chapter 7) indicate that the computational complexity of this process is greater than that of all the remaining encoding steps combined. Thus, by reducing the complexity of this process, the overall complexity of the codec can be reduced.

3. **Motion-compensated error concealment techniques**. Apart from control and header data, the output of a typical video codec is one of two types: motion data or error (i.e., DFD) data.[5] Among the two types, motion data carries, in general, most of the information about a frame. In fact, at very low bit rates (typical of mobile video communication), motion data consumes a very high percentage of the available bit budget [8]. Thus, in the case of errors, it is very important to recover lost or erroneously received motion information. A class of error-resilience techniques that achieves this is motion-compensated error concealment, also known as *temporal error concealment*. Such techniques are particularly suited for mobile video communication, since, unlike other error resilience techniques, they do not increase the bit rate and they do not introduce any delay.

[4] An increase in motion overhead can be tolerated provided that the overall rate-distortion performance is improved.

[5] In the case of intracoded frames, the error signal is the same as the frame signal and no motion data is transmitted.

Part I

Introduction to Video Coding

This part gives an introduction to video coding. It contains two chapters. Chapter 2 introduces some of the fundamentals of video source coding. It starts by giving some basic definitions and then covers both analog and digital video along with some basic video coding techniques. It also presents the performance measures and the test sequences that will be used throughout the book. It then reviews both intraframe and interframe video coding methods.

Chapter 3 gives a brief introduction to video coding standards. The chapter starts by highlighting the need for video coding standards. It then outlines the chronological development of video coding standards, highlighting their main techniques and targeted applications. The chapter then concentrates on H.263 (and its recent extensions: H.263+ and H.263++) and MPEG-4 as examples of the state-of-the-art video coding standards.

Chapter 2

Video Coding: Fundamentals

2.1 Overview

This chapter gives a brief introduction to some fundamentals of video coding. Many of the concepts introduced in this chapter will be referenced and used in subsequent chapters. Section 2.2 gives some definitions. Section 2.3 covers analog video, whereas Section 2.4 concentrates on digital video. Section 2.5 introduces some of the basics of video coding. It also presents the performance measures and the test sequences that will be used in this book. Section 2.6 reviews intraframe video coding methods, whereas Section 2.7 reviews interframe coding methods.

2.2 What Is Video?

A *still image* is a spatial distribution of intensity[1] that is constant with respect to time [10]. *Video*, on the other hand, is a spatial intensity pattern that changes with time. Another common term for video is *image sequence*, since video can be represented by a time sequence of still-images.

2.3 Analog Video

2.3.1 Analog Video Signal

Video has traditionally been captured, stored, and transmitted in analog form. The term *analog video signal* refers to a one-dimensional (1-D) electrical

[1] Intensity is a measure over some interval of the electromagnetic spectrum of the flow of power that is radiated from, or incident on, a surface. It is usually measured in watts per square meter [9].

signal of time that is obtained by sampling the video intensity pattern in the *vertical* and *temporal* coordinates and converting intensity to electrical representation. This sampling process is known as *scanning*.

Raster scanning begins at the top-left corner and progresses horizontally, with a slight slope vertically, across the image. When it reaches the right-hand edge it snaps back to the left-hand edge (*horizontal retrace*) to start a new scan line. On reaching the bottom-right corner, a complete *frame* has been scanned and scanning snaps back to the top-left corner (*vertical retrace*) to begin a new frame. During retrace, blanking (black) and synchronization pulses are inserted.

The most commonly used raster scanning methods are progressive and interlaced, as illustrated in Figure 2.1. In *progressive* (also known as *noninterlaced* or 1:1) *scanning*, a frame is formed by a single scanning pass. In *interlaced* (or 2:1) *scanning*, however, a frame is formed by two successive scanning passes. In the first pass, the odd lines are scanned to form the first *field*, then the even lines are scanned to form the second field. When interleaved, the lines of the two fields form a single frame.

The aspect ratio, vertical resolution, frame rate, and refresh rate are important parameters of the video signal. The *aspect ratio* is the ratio of the width to the height of a frame. The *vertical resolution* is related to the number of scan lines per frame (including the blanking intervals). The *frame rate* is the number of frames scanned per second. The effect of smooth motion can be achieved using a frame rate of about 25–30 frames/s. However, at these frame rates the human eye picks up the flicker produced by refreshing the display between frames. To avoid this, the display refresh rate must be above 50 Hz.

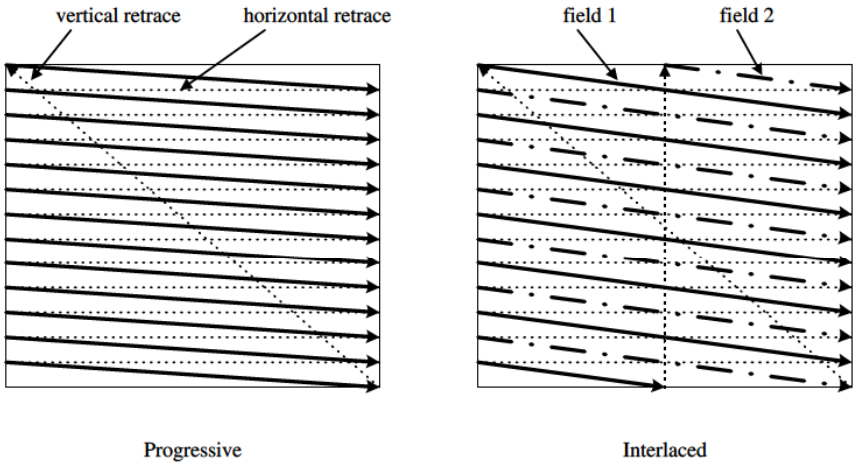


Figure 2.1: Raster scanning methods

Different industries employ different combinations of video parameters. For example, the computer industry uses progressive scanning with a frame rate of 72 frames/s. To reduce bandwidth requirements, the television industry uses interlaced scanning. In this case, the field rate is set to 50 or 60 fields/s to avoid refresh flicker,[2] while the frame rate (which, in interlaced video, is half the field rate) is 25 or 30 frames/s to maintain smooth motion. Note that this saving in bandwidth is at the expense of vertical resolution. There are two main television scanning systems: 625/50 (625 scan lines and 50 fields/s) and 525/60.

2.3.2 Color Representation

The preceding discussion considered monochrome video. In practice, however, most videos are in color. According to the *trichromatic theory* of color vision [11], color is perceived via *three* classes of cone cells, or photoreceptors, in the eye. Consequently, a color video can be produced by the superposition of three video signals. Each signal represents one of the three *primary* colors: red, green, and blue (RGB).[3] Practical television (TV) and video systems usually convert this RGB representation to a different color space of *luminance*[4] (which is closely related to the perception of brightness[5]) and *chrominance* (which is related to the perception of color hue[6] and saturation[7]). This representation serves two purposes. First, luminance ensures backward compatibility with monochrome video. Second, this representation lends itself more easily to video compression. This can be explained as follows. The human visual system (HVS) has poor response to color (chrominance) spatial detail compared to its response to luminance spatial detail [9]. Thus, the chrominance signals can be bandlimited or subsampled to achieve compression.

There are three main analog color coding systems: *Phase Alternation Line* (PAL), *SEquential Couleur Avec Memoire* (SECAM) and *National Television System Committee* (NTSC). They differ mainly in the way they

[2]Originally, television refresh rates were chosen to match the local AC power line frequency.

[3]The RGB is an *additive* color system. This means that when all the primaries are *added* in equal maximum quantities, the color white is perceived. In printing and painting, the cyan, magenta, and yellow (CMY) system is used. This is a *subtractive* color system since the total absorbtion of all three primaries produces the color white.

[4]Luminance is proportional to the light energy emitted per unit area of the source, but this energy is weighted according to the spectral sensitivity of the eye [9].

[5]Brightness is the attribute of a visual sensation according to which an area appears to emit more or less light [9].

[6]Hue is the attribute of a visual sensation according to which an area appears to be similar to one of the perceived colors, red, yellow, green and blue, or a combination of two of them [9].

[7]Saturation is the colorfulness of an area judged in proportion to its brightness [9].

calculate the luminance/chrominance components from the RGB components. For example, the PAL system calculates the luminance/chrominance components as follows:

$$\begin{aligned} Y' &= +0.299R' + 0.587G' + 0.114B', \\ U' &= -0.147R' - 0.289G' + 0.436B' = 0.493(B' - Y'), \\ V' &= +0.615R' - 0.515G' - 0.100B' = 0.877(R' - Y'), \end{aligned} \tag{2.1}$$

where $R'G'B'$ are *gamma-corrected*[8] components in the range $[0, 1]$. Note that Y' is closely related to gamma-corrected luminance and is usually referred to as *luma*. The chrominance is calculated as two *color-difference* components U' and V'. Again, since they are gamma-corrected, they are referred to as *chroma* components. The NTSC and SECAM systems calculate luma in the same way but use different coefficients for obtaining the chroma components (I' and Q' in NTSC, and D'_B and D'_R in SECAM).

2.3.3 Analog Video Systems

There are three main analog video systems. In most of Western Europe, a 625/50 PAL system is used. In Russia, France, the Middle East, and Eastern Europe, a 625/50 SECAM system is used. In North America and Japan, a 525/60 NTSC system is used. All three systems are interlaced with a 4:3 aspect ratio.

The three systems are *composite*. This means that the chroma components are first bandlimited and then combined (for example, by frequency interleaving) with the luma component. The resulting composite video signal has the same bandwidth as the original luma signal. For example, in the 625/50 PAL system, the luma signal has a bandwidth of 5.5 MHz. The chroma signals are bandlimited to about 1.5 MHz and then QAM (quadrature amplitude modulation) modulated with a color subcarrier at 4.43 MHz above the picture carrier. For a more detailed discussion of these systems the reader is referred to Ref. 13. There are also other analog video systems that use separate components (component video) or a separate luma component and a composite chroma component (S-video) [10].

[8] In a video system, it is important to convey luminance in such a way that noise and quantization have a perceptually similar effect across the entire scale from black to white. This is achieved by applying a nonlinear function to each of the linear RGB components. This process is known as *gamma-correction* [12].

2.4 Digital Video

2.4.1 Why Digital?

For the past two decades or so, the world has been experiencing a digital revolution. Most industries have witnessed a change from analog to digital technology, and video was no exception. Digital video has the following advantages over analog video

- Ease of editing, enhancing, and creating special effects.
- Avoidance of artefacts typical of analog video, like, for example, those caused by repeated recording on tapes, and errors in color rendition due to inaccuracies in the separation of composite video signals.
- Easy software conversion from one standard to another. For analog video conversion, expensive transcoders are needed.
- Robustness to noise and ease of encryption.
- Ease of scalability (spatial, temporal, or signal-to-noise ratio (SNR)). This facilitates the provision of the same service over a wide range of networks and hardware platforms.
- Interactivity.
- Ease of indexing, search and retrieval. For analog video, this requires tedious visual scanning.

These advantages allowed a number of new applications and services to be introduced. For example, the TV broadcasting industry is introducing new services like interactivity, search and retrieval, video-on-demand, and high-definition television (HDTV). The telecommunication industry is providing videoconferencing and videophones over a wide range of wired and wireless networks. The computer industry is providing desktop video and videoconferencing. Other applications include intelligent highway traffic control systems, medical imaging, surveillance, and flight simulation, to mention a few.

2.4.2 Digitization

The process of digitizing video involves three basic operations: *filtering*, *sampling*, and *quantization*. If the frequency content of the input analog signal exceeds half the sampling frequency, aliasing artefacts will occur. Thus, the filtering operation is used to bandlimit the input signal and condition it for the following sampling operation.

The amplitude of the filtered analog signal is then sampled at specific time instants to generate a discrete-time signal. The minimum sampling rate is known as the *Nyquist rate* and is equal to twice the signal bandwidth.

The resulting discrete-time samples have continuous amplitudes. Thus, it would require infinite precision to represent them. The quantization operation is used to map such values onto a finite set of discrete amplitudes that can be represented by a finite number of bits.

Each discrete-time, discrete-amplitude sample is called a *picture element* and is usually abbreviated to a *pel* or a *pixel*. The pels are arranged in a two-dimensional (2-D) array to form a digital still image or a digital frame. A digital video consists of a sequence of such digital frames.

For color video, the foregoing operations are repeated for each component. Thus, a digital still image would normally be represented by three 2-D arrays. Almost all digital video systems use component representation. This avoids the artefacts that result from composite encoding.[9]

As an example, consider the digitization of a 625/50 PAL analog signal. The luma and chroma components are first filtered to 5.5 MHz and 1.5 MHz, respectively. During sampling, minimum sampling frequencies of 11 MHz and 3 MHz must be used to sample the luma and chroma components, respectively. The resulting discrete-time signals are then quantized to a given precision (usually 8 bits).

2.4.3 Chroma Subsampling

As already mentioned, the HVS has poor response to chrominance spatial detail compared to its response to luminance spatial detail. This property can be exploited to reduce bandwidth requirements by subsampling the chroma components. The most commonly used subsampling patterns are illustrated in Figure 2.2. In 4:2:2 subsampling, the chroma components are subsampled by a factor of 2 horizontally. This gives a reduction of about 33% in the overall raw data rate. In 4:1:1 subsampling, the chroma components are subsampled by a factor of 4 horizontally, giving a reduction of 50%. In 4:2:0 subsampling, the chroma components are subsampled by a factor of 2 both horizontally and vertically, giving a reduction of 50% in the overall raw data rate. Vertically subsampled chroma samples are always sited midway between luma samples. Horizontally subsampled chroma samples, however,

[9] As already discussed, composite encoding is used in analog systems to save bandwidth. In digital systems, however, bandwidth is saved using digital video compression techniques, as will be described later.

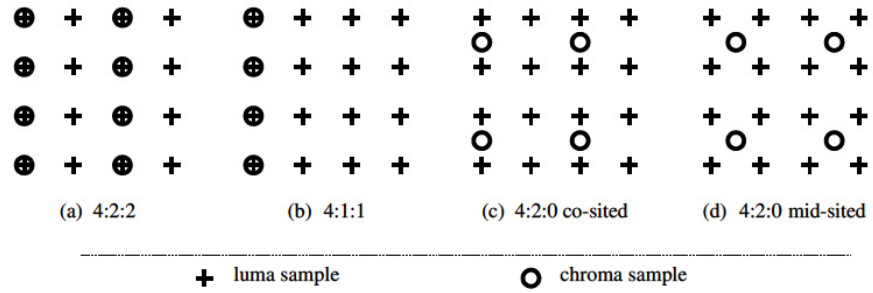


Figure 2.2: Chroma subsampling patterns

can be either midway between luma samples (Figure 2.2(d)) or co-sited with odd-numbered luma samples (Figure 2.2(c)).[10]

2.4.4 Digital Video Formats

Exchange of digital video between different industries, applications, networks, and hardware platforms requires standard digital video formats. Following are the most commonly used formats.

2.4.4.1 CCIR-601

The *International Consultative Committee for Radio* (CCIR)[11] Recommendation 601 [14] defines a digital video format for the international exchange and broadcast of production-quality TV programs. As with analog standards, CCIR-601 defines two interlaced systems: 525/60 and 625/50. The main family within the standard uses a chroma subsampling of 4:2:2. The luma sampling frequency is 13.5 MHz, the chroma sampling frequency is $13.5 \times 0.5 = 6.75$ MHz, and the components are quantized to 8 bits. In the 525/60 system, the luma component of the frame has active dimensions of 720 pels $\times$ 480 lines and the chroma components have 360 pels $\times$ 480 lines. In the 625/50 system, the corresponding values are 720×576 for luma and 360×576 for chroma. Note that despite the differences between the two systems, they generate the same raw bit rate[12] of 165.89 Mbits/s. The standard is based on component

[10] In view of this lack of consistency, the authors adopt the terms *mid-sited* and *co-sited* to describe the two cases.

[11] The CCIR is currently known as ITU-R (*International Telecommunications Union—Radio Sector*).

[12] Bit rate $= [(720 \times 480) + 2(360 \times 480)] \times 30 \times 8 = [(720 \times 576) + 2(360 \times 576)] \times 25 \times 8 =$ 165888000 bits/s, where the 2 refers to the two chroma components, the 30 and the 25 are the frame rates of the two systems, and the 8 is the number of bits per sample.

video with one luma (Y') and two chroma (C'_R and C'_B) components calculated as follows:

$$\begin{aligned} Y' &= 219(+0.299R' + 0.587G' + 0.114B') + 16, \\ C'_B &= 224(-0.169R' - 0.331G' + 0.500B') + 128, \\ C'_R &= 224(+0.500R' - 0.419G' - 0.081B') + 128, \end{aligned} \tag{2.2}$$

where Y' has 220 levels in the range [16, 235], with black at 16 and white at 235, and C'_B and C'_R have 225 levels in the range [16, 240], with zero difference at 128. Note that other levels within the 8-bit range [0, 255] are reserved for synchronization and signal processing head- and foot-rooms.

2.4.4.2 SIF and QSIF

CCIR-601 was defined mainly for broadcast-quality applications. For storage applications, a lower-resolution format called the *Source Input Format* (SIF) was defined. This is a progressive 4:2:0 mid-sited format with a luma component that is half the CCIR-601 active luma component in both dimensions. The CCIR-601 format has 720 luminance pels/line, which means that an SIF format must have $720/2 = 360$ luma pels/line. Since 360 is not divisible by 16 (which is the main coding unit within standard video codecs), 8 pels (4 from each side) are usually discarded to reduce the number of luma pels per line to 352. Since there are two CCIR-601 systems, there are two SIF formats: the first has a luma component of 352×240, chroma components of 176×120, and a frame rate of 30 frames/s, whereas the second format has a luma of 352×288, chromas of 176×144, and a frame rate of 25 frames/s.

A lower-resolution version of SIF is the *quarter-SIF* (QSIF) format. It has half the dimensions of SIF in both directions. This means it has quarter the number of samples, hence the name. Again, two versions are available: the first has a luma of 176×120, chromas of 88×60, and a frame rate of 30 frames/s, whereas the second has a luma of 176×144, chromas of 88×72, and a frame rate of 25 frames/s. For methods of converting between CCIR-601, SIF and QSIF, refer to Ref. 15.

2.4.4.3 CIF and Its Family

In order for video codecs to cope with both 525/60 and 625/50 formats, a common format was defined. In this format, the luma component has a horizontal resolution that is half that of both CCIR-601 systems, a vertical resolution that is half that of the 625/50 system, and a temporal resolution that is half that of the 525/60 system. This intermediate choice of vertical resolution from one system and temporal resolution from the other leads to the name *Common Intermediate Format* (CIF). The CIF is progressive, with

Table 2.1: The CIF family

	Luma		Chromas	
	pels/line	lines/frame	pels/line	lines/frame
SQCIF	128	96	64	48
QCIF	176	144	88	72
CIF	352	288	176	144
4CIF	704	576	352	288
16CIF	1408	1152	704	576

4:2:0 mid-sited chroma subsampling and a frame rate of 30 frames/s. There are a number of lower- and higher-resolution members in the CIF family. Those are defined in Table 2.1.

2.4.4.4 Other Formats

There are a number of other formats. For example, some HDTV systems[13] use a 1440×1050 luma at 30 frames/s with progressive scanning and no chroma subsampling (i.e., 4:4:4).

2.5 Video Coding Basics

2.5.1 The Need for Video Coding

Table 2.2 shows the raw data rates of a number of typical video formats, whereas Table 2.3 shows a number of typical video applications and the bandwidths available to them. It is immediately evident that video coding (or *compression*) is a key enabling technology for such applications. Consider a 2-hour CCIR-601 color movie. Without compression, a 5-Gbit compact disc (CD) can hold only 30 seconds of this movie. To store the entire movie on the same CD requires a compression ratio of about 240:1. Without compression, the same movie will take about 36 days to arrive at the other end of a 384 kbits/s Integrated Services Digital Network (ISDN) channel. To achieve real-time transmission of the movie over the same channel, a compression ratio of about 432:1 is required.

[13] A range of HDTV formats exist.

Table 2.2: Raw data rates of typical video formats

Format	Raw data rate
HDTV	1.09 Gbits/s
CCIR-601	165.89 Mbits/s
CIF @ 15 f.p.s.	18.24 Mbits/s
QCIF @ 10 f.p.s.	3.04 Mbits/s

Table 2.3: Typical video applications

Application	Bandwidth
HDTV (6-MHz channel)	20 Mbits/s
Desktop video (CD-ROM)	1.5 Mbits/s
Videoconferencing (ISDN)	384 kbits/s
Videophone (PSTN)	56 kbits/s
Videophone (GSM)	10 kbits/s

2.5.2 Elements of a Video Coding System

The aim of video coding is to reduce, or compress, the number of bits used to represent video. Video signals contain three types of redundancy: statistical, psychovisual, and coding redundancy. *Statistical redundancy* is present because certain data patterns are more likely than others. This is mainly due to the high spatial (*intraframe*) and temporal (*interframe*) correlations between neighboring pels. *Psychovisual redundancy* is due to the fact that the HVS is less sensitive to certain visual information than to other visual information. If video is coded in a way that uses more and/or longer code symbols than absolutely necessary, it is said to contain *coding redundancy*. Video compression is achieved by reducing or eliminating these redundancies.

Figure 2.3 shows the main elements of a video encoder. Each element is designed to reduce one of the three basic redundancies.

The *mapper* (or *transformer*) transforms the input raw data into a representation that is designed to reduce statistical redundancy and make the data more amenable to compression in later stages. The transformation is a one-to-one mapping and is, therefore, reversible.

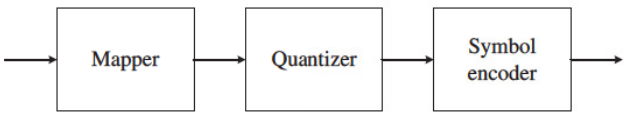


Figure 2.3: Elements of a video encoder

The *quantizer* reduces the accuracy of the mapper's output, according to some fidelity criterion, in an attempt to reduce psychovisual redundancy. This is a many-to-one mapping and is, therefore, irreversible.

The *symbol encoder* (or *codeword assigner*) assigns a codeword, a string of binary bits, to each symbol at the output of the quantizer. The code must be designed to reduce coding redundancy. This operation is reversible.

In general, compression methods can be classified into lossless methods and lossy methods. In *lossless* methods the reconstructed (compressed-decompressed) data is identical to the original data. This means that such methods do not employ a quantizer. Lossless methods are also known as *bit-preserving* or *reversible* methods. In lossy methods the reconstructed data is not identical to the original data; that is, there is loss of information due to the quantization process. Such methods are therefore *irreversible*, and they usually achieve higher compression than lossless methods.

2.5.3 Elements of Information Theory

A source $\mathscr{S}$ with an alphabet $\mathscr{A}$ can be defined as a discrete random process $\mathscr{S} = S_1, S_2, \ldots$, where each random variable S_i takes a value from the alphabet $\mathscr{A}$.

In a *discrete memoryless source* (DMS) the successive symbols of the source are statistically independent. Such a source can be completely defined by its alphabet $\mathscr{A} = \{a_1, a_2, \ldots, a_N\}$ and the associated probabilities $\mathscr{P} = \{p(a_1), p(a_2), \ldots, p(a_N)\}$, where $\sum_{i=1}^{N} p(a_i) = 1$. According to information theory, the information I contained in a symbol a_i is given by

$$I(a_i) = \log_2 \frac{1}{p(a_i)} = -\log_2 p(a_i) \qquad \text{(bits)}, \tag{2.3}$$

and the average information per source symbol $H(\mathscr{S})$, also known as the *entropy* of the source, is given by

$$H(\mathscr{S}) = \sum_{i=1}^{N} p(a_i) I(a_i) = -\sum_{i=1}^{N} p(a_i) \log_2 p(a_i) \qquad \text{(bits/symbol)}. \tag{2.4}$$

A more realistic approach is to model sources using Markov-K random processes. In this case the probability of occurrence of a symbol depends on the values of the K preceding symbols. Thus, a Markov-K source can be specified by the conditional probabilities $p(S_j = a_i | S_{j-1}, \ldots, S_{j-K})$, for all j, $a_i \in \mathscr{A}$. In this case, the entropy is given by

$$H(\mathscr{S}) = \sum_{S^K} p(S_{j-1}, \ldots, S_{j-K}) H(\mathscr{S} | S_{j-1}, \ldots, S_{j-K}), \tag{2.5}$$

where S^K denotes all possible realizations of $S_{j-1},\ldots,S_{j-K}$, and

$$H(\mathcal{S}|S_{j-1},\ldots,S_{j-K}) = -\sum_{a_i\in\mathcal{A}} p(S_j=a_i|S_{j-1},\ldots,S_{j-K})\log p(S_j=a_i|S_{j-1},\ldots,S_{j-K}). \tag{2.6}$$

The performance bound of a lossless coding system is given by the lossless coding theorem [16]:

> *Lossless coding theorem*: The minimum bit rate $R_{\min}$ that can be achieved by lossless coding of a source $\mathcal{S}$ can be arbitrarily close, but not less than, the source entropy $H(\mathcal{S})$. Thus $R_{\min}=H(\mathcal{S})+\varepsilon$, where ε is a positive quantity that can be made arbitrarily close to zero.

For a DMS, this lower bound can be approached by coding symbols independently, whereas for a Markov-K source, blocks of K symbols should be encoded at a time.

The performance bounds of lossy coding systems are addressed by a branch of information theory known as *rate-distortion theory* [16, 17, 18]. This theory provides lower bounds on the obtainable average distortion for a given average bit rate, or vice versa. It also promises that codes exist that approach the theoretical bounds when the code dimension and delay become large. An important theorem in this branch is the source coding theorem [17]:

> *Source coding theorem*: There exists a mapping from source symbols to codewords such that for a given distortion D, $R(D)$ bits/symbol are sufficient to achieve an average distortion that is arbitrarily close to D.

The function $R(D)$ is known as the *rate-distortion function*. It is a convex, continuous, and strictly decreasing function of D, as illustrated in Figure 2.4. This function is normally computed using numerical methods [18], although for simple source and distortion models it can be computed analytically. Although rate-distortion theory does not give an explicit method for constructing practical optimum coding systems, it gives very important hints about the properties of such systems.

2.5.4 Quantization

As already discussed, quantization is a key element of a video coding system. Quantization can be viewed as a many-to-one mapping. It represents a set of continuous-valued samples with a finite number of symbols. If each input sample is quantized independently, then the process is referred to as *scalar*

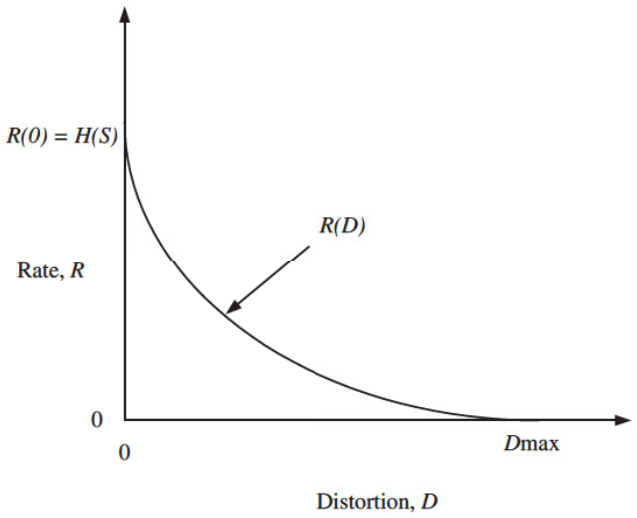


Figure 2.4: Rate-distortion function

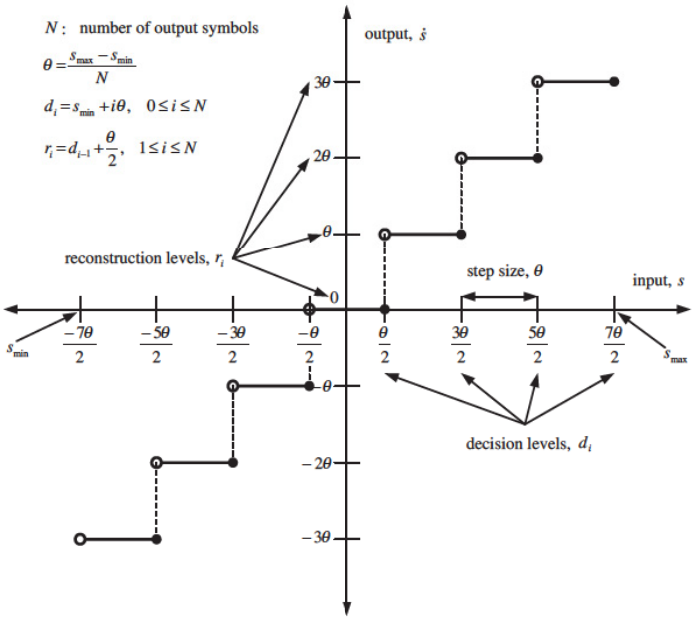
quantization. If, however, the input samples are grouped into a set of vectors and this set is mapped to a finite number of vectors, then the process is known as *vector quantization*. Vector quantization is discussed in more detail in Section 2.6.4.

Assume that the quantizer input s varies between $s_{\min}$ and $s_{\max}$ and that this range is to be mapped to a finite set of N symbols, then a set of $N+1$ *decision levels* d_i, $0 \le i \le N$, are first defined, where $d_0 = s_{\min}$ and $d_N = s_{\max}$. This divides the input range into N *quantization intervals*. At the output of the quantizer, each quantization interval is then represented by a *reconstruction level* r_i, $1 \le i \le N$. Thus, a scalar quantizer $Q(\cdot)$ can be defined as follows:

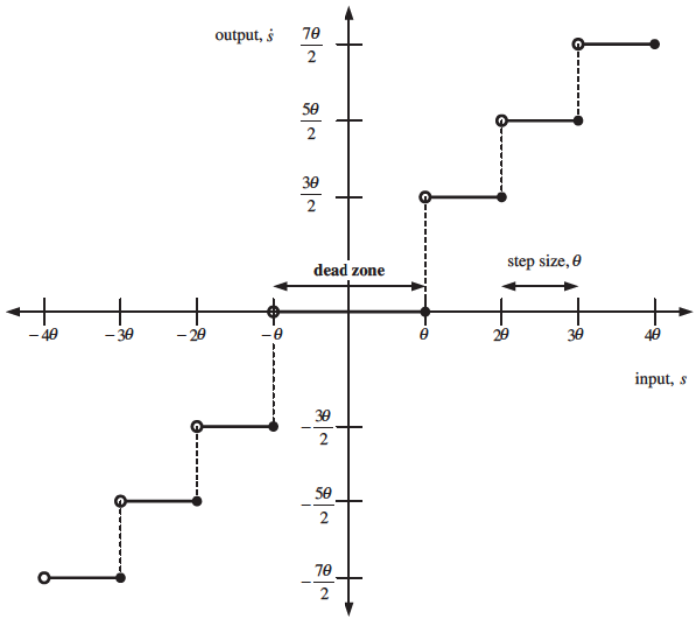
$$\dot{s} = Q(s) = r_i, \qquad \text{if } d_{i-1} < s \le d_i, \quad \text{where } 1 \le i \le N, \tag{2.7}$$

where $\dot{s}$ is the quantized output. There are, in general, two types of optimum scalar quantizers: Lloyd-Max and entropy-constrained. Lloyd-Max [19, 20] quantizers are designed to minimize the mean squared error with a fixed number of levels. Entropy-constrained quantizers [21] are designed to minimize a distortion measure for a constant output entropy.

The simplest form of scalar quantization is *uniform quantization*. In this case, the decision levels (and the reconstruction levels) are equally spaced, with a *quantizer step size* θ. In addition, the reconstruction levels are set to the midpoints of the quantization intervals. Figure 2.5(a) shows an example of a uniform quantizer, with $N = 7$ reconstruction levels. In this case,



(a) Without dead zone



(b) With dead zone

Figure 2.5: Uniform threshold quantizers

the quantization process can be implemented at the encoder using

$$\hat{s} = \text{NINT}\left[\frac{s}{\theta}\right], \tag{2.8}$$

where NINT[·] is the operation of rounding to the nearest integer and $\hat{s}$ is called the *quantization index*. It is the quantization index that is encoded and sent to the decoder. The decoder can then dequantize this index to obtain the reconstructed output as follows:

$$\dot{s} = \theta \cdot \hat{s}. \tag{2.9}$$

This type of quantizer is also known as a *threshold quantizer*, because it quantizes to zero all those inputs whose magnitudes are below a threshold. As will be discussed later, this type of quantizer is usually used in transform coding to reduce the number of transform coefficients that need to be encoded. Another example of uniform threshold quantizers is illustrated in Figure 2.5(b). In this case, the quantization interval around zero has been extended to form a *dead zone*. This causes more nonsignificant inputs to be quantized to zero and, thus, increases compression. The quantization equation for this quantizer is given by

$$\hat{s} = \text{FIX}\left[\frac{s}{\theta}\right], \tag{2.10}$$

where FIX[·] is the operation of rounding to the nearest integer toward zero (i.e., truncation). The corresponding dequantization equation is given by

$$\dot{s} = \theta \cdot \hat{s} + \text{SIGN}(\hat{s}) \cdot \frac{\theta}{2}, \tag{2.11}$$

$$\text{SIGN}(a) = \begin{cases} +1, & a > 0, \\ 0, & a = 0, \\ -1, & a < 0. \end{cases} \tag{2.12}$$

Scalar quantizers can also be *nonuniform*. In this case, more reconstruction levels are assigned to more significant subintervals within the input range. This yields a higher overall accuracy.

2.5.5 Symbol Encoding

Another key element of video coding systems is the symbol encoder. This assigns a codeword to each symbol at the output of the quantizer. The symbol encoder must be designed to reduce the coding redundancy present in the set of symbols. Following are a number of commonly used techniques that can be applied individually or in combinations.

2.5.5.1 Run-Length Encoding

The output of the quantization step may contain long runs of identical symbols. One way to reduce this redundancy is to employ *run-length encoding* (RLE). There are different forms of RLE. For example, if the quantizer output contains long runs of zeros, then RLE can represent such runs with intermediate symbols of the form (RUN, LEVEL). For example, a run of the form $0,0,0,0,0,9$ can be represented by the intermediate symbol (5,9).

2.5.5.2 Entropy Encoding

The quantizer can be considered a DMS $\mathcal{Q}$ that can be completely specified by its alphabet $\mathcal{R} = \{r_1, r_2, \ldots, r_N\}$, where r_i are the reconstruction levels and the associated probabilities of occurrence $\mathcal{P} = \{p(r_1), p(r_2), \ldots, p(r_N)\}$. The information contained in a symbol $I(r_i)$ is given by Equation (2.3), whereas the entropy of the source $H(\mathcal{Q})$ is given by Equation (2.4).

Now consider a symbol encoder that assigns a codeword c_i of length $l(c_i)$ bits to symbol r_i. Then the average word length $\bar{L}$ of the code is given by

$$\bar{L} = \sum_{i=1}^{N} p(r_i) l(c_i) \qquad \text{(bits)}, \tag{2.13}$$

and the efficiency (η) of the code is

$$\eta = \frac{H(\mathcal{Q})}{\bar{L}}. \tag{2.14}$$

Thus, an optimal ($\eta = 1$) code must have an average word length that is equal to the entropy of the source; i.e., $\bar{L} = H(\mathcal{Q})$. Clearly, this can be achieved if each codeword length is equal to the information content of the associated symbol, that is, $l(c_i) = I(r_i)$. Since $I(r_i)$ is inversely proportional to $p(r_i)$ (from Equation (2.3)), then an efficient code must assign shorter codewords to more probable symbols, and vice versa. This is known as *entropy encoding* or *variable-length coding* (VLC) (as opposed to *fixed-length coding* (FLC)).

The most commonly used VLC is *Huffman coding* [22]. Given a finite set of symbols and their probabilities, Huffman coding yields the optimal[14] integer-length prefix[15] code. The basic principles of Huffman coding can be illustrated using the example given in Figure 2.6. In each stage, the two least probable symbols are combined to form a new symbol with a probability equal

[14] Huffman is optimal in the sense that no other integer-length VLC can achieve a smaller average word length.

[15] In a prefix code, no codeword is a prefix of another codeword. This makes the code uniquely decodable.

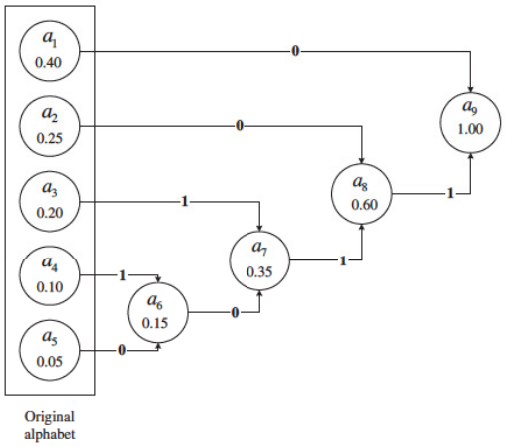


Figure 2.6: Huffman coding example

Table 2.4: Comparison between VLC (of Figure 2.6) and a 3-bit FLC

r_i	$p(r_i)$	$I(r_i)$	VLC c_i	FLC c_i
a_1	0.40	1.32 bits	0 (1 bit)	000
a_2	0.25	2.00 bits	10 (2 bits)	001
a_3	0.20	2.32 bits	111 (3 bits)	010
a_4	0.10	3.32 bits	1101 (4 bits)	011
a_5	0.05	4.32 bits	1100 (4 bits)	100
$H(R) \approx 2.04$ bits/symbol				
$\bar{L}_{\text{FLC}} = 3$ bits/word			$\bar{L}_{\text{VLC}} \approx 2.1$ bits/word	
$\eta_{\text{FLC}} \approx 0.68$			$\eta_{\text{VLC}} \approx 0.97$	

to the sum of their probabilities. This new symbol creates a new node in the tree, with two branches connecting it to the original two nodes. A "0" is assigned to one branch and a "1" is assigned to the other. The original two nodes are then removed from the next stage. This process is continued until the new symbol has a probability of 1. Now, to find the codeword for a given symbol, start at the right-hand end of the tree and follow the branches that lead to the symbol of interest combining the "0"s and "1"s assigned to the branches. Table 2.4 shows the obtained VLC and compares it to an FLC of 3 bits. Clearly, the Huffman VLC is much more efficient than the FLC.

There are more efficient implementations of Huffman coding. For example, in many cases, most of the symbols of a large symbol set have very small probabilities. This leads to very long codewords and consequently to large

storage requirements and high decoding complexity. In the *modified Huffman code* [23] the less probable symbols (and their probabilities) are lumped into a single symbol like ESCAPE. A symbol in this new ESCAPE category is coded using the VLC codeword for ESCAPE followed by extra bits to identify the actual symbol. Standard video codecs also use 2-D and 3-D versions of the Huffman code. For example, the H.263 standard (see Section 3.4) uses a 3-D Huffman code where three different symbols (LAST, RUN, LEVEL) are lumped into a single symbol (EVENT) and then encoded using one VLC codeword.

One disadvantage of the Huffman code is that it can only assign integer-length codewords. This usually leads to a suboptimal performance. For example, in Table 2.4, the symbol a_3 was represented with a 3-bit codeword, whereas its information content is only 2.32 bits. In fact, Huffman code can be optimal only if all the probabilities are integer powers of 1/2. An entropy code that can overcome this limitation and approach the entropy of the source is *arithmetic coding* [24]. In Huffman coding there is a one-to-one correspondence between the symbols and the codewords. In arithmetic coding, however, a single variable-length codeword is assigned to a variable-length block of symbols.

2.5.6 Performance Measures

When evaluating the performance of a video coding system, a number of aspects need to be assessed and measured. One important aspect is the amount of *compression* (C) achieved by the system. This can be measured in a number of ways:

$$C = \frac{\text{number of bits in original video}}{\text{number of bits in compressed video}} \quad \text{(unitless)}, \tag{2.15}$$

$$C = \frac{\text{number of bits in compressed video}}{\text{number of pels in original video}} \quad \text{(bits/pel)}, \tag{2.16}$$

$$C = \frac{\text{number of bits in compressed video}}{\text{number of frames in original video}} \times \text{frame rate} \quad \text{(bits/s)}. \tag{2.17}$$

Another important aspect is the *reconstruction quality*. This can be assessed using a number of subjective and objective measures. *Subjective measures* are normally evaluated by showing the reconstructed video to a group of subjects and asking for their views on the perceived quality. A number of subjective assessment methodologies have been developed over the years. Examples are

the double stimulus impairment scale (DSIS) and the double and single stimulus continuous quality scales, (DSCQS) and (SSCQS), respectively. For a detailed description of such experiments the reader is referred to Ref. 25.

Despite their reliability, subjective quality experiments are expensive and time consuming. *Objective measures* provide cheaper and faster alternatives. One commonly used objective measure is the *mean squared error* (MSE), which is defined as

$$\mathrm{MSE} = \frac{1}{H \times V} \sum_{x=1}^{H} \sum_{y=1}^{V} [f(x,y) - \hat{f}(x,y)]^2, \tag{2.18}$$

where H and V are the horizontal and vertical dimensions of the frame, respectively, and $f(x,y)$ and $\hat{f}(x,y)$ are the pel values at location (x,y) of the original and reconstructed frames, respectively. Care should be taken to include color components and to take into account any chroma subsampling. For example, the MSE of a reconstructed 4:2:0 color frame can be calculated as

$$\begin{aligned} \mathrm{MSE}_{4:2:0} = \frac{1}{\frac{3}{2} H \times V} \Bigg(& \sum_{x=1}^{H} \sum_{y=1}^{V} [Y'(x,y) - \hat{Y}'(x,y)]^2 \\ & + \sum_{x=1}^{H/2} \sum_{y=1}^{V/2} [C'_R(x,y) - \hat{C}'_R(x,y)]^2 \\ & + \sum_{x=1}^{H/2} \sum_{y=1}^{V/2} [C'_B(x,y) - \hat{C}'_B(x,y)]^2 \Bigg) \\ = \tfrac{2}{3} & (\mathrm{MSE}_{Y'} + \tfrac{1}{4}\mathrm{MSE}_{C'_R} + \tfrac{1}{4}\mathrm{MSE}_{C'_B}). \end{aligned} \tag{2.19}$$

A more common form of the MSE measure is the *peak signal-to-noise ratio* (PSNR), which is defined as

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{f_{\max}^2}{\mathrm{MSE}} \right) \quad (\mathrm{dB}), \tag{2.20}$$

where $f_{\max}$ is the maximum possible pel value (for example, 255 for an 8-bit resolution component). Although this measure does not always correlate well with perceived video quality, its relative simplicity makes it a very popular choice in the video coding community. Thus, to facilitate comparisons with other algorithms reported in the literature, this book adopts the PSNR measure. If accuracy is a major concern, then more sophisticated objective measures based on perceptual models can be used [26].

When testing a video coding algorithm, it is very important to subject it to a range of input video sequences with different characteristics and a reasonable



(a) FOREMAN (b) AKIYO (c) TABLE TENNIS

Figure 2.7: Three test sequences

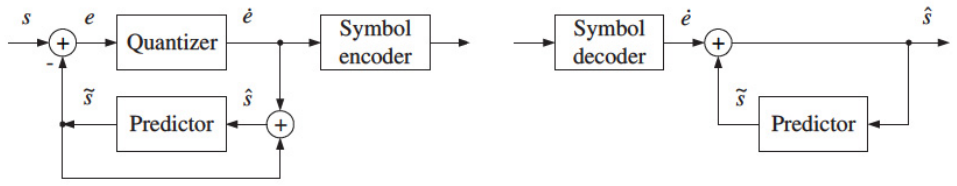
spread of data properties. The *Moving Picture Experts Group* (MPEG) established a library of CCIR-601 test sequences divided into five classes: class A (low spatial detail and low amount of motion), class B (medium spatial detail and low amount of motion or vice versa), class C (high spatial detail and medium amount of motion, or vice versa), class D (stereoscopic), and class E (hybrid of natural and synthetic content) [27]. The first three classes are more relevant to the work carried out in this book. Thus, the book uses three test sequences: AKIYO, FOREMAN, and TABLE TENNIS, where each sequence is a representative of one of the three relevant classes, A, B, and C, respectively. The three sequences are at QSIF resolution and include 300 frames each. This resolution is typical of the sequences used in very-low-bit-rate applications. Both AKIYO and TABLE TENNIS have luma components of 176×120 and a frame rate of 30 frames/s, whereas FOREMAN has a luma component of 176×144 and a frame rate of 25 frames/s. Figure 2.7 shows the luma component of the first frame of each of the three test sequences.

2.6 Intraframe Coding

Intraframe coding refers to video coding techniques that achieve compression by exploiting (reducing) the high spatial correlation between neighboring pels within a video frame. Such techniques are also known as *spatial redundancy reduction* techniques or *still-image coding* techniques.

2.6.1 Predictive Coding

Predictive coding was originally proposed by Cutler in 1952 [28]. In this method, a number of previously coded pels are used to form a *prediction* of the current pel. The *difference* between the pel and its prediction forms the signal to be coded. Obviously, the better the prediction, the smaller the error



(a) Encoder (b) Decoder

Figure 2.8: Block diagram of a predictive coding system

signal and the more efficient the coding system. At the decoder, the same prediction is produced using previously decoded pels, and the received error signal is added to reconstruct the current pel. A block diagram of a predictive coding system is depicted in Figure 2.8. Predictive coding is commonly referred to as *differential pulse code modulation* (DPCM). A special case of this method is *delta modulation* (DM), which quantizes the error signal using two quantization levels only.

Predictive coding can take many forms, depending on the design of the predictor and the quantizer blocks. The predictor can use a linear or a nonlinear function of the previously decoded pels, it can be 1-D (using pels from the same line) or 2-D (using pels from the same line and from previous lines), and it can be fixed or adaptive. The quantizer also can be uniform or nonuniform, and it can be fixed or adaptive.

The minimal storage and processing requirements were partly responsible for the early popularity of this method, when storage and processing devices were scarce and expensive resources. The method, however, provides only a modest amount of compression. In addition, its performance is highly dependent on the statistics of the input data, and it is very sensitive to errors (feedback through the prediction loop can cause error propagation). As processing and storage devices became more available, more complex, more efficient methods like transform coding have become more popular. Despite this, predictive coding is still used in video coding, as, for example, in the lossless coding of motion vectors.

2.6.2 Transform Coding

Transform coding, developed more than two decades ago, has proven to be a very effective video coding method. Today, it forms the basis of almost all video coding standards. Figure 2.9 shows a block diagram of a typical transform coding system. The input frame is first segmented into $N \times N$ blocks.

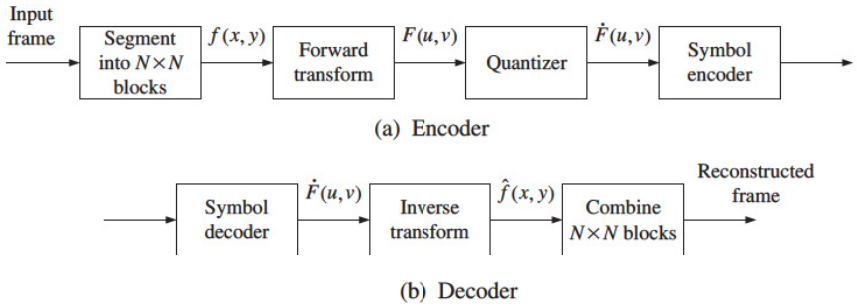


Figure 2.9: Block diagram of a transform coding system

A unitary[16] space-frequency transform is applied to each block to produce an $N \times N$ block of transform (spectral) coefficients that are then suitably quantized and coded. At the decoder, an inverse transform is applied to reconstruct the frame. The main goal of the transform is to decorrelate the pels of the input block. This is achieved by redistributing the energy of the pels and concentrating most of it in a small set of transform coefficients. This is known as *energy compaction*. The transform process can also be interpreted as a coordinate rotation of the input or as a decomposition of the input into orthogonal basis functions weighted by the transform coefficients [29]. Compression comes about from two main mechanisms. First, low-energy coefficients can be discarded with minimum impact on the reconstruction quality. Second, the HVS has differing sensitivity to different frequencies. Thus, the retained coefficients can be quantized according to their visual importance.

When choosing a transform, three main properties are desired: good energy compaction, data-independent basis functions, and fast implementation. The Karhunen-Loève transform (KLT) is the optimal transform in an energy-compaction sense. Unfortunately, this optimality is due to the fact that the KLT basis functions are dependent on the covariance matrix of the input block. Recomputing and transmitting the basis functions for each block is a nontrivial computational task. These disadvantages severely limit the use of the KLT in practical coding systems. The performance of many suboptimal transforms with data-independent basis functions have been studied [30]. Examples are the discrete Fourier transform (DFT), the discrete cosine transform (DCT), the Walsh-Hadamard transform (WHT), and the Haar transform. It has been demonstrated that the DCT has the closest energy-compaction performance to that of the optimum KLT [30]. This has motivated the development of a number of fast DCT algorithms, e.g., Ref. 31. Due to these

[16] A unitary transform is a reversible linear transform with orthonormal basis functions [29].

attractive features, i.e., near-optimum energy-compaction, data-independent basis functions and fast algorithms, the DCT has become the "workhorse" of most image and video coding standards.

The DCT was developed by Ahmed *et al.* in 1974 [32]. There are four slightly different versions of the DCT [33], but the one commonly used for video coding is denoted by DCT-II. The 2-D DCT-II of an $N \times N$ block of pels is given by

$$F(u,v) = C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left(\frac{(2x+1)u\pi}{2N}\right)\cos\left(\frac{(2y+1)v\pi}{2N}\right), \tag{2.21}$$

where $f(x,y)$ is the pel value at location (x,y) within the block, $F(u,v)$ is the corresponding transform coefficient, $0 \le u,v,x,y \le N-1$, and

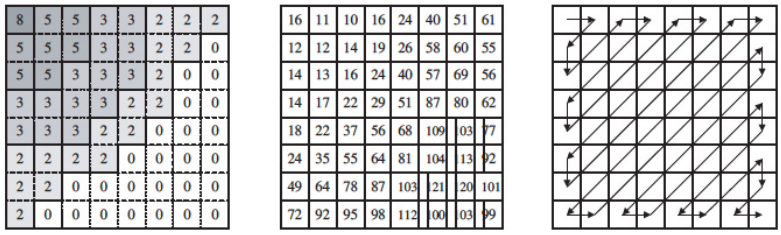
$$C(\alpha) = \begin{cases} \sqrt{\frac{1}{N}}, & \alpha = 0, \\ \sqrt{\frac{2}{N}}, & \text{otherwise.} \end{cases} \tag{2.22}$$

The transform coefficient $F(0,0)$ at the top-left corner of the transformed block is called the DC coefficient because it contains the lowest frequencies in both the horizontal and vertical dimensions. The corresponding inverse DCT transform is given by

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u)C(v)F(u,v)\cos\left(\frac{(2x+1)u\pi}{2N}\right)\cos\left(\frac{(2y+1)v\pi}{2N}\right). \tag{2.23}$$

It can be deduced from Equation (2.21) that the computational complexity of an $N \times N$ 2-D DCT is of the order $\mathcal{O}(N^4)$. However, one of the advantages of the DCT is that it is separable. This means that a 2-D DCT can be separated into a pair of 1-D DCTs. Thus, to obtain the 2-D DCT of an $N \times N$ block, a 1-D DCT is performed first on each of the N rows of the block and then on each of the N columns of the resulting block (or vice versa). The same applies to the inverse DCT. This reduces the complexity to $\mathcal{O}(2N^3)$. Further reductions in complexity can be achieved using a number of fast DCT algorithms [31].

Beside transform selection, a significant factor that affects transform coding performance and computational complexity is the block size. In general, the



(a) Zonal mask with bit allocation

(b) Quantization matrix for threshold coding

(c) Zigzag scanning

Figure 2.10: Transform coefficient bit allocation

use of smaller block sizes reduces computational complexity.[17] However, as will be discussed later, transform coding suffers from blocking artefacts at very low bit rates. Such artefacts are more disturbing with smaller block sizes [15]. As a compromise between computational complexity and blocking artefacts, most transform coding systems employ a block size of 8×8 or 16×16. Note that both sizes are powers of 2, which simplifies computations.

Another important factor in transform coding is *bit allocation*. This refers to the process of determining which coefficients should be retained for coding and how coarsely each retained coefficient should be quantized. There are two main approaches: zonal coding and threshold coding. In *zonal coding* the retained coefficients are selected on the basis of maximum variance. Thus, the locations of the retained coefficients with the largest variances are indicated by a zonal mask that is the same for all blocks. Once the retained coefficients are decided, a number of methods can be used to decide the number of bits allocated to each. One method is to choose the number of bits to be proportional to the variance of the coefficient. Figure 2.10(a) shows a zonal mask with the allocated bits. Once the number of bits allocated for each coefficient is determined, a different quantizer can be designed for each coefficient.

One disadvantage of zonal coding is that the locations of the retained coefficients and the bits allocated to them are fixed for all blocks. In *threshold coding*, however, the locations and the bit allocation can be adapted to the characteristics of the block. For this reason, this method is employed by most video coding standards. In threshold coding, the retained coefficients are selected on the basis of maximum magnitude. Thus, only those coefficients whose

[17] For example, if a 256×256 frame was divided into 256 blocks, each of 16×16 pels, then a direct implementation of the 2-D DCT will require: blocks $\times N^4 = 256 \times 16^4 = 16,777,216$ multiplications. If, however, the same frame was divided into 4096 blocks, each of 4×4 pels, then $4096 \times 4^4 = 1,048,576$ multiplications will be required.

magnitudes are above a threshold are retained. In practice, the thresholding and the following quantization operations are combined in one operation using a uniform threshold quantizer as was described in Section 2.5.4 (see Figure 2.5 and Equations (2.8) and (2.10)). In this case, a quantization matrix is used to define the quantizer step size, θ, for each coefficient in the block. A typical quantization matrix is given in Figure 2.10(b). Note that low-frequency coefficients (toward top-left corner) are more finely quantized (i.e., quantized with a smaller step size) because of two reasons. First, the DCT tends to concentrate most of the energy in low frequencies. Second, the HVS is more sensitive to variations in low frequencies. Since in threshold coding the locations of the retained coefficients vary from block to block, those locations need to be encoded. A commonly used strategy is to zigzag scan the transform coefficients, as illustrated in Figure 2.10(c), in an attempt to produce long runs of zeros, and then RLE is used to encode the resulting array.

Compared to predictive coding, transform coding provides higher compression with less sensitivity to errors and less dependence on the input data statistics. Its higher computational complexity and storage requirements have been offset by advances in integrated circuit technology. One disadvantage, however, is that when compression factors are pushed to the limit, three types of artefacts start to occur: (i) "graininess" due to coarse quantization of some coefficients, (ii) "blurring" due to the truncation of high-frequency coefficients, and (iii) "blocking artefacts," which refer to artificial discontinuities appearing at the borders of neighboring blocks due to independent processing of each block. Since blocking artefacts are the most disturbing, a number of methods have been proposed to reduce them. Examples are overlapping blocks at the encoder [34], the use of the lapped orthogonal transform (LOT) [35], and postprocessing using filtering and image restoration techniques [36].

2.6.3 Subband Coding

As already mentioned, rate-distortion theory can provide insights into the design of efficient coders. For example, in Ref. 37 it is shown that the mathematical form of the rate-distortion function suggests that an efficient coder splits the original signal into spectral components of infinitesimal bandwidth and encodes these spectral components independently. This is the basic idea behind subband coding. Subband coding was first introduced by Crochiere *et al.* in 1976 in the context of speech coding [38] and was applied to image coding by Woods and O'Neil in 1986 [39]. In subband coding the input image is passed through a set of bandpass filters to create a set of bandpass images, or *subbands*. Since a bandpass image has a reduced bandwidth compared to the original image, it can be downsampled (subsampled or decimated). This

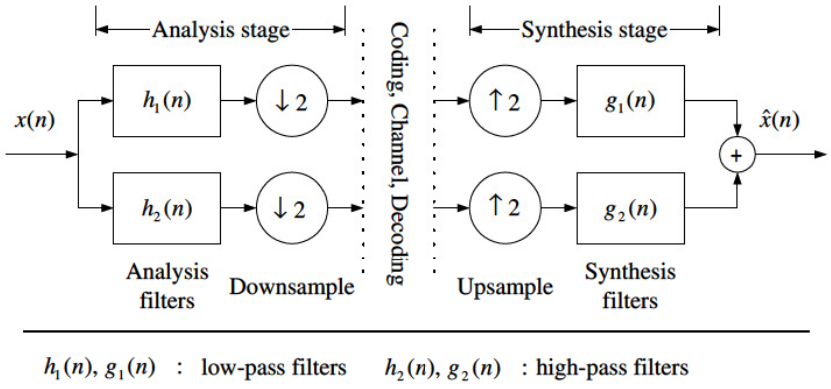
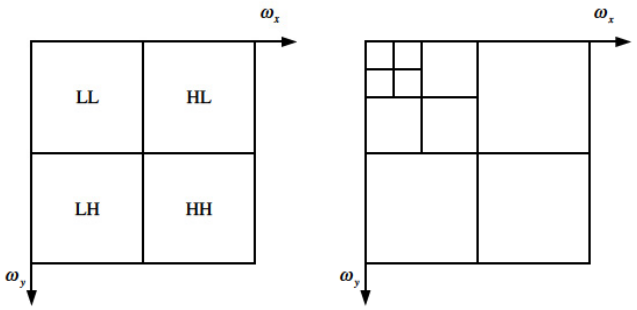


Figure 2.11: A 1-D, two-band subband coding system

process of filtering and downsampling is called the *analysis* stage. The subbands are then quantized and coded independently. At the decoder, the decoded subbands are upsampled (interpolated), filtered, and added together to reconstruct the image. This is knows as the *synthesis* stage. Note that subband decomposition does not lead to any compression in itself, since the total number of samples in the subbands is equal to the number of samples in the original image (this is known as *critical decimation*). The power of this method resides in the fact that each subband can be coded efficiently according to its statistics and visual importance. A block diagram of a basic 1-D, two-band subband coding system is presented in Figure 2.11.

Ideally, the frequency responses of the low-pass and high-pass filters should be nonoverlapping but contiguous and have unity gain over their bandwidths. In practice, however, filters are not ideal and their responses must be overlapped to avoid frequency gaps. The problem with overlapping is that aliasing is introduced when the subbands are downsampled. A family of filters that circumvent this problem is the *quadrature mirror filter* (QMF). In the QMF, the filters are designed in such a way that the aliasing introduced by the analysis stage is exactly cancelled by the synthesis stage.

The 1-D decomposition can easily be extended to 2-D using separable filters. In this case, 1-D filters can be applied first in one dimension and then in the other dimension. Using a 1-D two-band decomposition in each direction results in four subbands: horizontal low/vertical low (LL), horizontal low/vertical high (LH), horizontal high/vertical low (HL), and horizontal high/vertical high (HH), as illustrated in Figure 2.12(a). This four-band decomposition can be continued by repetitively splitting all subbands (uniform decomposition) or just the LL subband (nonuniform decomposition). A three-stage nonuniform decomposition is illustrated in Figure 2.12(b).



(a) One-stage 2-D decomposition

(b) Three-stage 2-D nonuniform decomposition

Figure 2.12: Two-dimensional subband decomposition

Note that nonuniform decomposition results in a multiresolution pyramidal representation of the image. A commonly used technique for nonuniform decomposition is the *discrete wavelet transform* (DWT). The DWT is a transform that has the ability to operate at various scales and resolution levels. Having used the DWT for decomposition, various methods can be used to encode the resulting subbands. One of the most efficient methods is the embedded zero-tree wavelet (EZW) algorithm proposed by Shapiro [40]. This algorithm assumes that if a coefficient at a low-frequency band is zero, it is highly likely that all the coefficients at the same spatial location at all higher frequencies will also be zero and, thus, can be discarded. The EZW algorithm encodes the most important information first and then progressively encodes less important refinement information. This results in an embedded bitstream that can support a range of bit rates by simple truncation. Further refinements to the EZW algorithm have been proposed by Said and Pearlman [41, 42]. In particular, the set partitioning in hierarchical trees (SPIHT) algorithm [42] has become the choice of most practical implementations.

One advantage of subband coding systems is that, unlike transform systems, they do not suffer from blocking artefacts at very low bit rates. In addition, they fit naturally with progressive and multiresolution transmission. One disadvantage, however, is that at very low bit rates, ringing artefacts start to occur around high-contrast edges. This is due to the Gibbs phenomenon of linear filters. To avoid this artefact, subband decomposition using nonlinear filters has been proposed [43, 44].

2.6.4 Vector Quantization

Vector quantization (VQ) is a block-based spatial-domain method that has become very popular since the early 1980s. In VQ, the input image data is

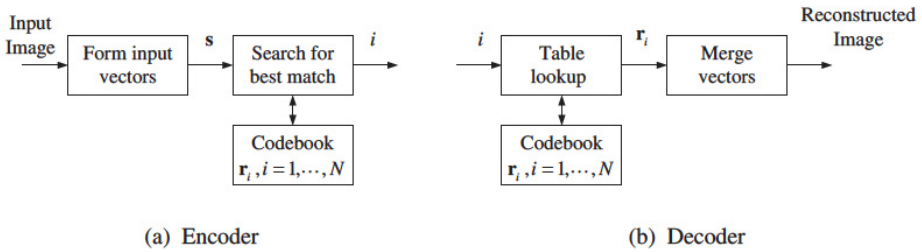


Figure 2.13: A vector quantization system

first decomposed into k-dimensional input *vectors*. Those input vectors can be generated in a number of different ways; they can refer to the pel values themselves or to some appropriate transformation of them. For example, a $k = M \times M$ block of pels can be ordered to form a k-dimensional input vector $\mathbf{s} = [s_1, \ldots, s_k]^T$. In VQ, the k-dimensional space $\mathscr{R}^k$ is divided into N regions, or *cells*, R_i. Any input vector that falls into cell R_i is represented by a representative *codevector* $\mathbf{r}_i = [r_1, \ldots, r_k]^T$. The set of codevectors $\mathscr{C} = \{\mathbf{r}_1, \ldots, \mathbf{r}_N\}$ is called the *codebook*. Thus, the function of the encoder is to *search* for the codevector $\mathbf{r}_i$ that best matches the input vector $\mathbf{s}$ according to some distortion measure $d(\mathbf{s}, \mathbf{r}_i)$. The index i of this codevector is then transmitted to the decoder using at most $I = \log_2 N$ bits. At the decoder, this index is used to *lookup* the codevector from an identical codebook. A block diagram of a vector quantization system is illustrated in Figure 2.13.

Compression in VQ is achieved by using a codebook with relatively few codevectors compared to the number of possible input vectors. The resulting bit rate of a VQ is given by I/k bits/pel. In theory, as $k \to \infty$, the performance of VQ approaches the rate-distortion bound. However, large values of k make codebook storage and searching impractical. Values of $k = 4 \times 4$ and $N = 1024$ are typical in practical systems.

A very important problem in VQ is the codebook design. A commonly used approach for solving this problem is the *Linde-Buzo-Gray* (LBG) algorithm [45], which is a generalization of the Lloyd-Max algorithm for scalar quantization. The LBG algorithm computes a codebook with a locally minimum average distortion for a given training set and given codebook size. Entropy-constrained vector quantization (ECVQ) [46] extends the LBG algorithm for codebook design under an entropy constraint. Another important problem is the codebook search. A full search is usually impractical, and a number of fast-search algorithms have been proposed, e.g., Ref. 47.

There are many variants of VQ [29]. Examples include adaptive VQ, classified VQ, tree-structured VQ, product VQ (including gain/shape VQ, mean/residual VQ, and interpolative/residual VQ), pyramid VQ, and finite-state VQ.

Theoretically, VQ is more efficient than scalar quantization for both correlated and uncorrelated data [48]. Thus, the scalar quantizer in predictive, transform, and subband coders can be replaced with a vector quantizer.

Vector quantization has a performance that rivals that of transform coding. Although the decoder complexity is negligible (a lookup table), the high complexity of the encoder and the high storage requirements of the method still limit its use in practice. Like transform coding, VQ suffers from blocking artefacts at very low bit rates.

2.6.5 Second-Generation Coding

The coding methods discussed so far are generally known as *waveform coding methods*. They operate on pels or blocks of pels based on statistical image models. This classical view of the image coding problem has three main disadvantages. First, it puts more emphasis on the codeword assignment (using information and coding theory) rather than on the extraction of representative messages. Because the encoded messages (pels or blocks) are poorly representative in the first place, a saturation in compression is eventually reached no matter how good is the codeword assignment. Second, the encoded entities (pels or blocks) are consequences of the technical constraints in transforming scenes into digital data, rather than being real entities. Finally, it does not place enough emphasis on exploiting the properties of the HVS. Efforts to utilize models of the HVS and to use more representative coding entities (real objects) led to a new class of coding methods known as the *second-generation coding methods* [49].

Second-generation methods can be grouped into two classes: *local-operator-based techniques* and *contour/texture-oriented techniques*. Local-operator-based techniques include *pyramidal coding* and *anisotropic nonstationary predictive coding*, whereas the contour/texture-oriented techniques include *directional decomposition coding* and *segmented coding*. Two commonly used segmented coding methods are *region-growing* and *split-and-merge*. For a detailed discussion of second-generation methods, the reader is referred to Refs. 49, 50, 51.

Second-generation methods provide higher compression than waveform coding methods at the same reconstruction quality. They also do not suffer from blocking and blurring artefacts at very low bit rates. However, the extraction of real objects is both difficult and computationally complex. In addition, such methods suffer from unnatural contouring effects, which can make the details seem artificial.

2.6.6 Other Coding Methods

There are many other intraframe coding techniques. Examples are block-truncation coding, fractal coding, quad-tree and recursive coding, multiresolution coding, and neural-network-based coding. A detailed (or even a brief) discussion of such techniques is beyond the scope of this book, and the interested reader is referred to Ref. 52.

2.7 Interframe Coding

As already discussed, video is a time sequence of still images or frames. Thus, a naive approach to video coding would be to employ any of the still-image (or intraframe) coding methods discussed in Section 2.6 on a frame-by-frame basis. However, the compression that can be achieved by this approach is limited because it does not exploit the high temporal correlation between the frames of a video sequence. *Interframe coding* refers to video coding techniques that achieve compression by reducing this temporal redundancy. For this reason, such methods are also known as *temporal redundancy reduction* techniques. Note that interframe coding may not be appropriate for some applications. For example, it would be necessary to decode the complete interframe coded sequence before being able to randomly access individual frames. Thus, a combined approach is normally used in which a number of frames are intraframe coded (I-frames) at specific intervals within the sequence and the other frames are interframe coded (predicted or P-frames) with reference to those anchor frames. In fact, some systems switch between interframe and intraframe within the same frame.

2.7.1 Three-Dimensional Coding

The simplest way to extend intraframe image coding methods to interframe video coding is to consider 3-D waveform coding. For example, in 3-D transform coding based on the DCT, the video is first divided into blocks of $M \times N \times K$ pels (M, N, K denote the horizontal, vertical, and temporal dimensions, respectively). A 3-D DCT is then applied to each block, followed by quantization and symbol encoding, as illustrated in Figure 2.14. A 3-D coding method has the advantage that it does not require the computationally intensive process of motion estimation (as will be discussed in Section 2.7.2). However, it requires K frame memories both at the encoder and decoder to buffer the frames. In addition to this storage requirement, the buffering process limits the use of this method in real-time applications because encoding/decoding cannot begin until all of the next K frames are available. In practical systems, K is typically set to 2–4 frames.

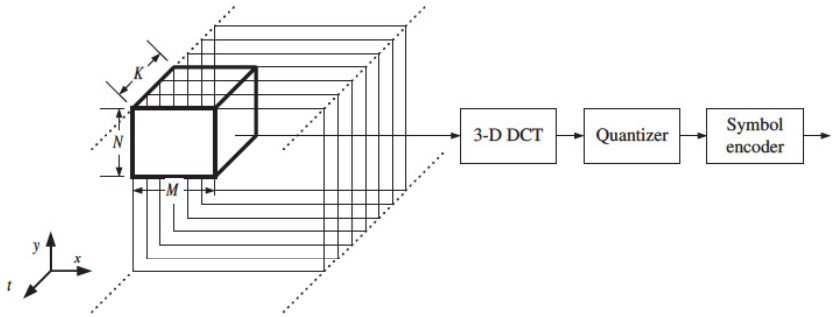


Figure 2.14: A 3-D transform coding system

2.7.2 Motion-Compensated Coding

One of the earliest approaches to interframe coding was *conditional replenishment* (CR) [53]. In this method, the input frame is divided into "changed" and "unchanged" regions with respect to a previously decoded reference frame, and the addresses of this segmentation are coded. Unchanged regions need not be coded because they can simply be copied from the reference frame, whereas changed regions need to be coded. One way of coding them is to use one of the intraframe coding methods discussed in Section 2.6. However, a more efficient approach is to predictively code them with respect to the corresponding regions in the reference frame. In this case, the coded prediction error signal is called the *frame difference* (FD) and the process is known as *frame differencing*.

An improved performance can be obtained by improving the prediction of changed regions. This can be achieved using motion estimation and compensation. Changes between frames are mainly due to the movement of objects. Using a model of the motion of objects between frames, the encoder estimates the motion that occurred between the reference frame and the current frame. This process is called *motion estimation* (ME). The encoder then uses this motion model and information to move the contents of the reference frame to provide a better prediction of the current frame. This process is known as *motion compensation* (MC), and the prediction so produced is called the *motion-compensated prediction* (MCP) or the *displaced-frame* (DF). In this case, the coded prediction error signal is called the *displaced-frame difference* (DFD). A block diagram of a motion-compensated coding system is illustrated in Figure 2.15. This is the most commonly used interframe coding method.

The reference frame employed for ME can occur temporally before or after the current frame. The two cases are known as *forward prediction* and *backward prediction*, respectively. In *bidirectional prediction*, however, two reference frames (one each for forward and backward prediction) are employed

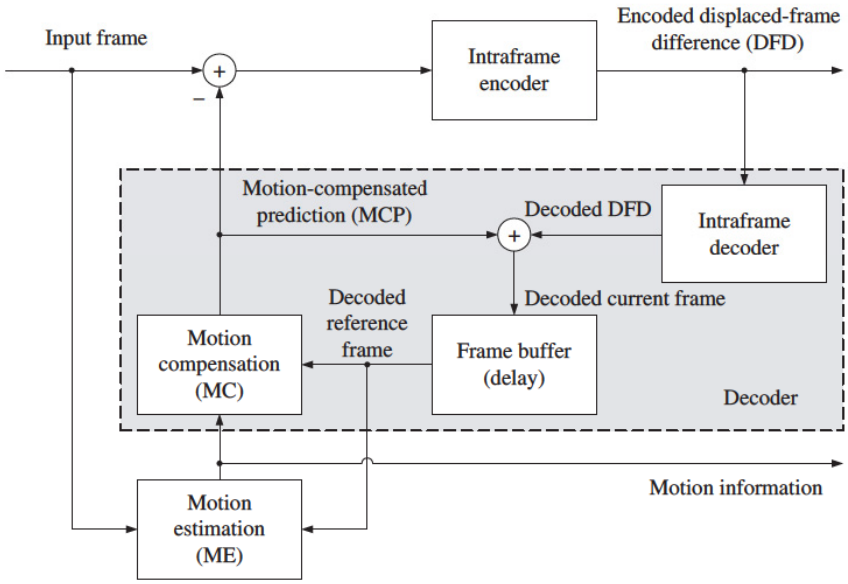


Figure 2.15: Motion-compensated coding system

and the two predictions are interpolated (the resulting predicted frame is called B-frame). The most commonly used ME method is the *block-matching motion estimation* (BMME) algorithm [54]. In this algorithm, the current frame is first divided into blocks. The motion of each block is then estimated by searching for the *best-match* block in the reference frame according to some distortion measure. This search is usually restricted to a *search window* centered around the corresponding block in the reference frame. The motion of the current block is then represented by a *motion vector*, which is the displacement between the block and its best-match block in the reference frame. The process of BMME is illustrated in Figure 2.16. Note that this algorithm is based on a *translational* model of the motion of objects between frames. It also assumes that all pels within a block undergo the same translational movement. There are many other ME methods, but BMME is normally preferred due to its simplicity and good compromise between prediction quality and motion overhead [55]. A more detailed discussion of BMME and other ME methods is deferred to Chapter 4.

As illustrated in Figure 2.15, the DFD signal can be coded using any of the intraframe coding methods discussed in Section 2.6. However, the most commonly used method is transform coding, in particular block-based DCT transform coding. This combination of block-matching motion-compensated prediction and block-based DCT coding of the prediction error has proved to be the most successful class of video coding methods. Today, most video

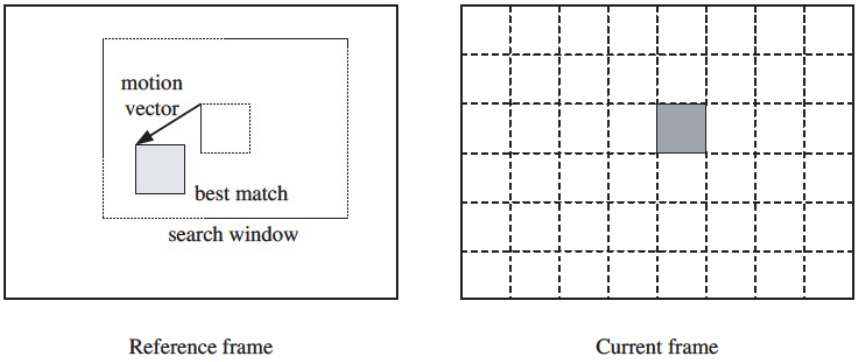


Figure 2.16: Block-matching motion estimation

coding standards are based on this so-called *hybrid* MC-DPCM/DCT coding method.

2.7.3 Model-Based Coding

At very low bit rates (below 64 kbits/s), the quality produced by conventional motion-compensated coding methods may be unacceptable for some applications [10]. In particular, at such bit rates, decoded frames using MC-DPCM/DCT generally suffer from blocking artefacts. This is mainly due to the translational block-based motion model. This has initiated research efforts into new motion-compensated methods based on more realistic structural motion models. Such methods are referred to as *model-based coding* methods.

Model-based coding is also known as *analysis-synthesis coding*, because it is characterized by two main processes: *analysis* and *synthesis*. Both processes usually make extensive use of sophisticated computer vision and computer graphics tools. At the encoder, the image sequence is initially segmented into a number of objects. Each object is then *analyzed* to decide its location, shape, and texture. The encoder then uses this analysis data to deform a general model to *synthesize* an approximation of the object. The same analysis data is also transmitted to the decoder to synthesize a similar approximation. When the object starts moving, tracking techniques are used, at the encoder, to estimate the associated animation data, which is then transmitted to the decoder to animate the same object. While animation data is sufficient for low quality reproduction at low bit rates, residual data can also be transmitted to achieve higher quality reproduction, but at the expense of a higher bit rate. Thus, once the whole scene is synthesized, only a few animation parameters and possibly some texture information need to be encoded. Hence, model-based

coding offers a potential saving in bit rate, which makes it attractive for very-low-bit-rate applications.

Model-based coding methods can be broadly classified as *object-based* or *knowledge-based*. Object-based coding methods deal with unknown (arbitrary) objects, whereas knowledge-based coding methods assume *a priori* knowledge of the objects being modeled (e.g., a 3-D wireframe face model is usually employed for head-and-shoulders sequences typical of videophone applications). Knowledge-based coding methods are generally successful in tracking the global motion of the object (e.g., rotation and translation of the head), but suffer from errors in estimating local motion (e.g., the movement of the eyes, lips, and so on). *Semantic-based coding* is a subset of knowledge-based coding methods that models local motion using a set of action units (e.g., a combination of facial action units can lead to a given facial expression).

Despite their good performance at very low bit rates, model-based coding methods have their problems. For example, at lower bit rates, the analysis and modeling processes become more complex and the model needs to be more object specific. In addition, the analysis and tracking methods usually require some degree of human intervention or some *a priori* assumptions about the nature of tracked objects. Another problem is that, in some cases, severe or sustained failure of tracking or modeling may occur, leading to an increase in the bit rate or a deterioration in the video quality. However, continuous research efforts in this area are addressing such problems. For example, switched model-based coders, with a fallback mode to conventional coding, have been proposed to solve the problem of model or tracking failure [56]. For a good review of model-based coding, the reader is referred to Ref. 57.

Chapter 3

Video Coding: Standards

3.1 Overview

This chapter gives a brief introduction to video coding standards. Section 3.2 highlights the need for video coding standards. Section 3.3 outlines the chronological development of video coding standards, highlighting their main techniques and targeted applications. The chapter then gives two examples of the state-of-the-art video coding standards: Section 3.4 concentrates on H.263 (and its recent extensions: H.263+ and H.263++), whereas Section 3.5 describes MPEG-4.

3.2 The Need for Video Coding Standards

For the past 25 years or so, the efficient coding of image and video signals has been the subject of considerable research. Over the years, the field has matured and has become a key enabling technology for a wide range of applications spanning a wide range of industries. This has moved the field from being a purely academic research area to become a highly commercial business. This increased commercial interest has ignited the efforts of international standardization of image and video coding. International standards enable image and video material from different sources and industries to be processed on different hardware platforms, to be stored on different storage devices, and to be transmitted on different communication networks. This interoperability opens a huge market for video equipment and at the same time gives consumers a wide range of services. International standards also allow for large scale production at considerably reduced costs.

3.3 Chronological Development

Video coding standardization activities started in the early 1980s. The activities were initiated by the *International Telegraph and Telephone Consultative Committee* (CCITT), which is currently known as the *International Telecommunications Union—Telecommunication Standardization Sector* (ITU-T). This was later followed by CCIR (currently ITU-R), the *International Organization for Standardization* (ISO), and the *International Electrotechnical Commission* (IEC). This has resulted in a number of standards, some of which are discussed here.

3.3.1 H.120

The first video coding international standardization activity was carried out by Study Group (SG) XV of CCITT during its study period 1980–1984. In 1984 it issued Recommendation H.120 in its first version, and in 1988 it issued the second version [58]. The standard was targeted for videoconferencing applications at the digital primary rates of 1.544 Mbits/s and 2.048 Mbits/s. The standard had three parts: Part 1 for 625/50 regional use at 2 Mbits/s, Part 3 for 525/60 regional use at 1.5 Mbits/s, and Part 2 for international use (both 525/60 and 625/50 at 1.5 Mbits/s). Parts 1 and 2 use CR with intrafield DPCM for changed regions, whereas Part 3 uses intrafield prediction, background prediction,[1] and motion compensated interfield prediction. This difference in coding techniques between the different parts was one of the reasons why H.120 never became a commercial success.

3.3.2 H.261

At the end of 1984, CCITT/SG XV agreed to define a standard targeted for videophone and videoconferencing applications at ISDN subprimary rates ($\leq$2 Mbits/s). Initially, it was thought that there would be two different algorithms efficient at 64 kbits/s or higher and 384 kbits/s or higher, respectively. It was found, however, that a single algorithm could cover all these rates. Thus, H.261 was drafted in 1989 to provide audiovisual services at $p \times 64$ kbits/s ($p = 1 \ldots 30$). This draft became an international standard in 1991 and was later revised in 1993 [59]. H.261 was the first widespread commercial success. In fact, its adopted techniques of hybrid MC-DPCM/DCT (16×16 macroblocks for MC and 8×8 blocks for DCT), SKIP/INTER/

[1] None of the later standards have included a background prediction mode, although sprite coding in MPEG-4 can be considered a form of background prediction.

INTRA mode switching on a macroblock level, zigzag scanning, RLE, scalar quantization, and VLC entropy coding have become key elements in most video coding standards.

3.3.3 CCIR-721

In parallel to the standardization activities of CCITT, CCIR started standardization of video coding for contribution-quality TV signals. Recommendation 721 [60] was issued in 1990. Its main target was the transmission of component coded digital TV signals for contribution-quality applications at bit rates near 140 Mbits/s. The recommendation used a simple form of intrafield DPCM to achieve a low implementation complexity and a high degree of random access (which is important for video postprocessing in studios).

3.3.4 CCIR-723

CCIR Recommendation 723 [61] was issued in 1992. Its main target was the coding of component digital TV signals for contribution-quality applications in the range 34–45 Mbits/s. The recommendation employs a hybrid MC-DPCM/DCT with one intrafield mode and two interfield modes (with and without MC). Both the CCIR-721 and the CCIR-723 recommendations are not generic. In contrast to other standards, they fully specify both the encoder and the decoder.

3.3.5 MPEG-1

In 1988, the *Moving Picture Experts Group* (MPEG) was created under Subcommittee (SC) 2 of ISO (ISO/SC2). The group is now Working Group (WG) 11 of SC29 under the Joint Technical Committee (JTC) 1 of ISO/IEC. Thus, its official denotation is ISO/IEC JTC1/SC29/WG11. The main aim of the group was to develop a video coding standard for digital media storage applications at up to 1.5 Mbits/s. In 1991 the group drafted its ISO/IEC 11172 (MPEG-1) standard [62], which became an international standard in 1992. The MPEG-1 video algorithm is very similar to the H.261 algorithm but with some advanced techniques, like bidirectional prediction and half-pel MC[2]. The standard also provides for some specific storage requirements, like random access and fast forward/reverse searches. Although the standard was developed mainly for storage applications, it was designed to be generic. Thus, it was

[2]Half-pel MC was proposed during the development of the H.261 but was thought to be too complex at that time.

designed as a toolbox, where the user can decide which tools to use for the particular application. In addition, the standard defines only the decoder and the bitstream syntax. This allows a large degree of freedom for manufacturers to propose their own optimized encoders. This generic design and large degree of freedom have contributed to the success of MPEG-1. It has been used in a wide range of applications, from interactive systems on CD-ROM to the delivery of video over telecommunication networks.

3.3.6 MPEG-2

In 1990, ISO/IEC JTC1/SC29/WG11 started studies on a new standard for applications not covered by MPEG-1. In particular, the new standard was intended to provide video quality not lower than NTSC/PAL and up to CCIR-601 quality at rates around 10 Mbits/s. This standardization activity was nicknamed MPEG-2 because it was seen as phase 2 of the work started in MPEG-1. In 1992, ITU-T/SG 15 joined this standardization effort to develop video coding for *Asynchronous Transfer Mode* (ATM) networks. In 1993, it was realized that the scope of MPEG-2 could be enlarged to suit coding of HDTV. This made an initially planned MPEG-3 for HDTV superfluous. In 1994, the ISO/IEC 13818 (MPEG-2) standard (ITU-T Recommendation H.262) was drafted [63], and later in the year it was accepted as an international standard. Like MPEG-1, the MPEG-2 standard is generic and flexible. In fact, MPEG-2 can be thought of as a superset of, and as such was designed to be backward compatible with, MPEG-1. There are many additional features provided by MPEG-2 over MPEG-1, including the support for interlaced video and scalability. Since implementation of the full MPEG-2 syntax may not be practical for most applications, MPEG-2 has introduced the concepts of "profiles," describing functionalities, and "levels," describing resolutions, to provide subset conformance levels. MPEG-2 has had even more success than MPEG-1, with applications in the areas of cable TV, networked ATM services, and satellite and terrestrial TV broadcasting.

3.3.7 H.263

The increasing demand for digital video communications over the public switched telephone network (PSTN) and mobile networks initiated a new standardization effort by ITU-T/SG 15. The aim was to develop a video coding standard for low-bit-rate applications below 64 kbits/s. The result of this effort was ITU-T Recommendation H.263 [64], which was completed in 1995 and approved in 1996. Although H.263 was based on the coding structure of H.261, it provides a significant improvement in performance. Side-by-side

comparisons indicate that H.263 provides the same subjective quality as H.261 but with less than half the bit rate [65]. This performance improvement is due to optimized coding techniques as well as advanced optional coding modes. Some of the new features of H.263 compared to H.261 are the support for more picture formats, half-pel MC, a 3-D (LAST-RUN-LEVEL) RLE instead of 2-D (RUN-LEVEL), more optimized VLC tables, optional extra headers to increase error resilience, advanced 2-D median predictor for motion vector coding, more optimized macroblock addressing and quantization adaptation, optional extended-range unrestricted motion vectors that can point outside frames, optional arithmetic coding, optional advanced prediction with overlapped motion compensation and four motion vectors per macroblock, and optional bidirectional prediction. H.263 is described in more detail in Section 3.4.

3.3.8 H.263+

Technically, H.263+ is version 2 of the H.263 standard [66]. This version was developed by ITU-T/SG16/Q15 Advanced Video Experts Group (previously under ITU-T/SG15), with technical content completed in 1997 and approved in 1998. The H.263+ standard added 12 new optional features to H.263. These new features support custom picture size and clock frequency, improve compression efficiency, allow scalability, enhance error resilience over wireless and packet-based networks, provide supplemental display and external usage capabilities, and ensure backward compatibility. H.263+ is described in more detail in Section 3.4.

3.3.9 MPEG-4

In 1993, the ISO/IEC JTC1/SC29/WG11 MPEG group initiated a new standardization activity called MPEG-4. The target was the very-low-bit range and the aim was to achieve higher compression efficiency than could be achieved by existing conventional techniques. In 1994, it was realized that too few improvements could be achieved over the H.263 and H.263+ compression results to justify a new standard. Thus, the group decided to broaden the objectives of the MPEG-4 effort and started an in-depth analysis of the trends within the audiovisual world. Particular attention was given to the convergence of the three traditionally separate industries of communications, computing, and TV/film/entertainment. This study concluded that MPEG-4 should support functionalities that would be useful in future applications but were not supported or not well supported by the available standards. Eight main new or improved functionalities were identified and then clustered in three classes:

content-based interactivity (content-based multimedia data-access tools, content-based manipulation and bitstream editing, hybrid natural and synthetic data coding, improved temporal random access), *compression* (improved coding efficiency, coding of multiple concurrent data streams), and *universal access* (robustness in error-prone environments, content-based scalability). Version 1 of the MPEG-4 standard was approved in October 1998. A second version was approved in December 1999 to add new functionalities and improve others. The MPEG-4 standard is officially known as ISO/IEC 14496 and is titled "Generic coding of audiovisual objects" [67]. This title describes two important properties of the MPEG-4 standard. The first property is that it is a generic standard. It is designed to cover a wide range of bit rates (typically, 5 kbits/s to 10 Mbits/s), picture formats (progressive and interlaced), resolutions (SQCIF to beyond TV), frame rates (still images to high frame rates), communication networks (wired or wireless), input material (natural or synthesized), etc. The second property is that it uses an object-based representation model, where a scene is represented, coded, and manipulated as individual audiovisual objects. This particular property (i.e., being object-based) sets MPEG-4 apart from earlier block-based standards. Thus, in addition to conventional block-based MC-DPCM/DCT techniques, MPEG-4 adopts more recent object-based techniques like second-generation coding techniques (Section 2.6.5) and model-based coding techniques (Section 2.7.3). MPEG-4 is described in more detail in Section 3.5.

3.3.10 H.263++

Technically, H.263++ is version 3 of the H.263 standard [68]. This version was developed by ITU-T/SG16/Q15, with technical content completed and approved late in the year 2000. The H.263++ standard added some more features to H.263 and H.263+. These new features improve coding efficiency, enhance error resilience, provide additional supplemental display and external usage capabilities, and define profiles and levels. H.263++ is described in more detail in Section 3.4.

3.3.11 H.26L

This is a project of ITU-T/SG16/Q15. The H.26L project is planned to be a new-generation video coding standard with improved efficiency, error resilience, and streaming support. It is scheduled for completion in 2002.

In addition to the standard documents themselves, interested readers are referred to some excellent reviews and tutorials available in the literature [69, 70, 65, 71–75, 11, 13, 15].

3.4 The H.263 Standard

3.4.1 Introduction

The H.263 recommendation specifies a coded representation that can be used for compressing the moving picture component of audiovisual services at low bit rates. The recommendation fully specifies the decoder and the bitstream syntax but does not explicitly specify the encoder. As already mentioned, this gives manufacturers a large degree of freedom to propose their own optimized encoders, as long as the output bitstream conforms to the standard decodable syntax. However, during the standardization process, a software-based codec (encoder-decoder) called the *test model* is developed to study the core elements of the standard. For example, version 5 of the *test model near-term* (TMN) is described in Ref. 76.

3.4.2 Source Format

The standard supports all five members of the CIF family described in Section 2.4.4 and Table 2.1. As a minimum requirement, all decoders shall be able to operate with SQCIF and QCIF. Encoders, on the other hand, shall be able to operate with either SQCIF or QCIF and are not obliged to be able to operate with both.

3.4.3 Video Source Coding Algorithm

The generalized form of the source coder is illustrated in Figure 3.1. It is a hybrid of interpicture prediction to utilize temporal redundancy and transform coding of the error signal to reduce spatial redundancy.

3.4.3.1 Picture Coding Structure

The input video consists of a sequence of pictures (or frames). Each picture is divided into *groups of blocks* (GOBs). A GOB consists of $k \times 16$ lines, depending on the picture format ($k = 1$ for SQCIF, QCIF, and CIF, $k = 2$ for 4CIF, and $k = 4$ for 16CIF). For example, there are 9 GOBs in a QCIF picture. Each GOB is divided into *macroblocks* (MBs). A macroblock consists of 16×16 samples of Y' and the spatially corresponding 8×8 samples of C'_B and C'_R. If we define a *block* as 8×8 samples of Y', C'_B, or C'_R, then a macroblock consists of 6 blocks: 4 luma blocks and the 2 spatially corresponding chroma blocks. Figure 3.2 illustrates the H.263 picture structure for a QCIF frame. As shown, GOBs are coded from top to bottom in increasing number. Within each GOB, the MBs are coded from left to right (and from top to bottom if the GOB contains more than one row of MBs) in increasing number. Within

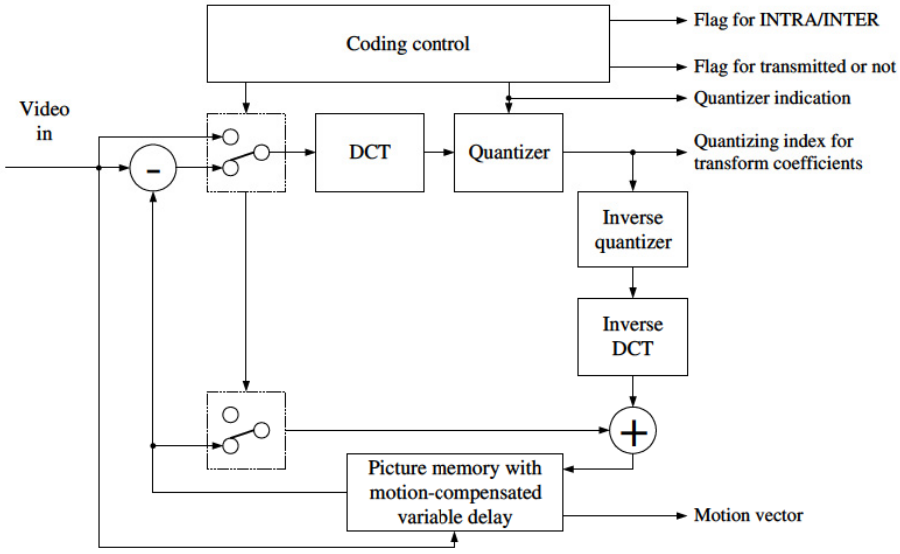


Figure 3.1: H.263 video encoder

each MB, the Y' blocks are first coded in the order shown (left to right and top to bottom), followed by the C'_B block and then the C'_R block.

3.4.3.2 Coding Modes

The coding mode in which interpicture prediction is applied is called the INTER mode. Prediction can optionally be augmented by motion compensation. If no prediction is applied, then the coding mode is called INTRA. The coding mode (INTRA/INTER) can be signaled at the picture level (resulting in I-pictures/P-pictures) or at the macroblock level in P-pictures. In PB-frames (discussed later) the B-pictures are always coded in INTER mode. The mode selection method is not defined by the standard. However, to control the accumulation of IDCT mismatch[3] error, the standard requires an MB to be coded in INTRA mode at least once every 132 times when coefficients are transmitted for this MB in P-pictures. In the INTER mode, a flag is used to indicate whether an MB is transmitted or not (conditional replenishment). This is sometimes referred to as the SKIP mode. Again, the method of reaching a

[3]The *inverse discrete cosine transform* (IDCT) is a common block between the encoder and the decoder. Differences in implementation between the encoder's IDCT and the decoder's IDCT cause mismatches between the reconstructed pictures at both ends. This is called the *IDCT mismatch error*. This mismatch accumulates due to interpicture prediction and can be stopped by forced INTRA updating.

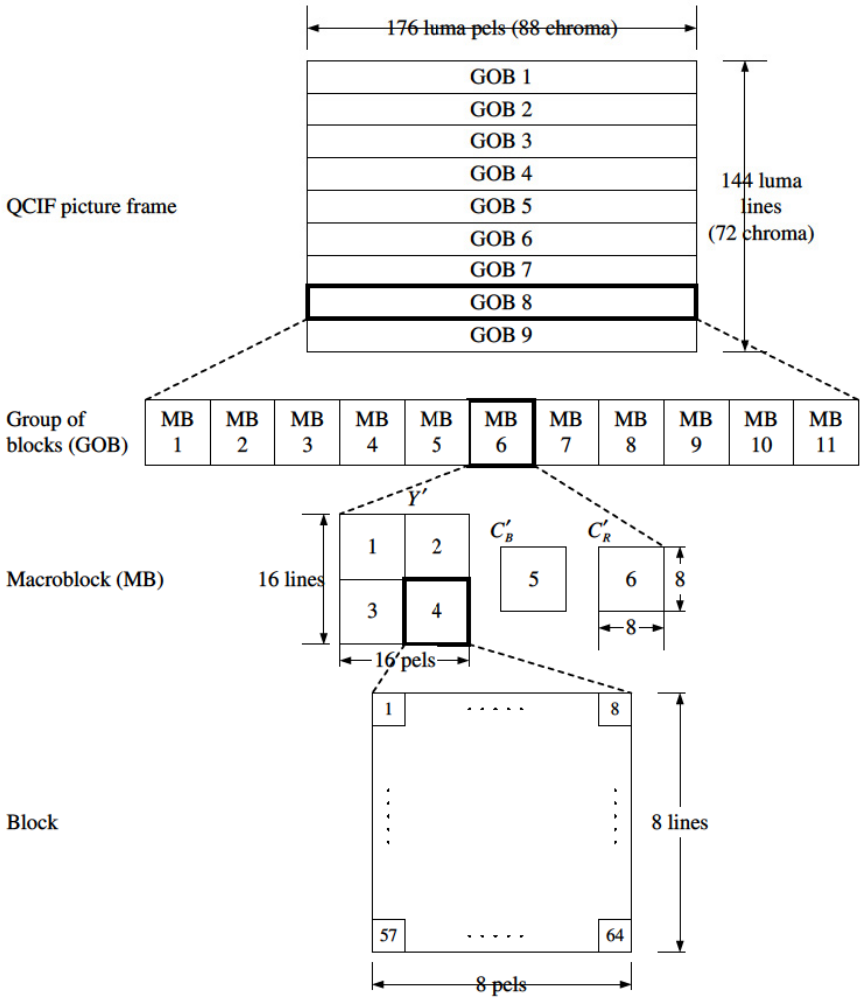


Figure 3.2: H.263 picture structure for a QCIF frame

decision to transmit an MB or not is not part of the standard. The different flags are encoded within the picture and MB headers.

3.4.3.3 Motion Estimation and Compensation

Without options, the encoder estimates one motion vector per MB. Both horizontal and vertical components of the vector have integer or half-integer values and are restricted to the range $[-16, +15.5]$. A positive value of the horizontal or vertical component means that the prediction is made from pels in the reference picture that are spatially to the right or below the pels being

predicted, respectively. Motion vectors are restricted such that all pels referenced by them are within the reference picture area. The standard does not explicitly specify an ME method. However, this MB-based structure implicitly supports block-based approaches and in particular the BMME algorithm.

3.4.3.4 Motion Vector Coding

The estimated motion vector $\mathbf{MV} = (\mathrm{MVx}, \mathrm{MVy})$ is predictively coded. This means that the motion vector difference $\mathbf{MVD} = \mathbf{MV} - \mathbf{MVP}$ is encoded instead of the motion vector itself. The motion vector predictor **MVP** is the median of three candidate predictors, which are the motion vectors of three surrounding macroblocks, as illustrated in Figure 3.3(a). The two components of the motion vector difference are then entropy coded using a standard VLC table. MVDx is encoded first, followed by MVDy.

3.4.3.5 Forward Transform

The forward DCT transform is applied either to the pel values, in the case of an INTRA MB, or to the DFD values, in the case of an INTER MB. In both cases, the DCT is applied on a block (8×8) basis. This results in six blocks of transform coefficients for each MB. The standard does not specify

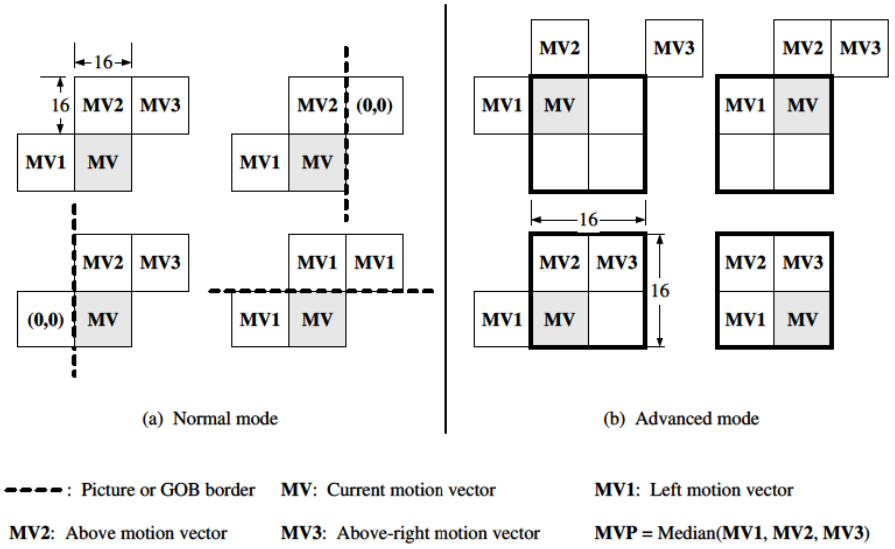


Figure 3.3: H.263 motion vector prediction

the method of implementing the forward DCT. Threshold coding, discussed in Section 2.6.2, is used to allocate bits to the transform coefficients, as will be discussed next.

3.4.3.6 Quantization

The six DC coefficients of an INTRA MB are quantized using a uniform scalar quantizer with a step size of $\theta = 8$ and no dead zone (this corresponds to Figure 2.5(a) and Equation (2.8)). All other coefficients are quantized using a uniform scalar quantizer with a step size of $\theta = 2 \times \text{QP}$ and a central dead zone around zero (this corresponds to Figure 2.5(b) and Equation (2.10)). There are 31 possible quantization parameters, $\text{QP} = 1 \ldots 31$. However, the quantization parameter is kept fixed for all coefficients within an MB. A high QP leads to higher compression but worse quality, whereas a low QP leads to better quality but less compression. The method to select a QP is not part of the standard. A change of QP to any of the 31 permissible values can be signaled in the picture or GOB headers. In the MB header, however, this change is limited to a maximum of ± 2. Again, the method to decide this change is not defined in the standard.

3.4.3.7 Quantized Coefficients Coding

A quantized INTRA DC coefficient is encoded using a standard 8-bit FLC table. Other quantized coefficients are first zigzag scanned, as described in Section 2.6.2 and Figure 2.10(c). The reordered coefficients are then encoded using 3-D RLE. Thus, the reordered coefficients are converted to an intermediate set of symbols or EVENTS of the form (LAST, RUN, LEVEL), where LAST is an indication of whether this is the last nonzero coefficient in the block or not, RUN is the number of successive zeros preceding the coded coefficient, and LEVEL is the nonzero value of the coded coefficient. The most commonly occurring EVENTs are coded using a standard VLC table, whereas the remaining EVENTs are coded using a concatenation of four standard FLC codewords for ESCAPE, LAST, RUN and LEVEL.

3.4.3.8 Coding Control

The coding control block is responsible for varying several parameters to control the rate or the quality of the coded video. Examples are the INTER/INTRA mode decision at the picture or MB level, the update pattern of the forced INTRA refresh, the TRANSMIT/SKIP decision at the MB level, and the QP and its change at the picture, GOB, or MB level. Such functions are not defined in the standard.

3.4.4 Decoding Process

3.4.4.1 Motion Vector Decoding

For each TRANSMITTED INTER MB, the decoder calculates the same motion vector predictor **MVP** used at the encoder and adds it to the decoded motion vector difference **MVD** to obtain the decoded motion vector **MV**. The motion vector of a SKIPPED INTER MB is set to **0**.

3.4.4.2 Motion Compensation

The decoded motion vector is used to compensate the four Y' blocks in the MB. Motion vectors for both C'_R and C'_B blocks are derived by dividing the component values of the decoded motion vector by 2. The resulting quarter-pel resolution components are modified toward the nearest half-pel resolution (both 0.25 and 0.75 are rounded to 0.5). If motion compensation requires accessing half-pel positions, then bilinear interpolation is used to calculate the pel values at those positions.

3.4.4.3 Inverse Quantization

As already discussed, quantization is achieved by dividing the transform coefficient by a quantization step size and rounding the result (refer to Equations (2.8) and (2.10)). Inverse[4] quantization is the process of reconstructing an approximation of the original coefficient by multiplying the quantized coefficient by the same step size (refer to Equations (2.9) and (2.11)). The reconstructed coefficients are then clipped to the range $[-2048, +2047]$ and inverse zigzag scanned to put them in an 8×8 block.

3.4.4.4 Inverse Transform

The reconstructed block of coefficients is processed by a separable 2-D 8×8 inverse DCT. The arithmetic procedures for computing the inverse DCT are not defined by the standard, but should meet a defined error tolerance.

3.4.4.5 Reconstruction of Blocks

For INTRA blocks, the reconstructed block is equal to the result of the inverse DCT. For INTER blocks, the reconstructed block is formed by summing the motion-compensated prediction and the result of the inverse DCT. The reconstructed values are clipped to the range $[0, 255]$.

[4]It should be emphasised that the term *inverse* here does not mean that quantization is a reversible process. Quantization is irreversible since rounding leads to loss of information.

3.4.5 Optional Coding Modes

There are four optional coding modes that can be signaled at the picture level. These modes are defined in annexes to the standard and are briefly described next.

3.4.5.1 Unrestricted Motion Vector Mode (Annex D)

In this mode, motion vectors are allowed to point outside the reference picture area. When a pel pointed to by a motion vector is outside the reference picture area, an edge pel is used instead. This edge pel is found by limiting the motion vector to the last full-pel position inside the reference picture area. Limitation of the motion vector is performed on a pel basis and separately for each component of the motion vector. In this mode also, the range for motion vector components is extended to $[-31.5, +31.5]$, with the restriction that if the predictor is in the range $[-15.5, +16]$, then only values that are within a range of $[-16, +15.5]$ around the predictor can be reached. If, however, the predictor is outside $[-15.5, +16]$, then all values within the range $[-31.5, +31.5]$ with the same sign as the predictor can be reached. Allowing motion vectors to point outside the reference picture area improves prediction along picture edges in the case of camera or background movement. This is particularly useful for small picture formats (where border MBs represent a high percentage of the picture area). The extended motion vector range allows better prediction for large picture formats and a high amount of movement.

3.4.5.2 Syntax-Based Arithmetic Coding Mode (Annex E)

In this mode, all VLC Huffman coding/decoding operations of H.263 are replaced with arithmetic coding/decoding operations. As already discussed in Section 2.5.5, arithmetic coding removes the restriction of representing each symbol by an integral number of bits, achieving more coding efficiency but at the expense of more computational complexity.

3.4.5.3 Advanced Prediction Mode (Annex F)

This optional mode includes two advanced prediction techniques: the use of four motion vectors per MB, and the use of *overlapped motion compensation* (OMC). In addition, this mode allows motion vectors to point outside the reference picture area. If this mode is used in combination with the unrestricted motion vector mode, then the motion vectors will also have an extended range. If the mode is used in combination with the PB-frames mode, then OMC is only used for P-pictures, not for B-pictures.

In this mode, the encoder makes a decision (which is not defined by the standard) whether to transmit one motion vector or four motion vectors per

MB. If one motion vector is transmitted (as in normal mode), then the decoder replicates it to four motion vectors. If four motion vectors are to be transmitted, then the motion vector prediction process is modified as illustrated in Figure 3.3(b). Motion vectors for chroma blocks are derived by calculating the sum of the four luma vectors and then dividing by 8. The resulting values of 1/16-pel resolution are modified toward the nearest 1/2-pel values (0, 1/16, and 2/16 are modified to 0; 14/16 and 15/16 are modified to 1; and all other values are modified to 1/2). This technique improves prediction if the MB contains different moving objects.

In OMC, each pel in an 8×8 luma prediction block is predicted as a weighted sum of three prediction values. To obtain the three prediction values, three motion vectors are used: the motion vector of the current luma block, and two out of four *remote* motion vectors. The four remote motion vectors are the motion vectors of the luma blocks to the left of, to the right of, above, and below the current luma block. The position of the pel within the block decides which two remote vectors to use. For example, all pels in the top-left quadrant of the block use the two remote vectors of the blocks above and to the left of the current luma block. The weight given to each of the three predictions also changes with pel position within the block. The weights are defined in three standard matrices. The weights for a remote prediction are designed to increase as the pel position moves away from the center of the block toward the corresponding remote block. This ensures a smooth transition at the borders of the block, which results in a visible reduction of blocking artefacts. If a remote MB was not coded, then the corresponding vector is set to zero. If a remote MB does not exist (out of the picture) or was INTRA coded, then the corresponding vector is set to the vector of the current MB. In PB-frames mode, however, INTRA MBs have motion vectors, and those are used as remote vectors. For chroma blocks, no overlapping is performed.

3.4.5.4 PB-Frames Mode (Annex G)

In this mode, two pictures are encoded as one unit called a PB-frame. Thus, a PB-frame consists of one P-picture that is predicted from the previous decoded P-picture (forward prediction) and one B-picture that is predicted from both the previous decoded P-picture and the P-picture currently decoded in the same PB-frame (bidirectional prediction). In a PB-frame, an MB consists of 12 blocks: the 6 blocks of the P-picture, followed by the 6 blocks of the B-picture. In this mode, an INTRA coding mode can also be used where P-blocks are INTRA coded and B-blocks are INTER coded with prediction as for an INTER block. In this case, motion vector data is included with the INTRA-coded P-blocks but are used for predicting B-blocks.

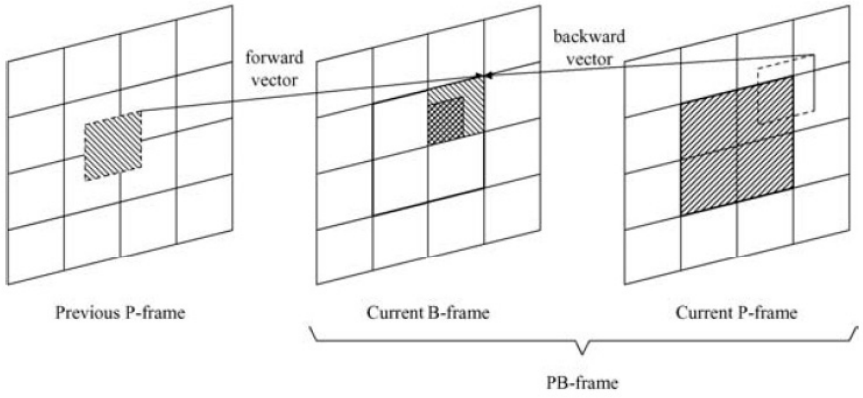


Figure 3.4: Prediction in PB-frames mode

For prediction of a B-block, both forward, $\mathbf{MV}_F$, and backward, $\mathbf{MV}_B$, motion vectors are needed. Those are not transmitted but are derived at the decoder by scaling the corresponding P-block motion vector, **MV**, using the temporal resolutions of the P- and B-pictures with respect to the previous P-picture. The derived motion vectors can be optionally enhanced using a transmitted delta vector **MVDB**. The forward (or backward) motion vectors for chroma blocks are derived by summing the corresponding luma forward (or backward) motion vectors, dividing by 8, and then rounding to the nearest half-pel resolution. To be able to predict a B-macroblock, the corresponding P-macroblock is first reconstructed. For pels of a B-block where $\mathbf{MV}_B$ points outside the reconstructed P-macroblock, forward prediction using $\mathbf{MV}_F$ and the previous decoded P-picture is used. However, for pels of the same B-block where $\mathbf{MV}_B$ points inside the reconstructed P-macroblock, bidirectional prediction is used. In this case, the prediction is the average (with truncation) of the forward prediction, using $\mathbf{MV}_F$ and the previous decoded P-picture, and the backward prediction, using $\mathbf{MV}_B$ and the reconstructed P-macroblock. This process is illustrated in Figure 3.4. With this mode, the frame rate can be increased without a significant increase in bit rate.

3.4.6 H.263, Version 2 (H.263+)

Version 2 of the H.263 standard is informally known as H.263+. This version adds a number of optional feature enhancements to version 1. In the process of adding these new features, the precise definition and requirements of the original version 1 syntax and semantics were not changed. In fact, version 2 is backward compatible with version 1. The additional optional feature set can be summarized in terms of the new types of pictures, a modified unrestricted

motion vector mode, and 12 new optional modes (annexes I–T). This is briefly described in what follows.

3.4.6.1 New types of pictures

Version 2 defines three new types of pictures:

1. *Scalability pictures*: Three types of scalability pictures were added, one that provides temporal scalability and two that provide SNR or spatial scalability:
   (a) B: a picture having two reference pictures, one of which temporally precedes the B picture and one of which is temporally subsequent to the B picture.
   (b) EI: a picture having a temporally simultaneous reference picture.
   (c) EP: a picture having two reference pictures, one of which temporally precedes the EP picture and one of which is temporally simultaneous.

   These pictures are described in more detail in Section 3.4.6.9 in the discussion of the new optional scalability mode (annex O).
2. *Improved PB-frames*: Recent investigations have indicated that the current PB-frames utilized by version 1 are not sufficiently robust for continual use. Encoders implementing the PB-frames mode are limited to use only bidirectional prediction. In some situations, this results in a lack of usefulness of the PB-frames mode. An improved, more robust type of PB-frames has been added to enable heavier, higher-performance use of the PB-frames mode. This is described in more detail in Section 3.4.6.7 in the discussion of the new optional improved PB-frames mode (annex M).
3. *Custom source formats*: As already discussed, version 1 allows only five video source formats (CIF family) with defined picture size, picture shape, and picture clock frequency. Version 2, however, allows a wide range of optional custom source formats in order to make the standard apply to a much wider class of video scenes and applications, such as resizable computer window-based displays, high refresh rates, and wide-format viewing screens.

3.4.6.2 Modified Unrestricted Motion Vector Mode (modified Annex D)

The optional unrestricted motion vector mode (annex D) of version 1 has been modified in version 2. Version 2 defines a new data field called PLUSPTYPE. When using the unrestricted motion vector mode, if PLUSPTYPE is present

in the picture header, then the following modifications apply:

1. The motion vector range no longer depends on the motion vector prediction value. There are two cases here:
   (a) If the UUI data field in the picture header is set to "1," the motion vector range depends on the picture format. For standardized picture formats up to CIF the range is $[-32, 31.5]$, for those up to 4CIF the range is $[-64, 63.5]$, for those up to 16CIF the range is $[-128, 127.5]$, and for even larger custom picture formats the range is $[-256, 255.5]$. In addition, the horizontal and vertical motion vector ranges may be different for custom picture formats.
   (b) If, however, the UUI data field is set to "01," the motion vectors are not limited except by their distance to the coded area border, as explained by the following restriction rule: the motion vector values are restricted such that no element of the 16×16 (or 8×8) region that is selected shall have a horizontal or vertical distance more than 15 pels outside the coded picture area. Q1
2. A new VLC table is employed to encode the motion vector differences. This table has the following properties:
   (a) The codes are single-valued. In other words, each codeword corresponds to a single motion vector difference value. This is in contrast to the double-valued VLC codes of version 1, where each codeword can represent one of two possible motion vector differences. Double-valued codes were not popular due to their high implementation cost and the limitations on their extendibility.
   (b) The table employs *reversible variable-length coding* (RVLC) codewords. Such codewords can be decoded in both the forward and backward directions. As discussed in Chapter 9, the use of RVLCs can increase the error resilience of video bitstreams. In addition, RVLCs are easier to implement because they can easily be generated and decoded using a simple state machine.

3.4.6.3 Advanced INTRA Coding Mode (Annex I)

This optional mode significantly improves the compression performance when coding INTRA macroblocks. The mode is applied both to INTRA macroblocks within INTRA-pictures and to INTRA macroblocks within INTER-pictures. The improved compression performance of this mode is achieved as follows:

1. INTRA blocks are predicted from their neighboring INTRA blocks. Block prediction always uses data from the same luma or chroma

component. There are three options for prediction:

(a) DC only: where the DC coefficient is predicted as the average of the corresponding coefficients from the block above and the block to the left.

(b) Vertical DC and AC: where the DC coefficient and the first row of AC coefficients are vertically predicted from the corresponding coefficients from the block above.

(c) Horizontal DC and AC: where the DC coefficient and the first column of AC coefficients are horizontally predicted from the corresponding coefficients from the block to the left.

Special cases are defined for situations in which the neighboring blocks are not INTRA coded or are not in the same video picture segment. The option that gives the best prediction for the whole macroblock is chosen.

2. The quantization of INTRA coefficients is modified. INTRA DC coefficients are quantized using a varying quantization step size, unlike the fixed quantization step size of 8 utilized when this mode is not in use. In addition, the quantization of all INTRA coefficients is performed without a dead-zone.

3. The scanning of DCT coefficients is adapted to the prediction method of the INTRA macroblock. For macroblocks predicted using the DC-only option, the normal zigzag scanning is utilized; for macroblocks predicted using the vertical DC and AC option, a new alternate horizontal scanning pattern (Figure 3.5(a)) is utilized; whereas for macroblocks predicted

0	1	2	3	10	11	12	13
4	5	8	9	17	16	15	14
6	7	19	18	26	27	28	29
20	21	24	25	30	31	32	33
22	23	34	35	42	43	44	45
36	37	40	41	46	47	48	49
38	39	50	51	56	57	58	59
52	53	54	55	60	61	62	63

(a) Alternate horizontal scan

0	4	6	20	22	36	38	52
1	5	7	21	23	37	39	53
2	8	19	24	34	40	50	54
3	9	18	25	35	41	51	55
10	17	26	30	42	46	56	60
11	16	27	31	43	47	57	61
12	15	28	32	44	48	58	62
13	14	29	33	45	49	59	63

(b) Alternate vertical scan

Figure 3.5: Alternate scans for the advanced INTRA mode of H.263+

using the horizontal DC and AC option, a new alternate vertical scanning pattern (Figure 3.5(b)) is utilized.

4. The quantized INTRA coefficients are encoded using a new VLC table optimized for the global statistics of INTRA macroblocks.

3.4.6.4 Deblocking Filter Mode (Annex J)

In this optional mode, a filter is applied, both at the encoder and at the decoder, across the boundaries of luma and chroma 8×8 blocks of reconstructed pictures before storing them in the picture memory. In other words, the filter affects the picture that is used for the prediction of subsequent pictures and thus lies within the motion prediction loop.

The deblocking filter operates using a set of four pel values either on a horizontal or on a vertical line of the reconstructed picture. Two of the four pels belong to one block, whereas the other two belong to a neighboring block. The weights of the filter's coefficients depend on the quantizer step size, where stronger coefficients are used for a coarser quantizer, and vice versa. No filtering is performed across a picture edge. Similarly, when the *Independent Segment Decoding* (ISD) mode is in use, no filtering is performed across slice edges (when the Slice Structured mode is in use) or across the top boundary of GOBs having GOB headers present (when the Slice Structured mode is not in use). When this mode is used together with the Improved PB-frames mode, the backward prediction of the B-macroblock is based on the reconstructed P-macroblock before the deblocking edge filter operations. The mode applies only for the P-, I-, EP-, or EI-pictures or the P-picture part of an Improved PB-frame. Possible filtering of B-pictures or the B-picture part of an Improved PB-frame is not a matter for standardization.

In addition to the filtering operation, this mode allows the use of four motion vectors per macroblock and also the use of unrestricted motion vectors. This mode improves the prediction quality and significantly reduces blocking artefacts.

3.4.6.5 Slice Structured Mode (Annex K)

In this optional mode, a *slice* layer is employed instead of the normal GOB layer. This mode is used to provide enhanced error resilience, to make the bitstream more amenable to use with packet-based networks, and to minimize video delay. A slice layer allows a flexible partitioning of the picture into segments containing a variable number of macroblocks. It also allows more control over the shape of segments. In addition, a slice structure provides more flexibility in the transmission order. This is in contrast with a GOB layer, which only allows partitioning into fixed-size, fixed-shape segments with fixed transmission order.

In order to facilitate optimal usage in a number of environments, this mode contains two submodes:

1. *The Rectangular Slice (RS) submode*: When RS is in use, the slice occupies a rectangular region of width specified in units of macroblocks and contains a number of macroblocks in scanning order within the rectangular region. When RS is not in use, the slice contains a number of macroblocks in scanning order within the picture as a whole.

2. *The Arbitrary Slice Ordering (ASO) submode*: when ASO is in use, the slices may appear in any order within the bitstream. When ASO is not in use, the slices must be sent in scanning order.

A slice video picture segment starts at a macroblock boundary in the picture and contains a number of macroblocks. Different slices within the same picture shall not overlap with each other, and every macroblock shall belong to one and only one slice.

A *slice* is defined as a slice header followed by consecutive macroblocks in scanning order. In order to allow slice header locations within the bitstream to act as resynchronization points for bit error and packet loss recovery and in order to allow out-of-order slice decoding within a picture, slice boundaries are treated differently than simple macroblock boundaries. Thus, no data dependencies can cross the slice boundaries within the current picture. An exception to this is the Deblocking Filter mode, which, when in use without the Independent Segment Decoding mode, filters across the boundaries of the blocks in the picture.

3.4.6.6 Supplemental Enhancement Information Mode (Annex L)

In this mode, additional supplemental information can be included in the bitstream to signal an enhanced display capability or to provide information for external usage. The supplemental information may be present in the bitstream even though the decoder may not be capable of providing the enhanced capability to use it or even to properly interpret it. In this case, the decoder can simply discard the supplemental information. The mode can be used to signal the following capabilities:

1. *Picture Freeze*: The mode can be used to signal that the contents of the entire prior-displayed picture, or a specified rectangular part of it, shall be kept unchanged. The mode can also be used to explicitly signal a picture freeze release.

2. *Picture Freeze with Resizing*: The mode can be used to signal that the contents of a specified rectangular area of the prior-displayed picture

should be resized to fit into a smaller part of the displayed video picture, which should then be kept unchanged.

3. *Picture Snapshot*: The mode can be used to signal that the current picture, or a specified rectangular part of it, is labeled for external use as a still-image snapshot of the video content.

4. *Video Time Segment*: The mode can be used to signal the beginning and the end of a specified subsequence of video data to be used externally.

5. *Progressive Refinement Segment*: The mode can be used to signal the beginning and the end of a specified subsequence of video data. Rather than being a continually moving scene, this subsequence of video includes a start picture followed by a sequence of zero or more pictures to refine its quality.

6. *Chroma-Keying Information*: The mode can be used to indicate that the *chroma-keying* technique is used to represent *transparent* and *semi-transparent* pels in the decoded video pictures. When being presented on the display, transparent pels are not displayed. Instead, a background picture is revealed that is either a prior reference picture or an externally controlled picture. Semitransparent pels are displayed by blending the pel value in the current picture with the corresponding value in the background picture.

3.4.6.7 Improved PB-Frames Mode (Annex M)

This mode represents an improvement compared to the original PB-frames optional mode (annex G). The main difference between the two modes is that the original PB-frames mode can utilize only bidirectional prediction to predict the B part in a PB-frame, whereas the improved PB-frames mode can utilize forward, backward, or bidirectional prediction.

The bidirectional prediction method is the same as in the original PB-frames mode, except that in this case no delta vector is transmitted. In the forward-prediction method, a B macroblock is predicted from the previously decoded P-picture and a forward motion vector is transmitted. In the backward-prediction method, a B-macroblock is predicted from the corresponding P-macroblock currently decoded in the same PB-frame, and therefore no backward motion vector needs to be transmitted.

This mode significantly improves coding efficiency in situations in which downscaled P-vectors (utilized in the original PB-frames mode) are not good candidates for B-prediction. In particular, the backward prediction is useful when there is a scene cut between the previous P-frame and the current PB-

frame. In general, it is advisable to use the Improved PB-frames mode instead of the original PB-frames mode.

3.4.6.8 Reference Picture Selection Mode (Annex N)

In normal operation, a picture is temporally predicted from the most recently decoded picture. The reference picture section (RPS) mode, however, allows temporal prediction from pictures other than the most recently decoded one. Thus, in this mode, both the encoder and the decoder use more than one picture memory. As discussed in Chapter 6, this method belongs to a class of motion estimation and compensation techniques called *multiple-reference motion-compensated prediction*. The information to signal which picture is selected for prediction is included by the encoder in the encoded bitstream. However, the strategy used by the encoder to select this picture is not subject for standardization.

This mode can be used to improve the performance of video communication over error-prone channels. In normal operation, if part of the reference picture is lost due, for example, to a transmission error, then this error will propagate to and severely degrade the quality of future pictures. In this mode, however, the encoder may switch to another reference picture to suppress the temporal error propagation due to interframe coding.

In order to utilize this mode, the encoder needs to have some knowledge about the conditions of the channel and the outcome of the decoding process (e.g., which parts of the reference picture have been decoded in error). One way to achieve this is to utilize a backward (feedback) channel. This mode has two back-channel mode switches that define whether a backward channel is used and what kind of messages are returned on that backward channel from the decoder. Together, the two switches define four basic methods of operation: NEITHER (no backward messages), ACK (acknowledgment messages only), NACK (negative acknowledgment messages only), and ACK+NACK (both acknowledgment and negative acknowledgment messages). There are also two methods of operation in terms of the channel for backward channel messages. The first method is the Separate Logical Channel mode, where back-channel data is delivered through a separate logical channel in the multiplex layer of the system, whereas the second method is the VideoMux mode, where back-channel data for received video is delivered within the forward video data of a video stream of encoded data.

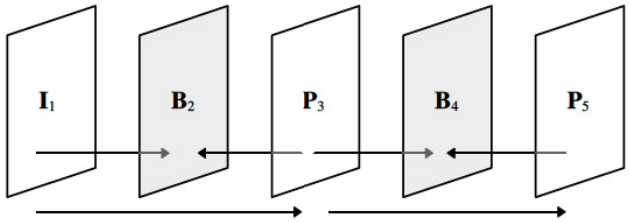
3.4.6.9 Temporal, SNR, and Spatial Scalability Mode (Annex O)

Scalability implies that a bitstream is composed of a *base layer* and one or more associated *enhancement layers*. The base layer is separately decodable. The enhancement layers can be decoded in conjunction with the base

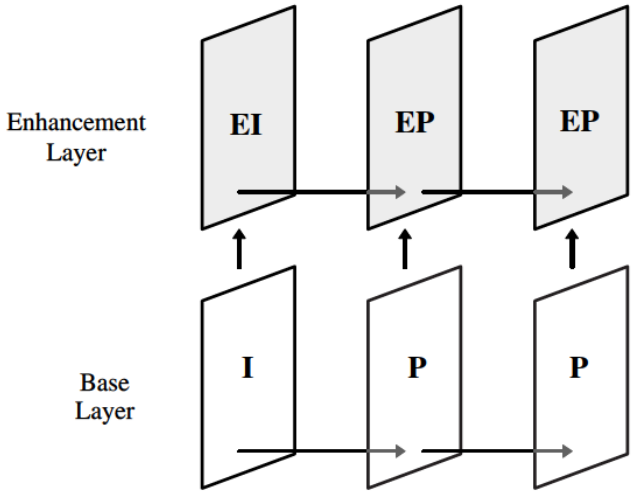
layer to increase perceived quality by either increasing the picture rate (temporal scalability), increasing the picture SNR quality (SNR scalability), or increasing the picture resolution (spatial scalability). This mode has support for three types of scalability: temporal, SNR, and spatial scalability, as detailed next. This mode can be helpful when used over heterogenous networks with varying bandwidth capacity and also in conjunction with error correction schemes.

a. **Temporal scalability:** *Temporal scalability* refers to enhancement information used to increase the picture quality by increasing the picture display rate. Temporal scalability is achieved by employing bidirectionally predicted pictures, or B-pictures. B-pictures can be predicted from a previous and/or a subsequent reconstructed picture in the reference layer (the layer used for prediction). B-pictures in this mode differ from the B-picture part of a PB- (or an Improved PB-) frame in that they are separate entities in the bitstream. In other words, they are not syntactically intermixed with a subsequent P-picture. It should be emphasised that B-pictures should not be used as reference pictures for the prediction of any other picture. This is particularly important to allow for B-pictures to be discarded if necessary without adversely affecting any subsequent pictures, thus providing temporal scalability. Figure 3.6(a) illustrates temporal scalability using B-pictures. It should be pointed out that the location of B-pictures in the bitstream is in a data-dependence order rather than in a temporal order. For example, in the case shown in Figure 3.6(a) the bitstream order of the encoded pictures is I_1, P_3, B_2, P_5, $B_4, \ldots$. There is no limit to the number of B-pictures that may be inserted between pairs of reference pictures in the reference layer. In this mode, motion vectors are allowed to extend beyond the picture boundaries of B-pictures.

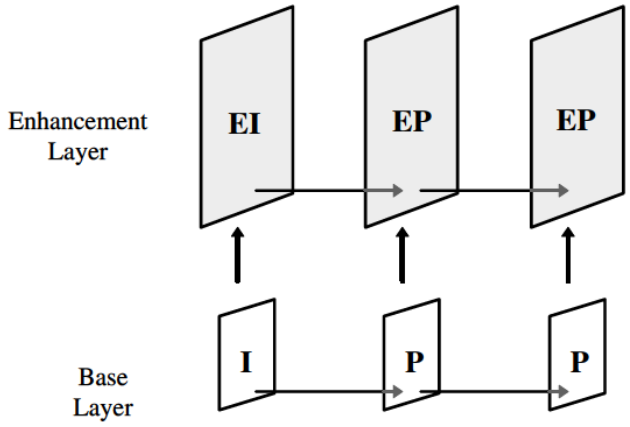
b. **SNR scalability:** *SNR scalability* refers to enhancement information used to increase the picture quality without increasing picture resolution. The process of compression usually introduces artefacts and distortions. As a result, the difference between a reconstructed picture and its original in the encoder is almost always a nonzero-valued picture. Normally, this coding error picture is lost at the encoder and never recovered. With SNR scalability, however, these coding-error pictures can be encoded and sent to the decoder. At the decoder, such coding-error pictures can be used to increase the signal-to-noise ratio of the decoded picture, and hence the term SNR scalability. Figure 3.6(b) illustrates SNR scalability. If the enhancement-layer picture is predicted only from a simultaneous lower-layer reference picture, then the enhancement-layer picture is referred to as an EI-picture. If, however, the enhancement-layer picture is bidirectionally predicted using both a prior enhancement-layer picture and a temporally simultaneous lower-layer reference picture, then the enhancement-layer picture is referred to as an EP-picture. The picture in the reference



(a) Temporal scalability



(b) SNR scalability



(c) spatial scalability

Figure 3.6: Temporal, SNR, and spatial scalability in H.263+

layer that is used for upward prediction of an EI- or EP-picture may be an I-picture, a P-picture, or the P part of a PB- or Improved PB-frame. Thus, an EI-picture in an enhancement layer may have a P-picture as its lower-layer reference picture, and an EP-picture may have an I-picture as its lower-layer enhancement picture. For both EI- and EP-pictures, the prediction from the lower reference layer uses no motion vectors. However, EP-pictures use motion vectors for the prediction from their prior reference picture in the same layer.

c. **Spatial scalability:** *Spatial scalability* refers to enhancement information used to increase the picture quality by increasing picture resolution either horizontally, vertically, or both. Spatial scalability is very similar to SNR scalability. The only difference is that before the picture in the reference layer is used to predict the picture in the enhancement layer, it is interpolated by a factor of 2 either horizontally or vertically (1-D spatial scalability) or both horizontally and vertically (2-D spatial scalability). The interpolation filters for this operation are defined by the standard. Spatial scalability is illustrated in Figure 3.6(c).

d. **Multilayer scalability:** It is possible not only for B-pictures to be temporally inserted between pictures of types I, P, PB, and Improved PB, but also between pictures of types EI and EP (whether these consist of SNR or spatial-enhancement pictures). It is also possible to have more than one SNR or spatial-enhancement layers in conjunction with a base layer. Thus a multilayer scalable bitstream can be a combination of SNR layers, spatial layers, and B-pictures.

3.4.6.10 Reference Picture Resampling Mode (Annex P)

In this mode, a resampling operation can be applied to the previously decoded picture in order to generate a new *warped* picture for use as reference for predicting the currently encoded picture. For example, if the previous reference picture and the current picture are of different source formats, then this mode can be used to resample the previous picture to match the source format of the current picture. Another example is to use this mode to warp the previous reference picture to compensate for global motion. Warping and warping-based motion estimation methods are discussed in Chapter 5.

3.4.6.11 Reduced-Resolution Update Mode (Annex Q)

This mode allows the encoder to send information encoded at a low resolution to update a higher-resolution reference picture and produce a final picture at the higher resolution. This mode is particularly useful when encoding a highly active scene, and allows an encoder to increase the picture rate at

which moving parts of a scene can be represented while maintaining a higher-resolution representation in more static areas of the scene.

The syntax of the bitstream in this mode is identical to the syntax for coding without the mode, but the semantics, or interpretation of the bitstream, is somewhat different. In this mode, the portion of the picture covered by a macroblock is twice as wide and twice as high. Thus, there is approximately one-quarter the number of macroblocks as there would be without this mode. Motion vector data also refers to blocks of twice the normal height and width, or 32×32 and 16×16 instead of the normal 16×16 and 8×8. For example, the decoder receives and decodes a 16×16 DFD block at the reduced resolution. The decoder then upsamples this block to 32×32 at the higher resolution. The decoder then upsamples the received motion vector by a factor of 2 and uses it to produce a 32×32 prediction from the reference picture. The DFD block and the prediction block are then added to produce a 32×32 block at the higher resolution.

3.4.6.12 Independent Segment Decoding Mode (Annex R)

This mode allows a picture to be constructed without any data dependencies that cross video picture segments. Thus, this mode provides error robustness by preventing the propagation of erroneous data across the boundaries of video picture segments.

In this mode, a video picture segment can be a slice, a GOB or multi-GOBs with nonempty GOB headers, or a complete picture. When this mode is in use, the video picture segment boundaries are treated as picture boundaries. In other words, each video picture segment is decoded with complete independence from all other video picture segments, and is also independent of all data outside the corresponding video picture segment in the reference picture(s).

For example, motion vectors of blocks outside the current video picture segment cannot be used when calculating the current motion vector predictor. Similarly, motion vectors of blocks outside the current video picture segment cannot be used as remote motion vectors for overlapped block-motion compensation when the Advanced Prediction mode is in use. In addition, no motion vectors are allowed to reference areas outside the corresponding video picture segment in the reference picture(s).

3.4.6.13 Alternative INTER VLC Mode (Annex S)

This mode improves the efficiency of encoding some INTER macroblocks by allowing a VLC table originally designed for INTRA macroblocks to be used for some INTER macroblocks. The INTRA VLC table used in the advanced INTRA coding mode (annex I) is designed to efficiently encode INTRA blocks. Thus, it is optimized for coding blocks with many large-valued coeffi-

cients and small runs of zeros. There are cases, however, where the statistics of INTER blocks can approximate the statistics of INTRA blocks. This is particularly possible when significant changes are evident in the picture or when small quantizer step sizes are employed. In such cases, it can become more efficient to encode INTER blocks using the INTRA VLC table.

In this mode, the encoder would normally choose to use the INTRA VLC table for coding an INTER block only when the use of this table results in fewer bits than the use of the INTER VLC table. This use of the alternative INTRA VLC table, however, is subject to the condition that the decoder would be able to detect which of the two tables was used for encoding. Thus, the alternative INTRA VLC table can be used, subject to the condition that decoding using the INTER VLC table would result in runs of zeros so long as to indicate the presence of more than 64 coefficients in the block.

3.4.6.14 Modified Quantization Mode (Annex T)

In this mode, the quantizer operation is modified. In particular, this mode includes the following four key features:

1. In normal mode, the change of the quantization parameter at the macroblock level is limited to a maximum of ± 2. This mode, however, improves the bit-rate control ability by allowing the quantization parameter to be changed at the macroblock level to any of its 31 permissible values.

2. In normal mode, the chroma quantizer step size is the same as that for luma. This mode, however, improves the fidelity of chroma by specifying a smaller quantizer step size for chroma than that for luma.

3. The true value of a DCT coefficient prior to quantization can be as high as 2040. Thus, when the quantization parameter is less than 8, the quantized DCT coefficients can be outside the range $[-127, +127]$ permissible in the normal mode. Such coefficients are clipped to the permissible range before being encoded. This mode, however, extends the range of representable quantized DCT coefficient values to allow the representation of any possible true coefficient value to within the accuracy allowed by the quantizer step size.

4. In this mode certain restrictions are placed on the encoded DCT coefficient values to improve the detectability of errors and to minimize decoding complexity.

Kossentini *et al.* [71, 72] provide an excellent overview of H.263+ and evaluate the performance of the modes individually and in different combinations.

3.4.7 H.263, Version 3 (H.263++)

Version 3 of the H.263 standard is informally known as H.263++. This version adds a number of optional feature enhancements to versions 1 and 2.

3.4.7.1 Enhanced Reference Picture Selection Mode (Annex U)

The *enhanced reference picture selection* (ERPS) mode is an enhancement to the RPS mode (annex N) of H.263+. In addition to enhancing error resilience, this mode provides benefits in terms of coding efficiency.

As with the RPS mode, the ERPS mode extends the motion estimation and compensation processes to use more than one reference picture. In the ERPS mode, however, enhanced performance is achieved by allowing reference picture selection on the macroblock, rather than the picture, level. Thus, in this case, each motion vector is extended by a picture reference parameter that is used to address a macroblock or block prediction region in any of the multiple reference pictures.

The ERPS mode also includes a submode for improving the coding efficiency of B-pictures. In this submode, encoders can use more than one reference picture for both forward and backward prediction of B-pictures.

Another submode of ERPS is provided to reduce memory requirements. In this submode, each reference picture is partitioned into smaller rectangular units called *subpictures*. The encoder can then indicate to the decoder that specific subpicture areas of specific reference pictures will not be used as a reference for the prediction of subsequent pictures. This allows the memory allocated in the decoder for storing these areas to be used to store data from other reference pictures.

3.4.7.2 Data Partitioned Slice Mode (Annex V)

In this mode, data is arranged in a video picture segment as defined in the independent segment decoding mode (annex R) of H.263+. The contents of this segment are rearranged such that the header information for *all* the MBs in the segment are encoded and transmitted *together*, followed by the motion vectors for all the MBs in the segment and then by the DCT coefficients for all the MBs in the segment. The segment header uses the same syntax as the slice structured mode (annex K) of H.263+. The header, motion vectors, and DCT partitions are separated by markers. In addition to data partitioning, this mode uses RVLC tables for encoding header and motion information. As will be discussed later, data partitioning and RVLC provide robustness in error-prone environments. Another error-resilience enhancement in this mode is that the motion vector predictor is no longer formed from three neighboring motion

vectors. Instead, a new prediction method is used to allow independent motion vector decoding in both the forward and backward directions.

3.4.7.3 Additional Supplemental Enhancement Information (Annex W)

This annex describes additional supplemental enhancement information that adds to the functionality of the supplemental enhancement information mode (annex L) of H.263+. In particular, the following additional information can be added to the bitstream:

1. Indication of the use of a specific fixed-point IDCT.
2. Picture messages, including the message types of:
   (a) Arbitrary binary data.
   (b) Text (arbitrary, copyright, caption, video description, or uniform resource identifier).
   (c) Picture header repetition (current, previous, next with reliable temporal reference, or next with unreliable temporal reference).
   (d) Interlaced field indications (top or bottom).
   (e) Picture number.
   (f) Spare reference picture identification.

3.4.7.4 Profiles and Levels Definitions (Annex X)

With the variety of optional modes available in H.263, it is crucial that several preferred mode combinations for operation be defined so that different terminals will have a high probability of connecting to each other. This annex contains a list of preferred mode combinations, which are structured into "profiles" of support. It also defines some groupings of maximum performance parameters as "levels" of support for these profiles.

The annex defines nine profiles (profile 0 to profile 8). Each profile is defined in terms of a set of features supported by the decoder. For example, the *Baseline Profile* (profile 0) refers to the syntax of H.263 with no optional modes of operation. Another example is *Version 2 Interactive and Streaming Wireless Profile* (profile 3). This profile is defined to provide enhanced coding efficiency performance and enhanced error resilience for delivery to wireless devices within the feature set available in H.263+. This profile of support is composed of the baseline design plus the following modes: advanced INTRA coding mode (Annex I), deblocking filter mode (Annex J), slice structured mode (Annex K), and the modified quantization mode (Annex T).

The annex also defines seven levels (level 10 to level 70) of performance capability for decoder implementation. For example, a decoder supporting the first level, level 10, must include support of QCIF and sub-QCIF resolution decoding, and must be capable of operation with a bit rate up to 64,000 bits per second with a picture decoding rate up to (15,000)/1001 pictures per second.

3.5 The MPEG-4 Standard

As already discussed, the formal title "Generic coding of audiovisual objects" given to MPEG-4 describes two important properties of the standard. The first property is that it is a generic standard. It defines tools and algorithms for the coding of natural, synthesis, and hybrid audiovisual objects with a wide range of bit rates, picture formats, transmission media, etc. It is, therefore, very difficult to describe the full functionality of such a generic standard in a volume of this size.[5] Thus, this section will concentrate on MPEG-4 *natural video coding*. In particular, the section will try to highlight the second property of MPEG-4, i.e., being object-based, which sets it apart from other standards.

3.5.1 An Object-Based Representation

MPEG-4 uses an object-based representation model. Thus, a scene is represented, coded, and manipulated as individual *audiovisual objects* (AVOs). This section concentrates on *natural video objects*.

As illustrated in Figure 3.7, an MPEG-4 *video session* (VS) is a collection of one or more *video objects* (VOs). A VO is an entity that a user is allowed to access (e.g., seek and browse) and manipulate (e.g., cut and paste). It can be a simple rectangular frame or it can be an arbitrarily shaped object. A VO can consist of one or more *video object layers* (VOLs). As is discussed later, each VO can be encoded in either a scalable (multiple VOLs) or a nonscalable (single VOL) form. Each VOL consists of an ordered sequence of *video object planes* (VOPs). A VOP is an instance (or a snapshot) of the corresponding VO at a given time. A number of VOPs can, optionally, be grouped together in a *group of video object planes* (GOV). GOVs can provide points in the bitstream where VOPs are encoded independently from each other. This provides random access points within the bitstream.

Figure 3.8 shows a general block diagram of an MPEG-4 codec. The input video is represented using a number of VOs. This object-based representation either already exists (e.g., generated with chroma-key technology) or is

[5]To give an indication of how generic the MPEG-4 standard is, the MPEG-4 draft [67] that was used in writing the current section is more than 300 pages.

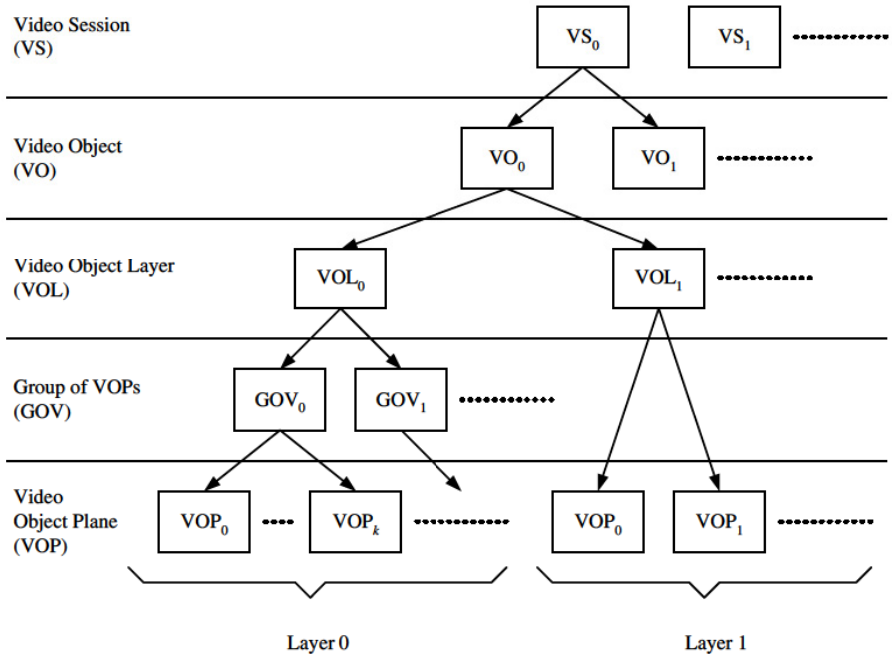


Figure 3.7: MPEG-4 video bitstream structure

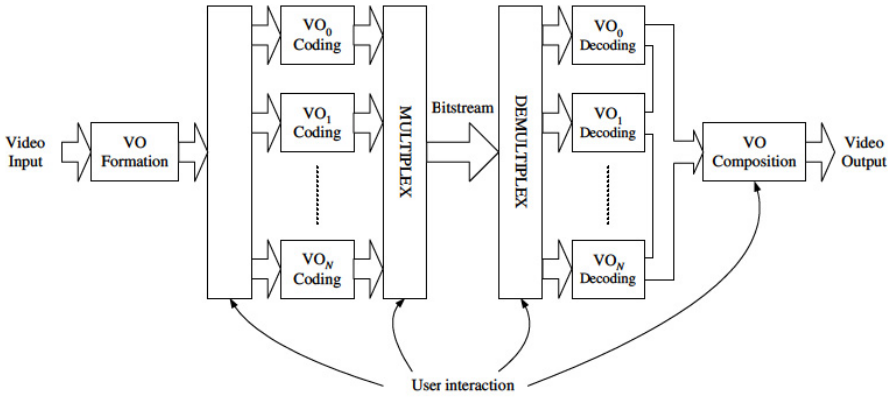


Figure 3.8: An MPEG-4 codec

generated using segmentation techniques. Each VO is encoded individually, and the resulting bitstreams are multiplexed to a single bitstream. At the decoder, the received bitstream is first demultiplexed to the individual bitstreams. Each bitstream is then decoded, and the decoded VOs are composited to

reconstruct the output video. As shown, at various points of this encoding-decoding process, users are allowed to interact with (access and/or manipulate) the individual VOs.

As an example, consider a sequence showing a hot-air balloon flying in the sky. In this case, the sequence can be represented using two VOs: the balloon and the sky background. Figure 3.9(a) shows a single frame of this sequence. At this particular instance of time the two VOs are represented by the two VOPs shown in Figures 3.9(b) and 3.9(c). At the encoder, each VOP is encoded individually and the two bitstreams are multiplexed. At the decoder, the received bitstream is demultiplexed to the two individual bitstreams. Each bitstream is then decoded to reconstruct the corresponding VOP. The two VOPs are then put together to reconstruct the transmitted frame. The user can optionally manipulate the decoded VOPs. For example, in Figure 3.9(d) the balloon VOP has been enlarged, rotated, and translated as compared to the original frame.

In addition to composition information (which indicates where and when the VOP is to be displayed), each VOP is encoded in terms of its *shape*,



(a) Balloon in Sky (original)



(b) Sky (background) VOP



(c) Balloon VOP



(d) Decoded and manipulated

Figure 3.9: Object-based representation, coding, and interaction

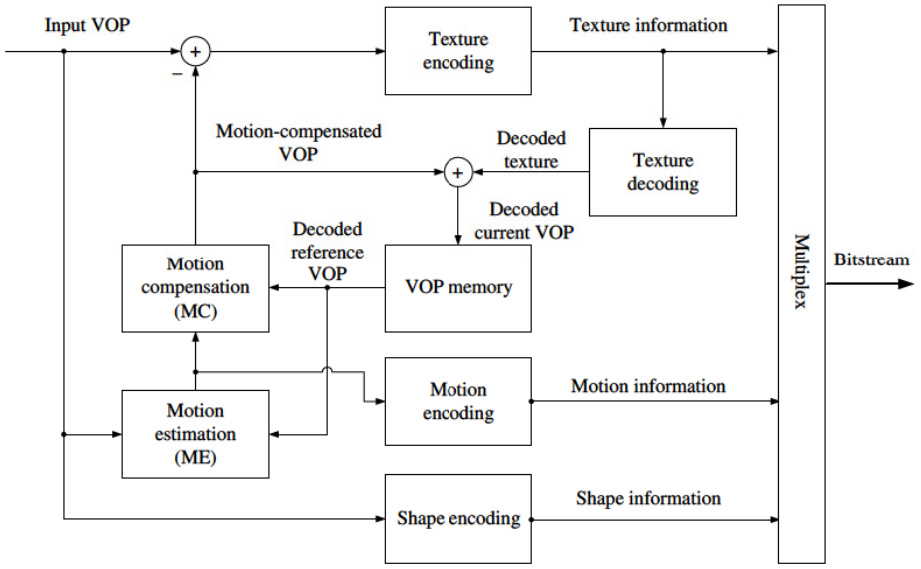


Figure 3.10: An MPEG-4 VOP encoder

motion, and *texture*. This is illustrated in Figure 3.10. As can be seen, an MPEG-4 VOP encoder has three main functionalities: *shape encoding*, *motion encoding* (along with motion estimation and compensation), and *texture encoding*. Note that the structure of this encoder is very similar to the MC-DPCM structure utilized by H.263 and most other standards. In fact, for most cases, the texture encoder is DCT-based and the structure is very similar to the conventional hybrid MC-DPCM/DCT encoder. The difference here is that the encoded entities can have arbitrary shapes rather than the fixed rectangular frame shape, and therefore additional shape information needs to be encoded and transmitted. Note that this object-based representation can be thought of as a generic representation. When a frame is encoded using a single VOP, this generic representation degenerates into the special case of rectangular frames and an MPEG-4 encoder becomes almost identical to an H.263 encoder. In fact, the MPEG-4 standard provides measures to ensure some level of interoperability with MPEG-1/2 and H.263.

A VOP is encoded on a macroblock (MB) basis. MPEG-4 supports a 4:2:0 subsampling format with 4–12 bits/sample. Thus, an MB consists of six 8×8 blocks: four luma blocks and two corresponding chroma blocks. To achieve efficient encoding, the arbitrary shaped VOP is first encapsulated within a bounding box. This bounding box is chosen such that it completely contains the VOP but uses the minimum number of macroblocks. This bounding box is illustrated for the Balloon VOP in Figure 3.11. Within this bounding box,

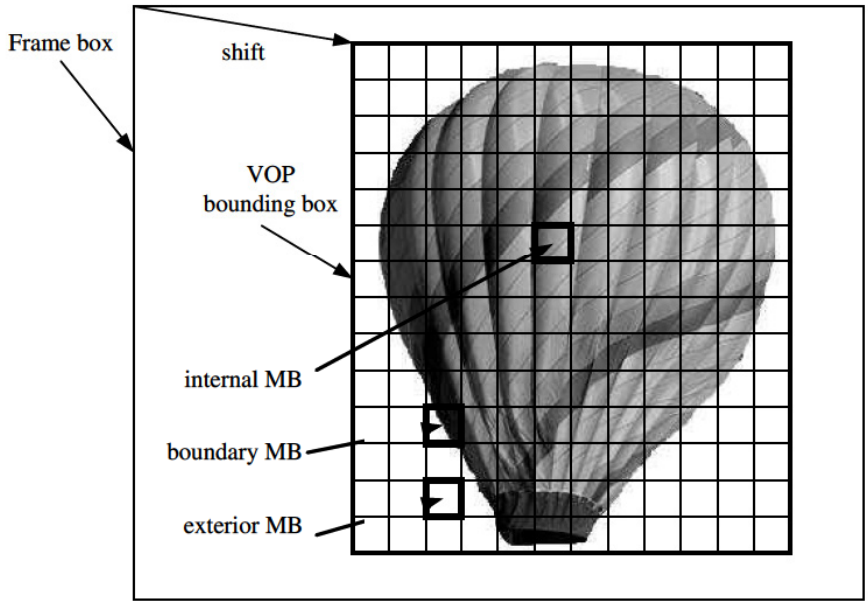


Figure 3.11: The bounding box of the Balloon VOP

there are three types of MBs: internal MBs, boundary MBs, and exterior MBs. An *internal* MB lies completely inside the VOP, whereas a *boundary* MB lies on the contour of the VOP; i.e., parts of it are inside the VOP and the other parts are outside the VOP. An *exterior* MB, on the other hand, lies completely outside the VOP. Note that the shape, size, and location of this bounding box can change from one time instance to another. Thus, the absolute (frame) coordinate system is used to define such bounding boxes.

The following subsections briefly describe the main building blocks of the MPEG-4 VOP encoder.

3.5.2 Shape Coding

In the context of MPEG-4, shape information is referred to as *alpha planes*. There are two types of alpha planes: binary and gray-scale. A *binary alpha plane* defines which pels within the bounding box belong to the video object at a given instant of time. A *gray-scale alpha plane*, on the other hand, is a more general form of alpha planes, for it includes transparency information.

3.5.2.1 Binary Shape Coding

A binary alpha plane is represented by a matrix the same size as the bounding box of the video object. Every element within this matrix can take one of two

possible values. If the corresponding pel belongs to the object, then the element is set to 255; otherwise it is set to 0. This matrix is sometimes referred to as a *binary mask* or as a *bitmap*. Figure 3.12 shows the binary alpha plane of the Balloon VOP.

Before encoding, the binary alpha plane is partitioned into 16×16 blocks called *binary alpha blocks* (BABs). A BAB with all elements equal to 0 is called a *transparent* BAB, whereas a BAB with all elements equal to 255 is called an *opaque* BAB. Each BAB is encoded separately. The main tools used for encoding BABs are *context-based arithmetic encoding* (CAE) and motion compensation. There are two variants of the CAE algorithm. One is used with motion compensation and is called InterCAE, whereas the other one is used without motion compensation and is called IntraCAE. There are seven possible modes for encoding a BAB:

1. The BAB is flagged transparent. In this case, no shape coding is necessary. In addition, texture information is not coded for this BAB.

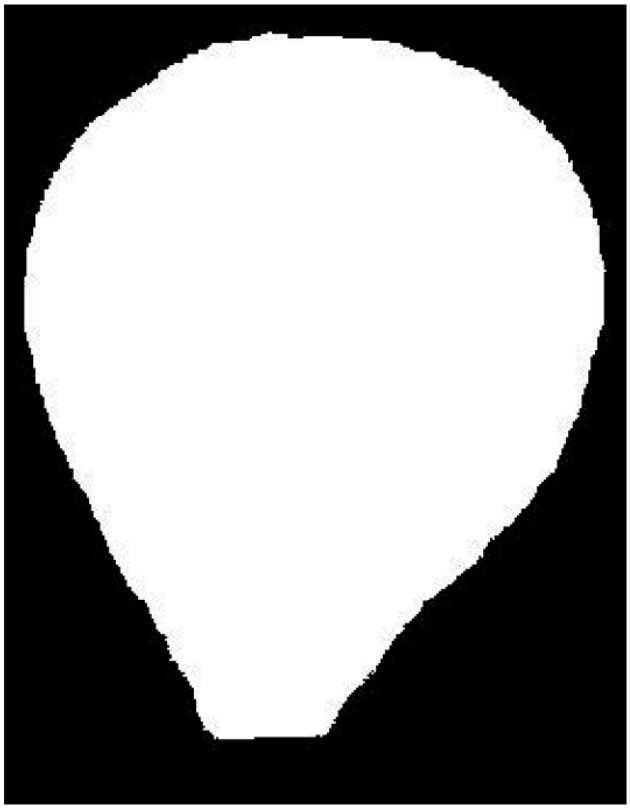


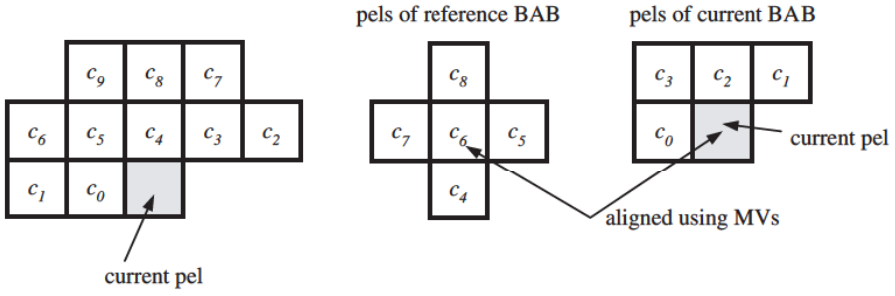
Figure 3.12: The binary alpha plane of the Balloon VOP

2. The BAB is flagged opaque. In this case, no shape coding is necessary, but texture information is coded.
3. The BAB is coded without motion compensation using IntraCAE.
4. The **MVDs** is zero (i.e., **MVs** = **MVPs**) and no block update is necessary.
5. The **MVDs** is zero and the block needs to be updated. In this case, InterCAE is used for coding the block update.
6. The **MVDs** is nonzero and no update is necessary.
7. The **MVDs** is nonzero and the block needs to be updated. In this case, InterCAE is used for coding the block update.

Modes 1 and 2 require no shape coding. For mode 3, shape is encoded using IntraCAE. For modes 4–7, motion estimation and compensation are employed. The motion vector difference for shape (**MVDs**) is the difference between the shape motion vector (**MVs**) and its predictor (**MVPs**). This predictor is estimated from either neighboring shape motion vectors or co-located texture motion vectors. When the mode indicates that no update is required, then the **MVs** is simply used to copy a displaced 16×16 block from the reference binary alpha plane to the current BAB. If, however, the mode indicates that an update is required, then the update is coded using InterCAE.

The CAE is a binary arithmetic coding algorithm where the probability of a symbol is determined from the *context* of the neighboring symbols. First, the arithmetic encoder is initialized. The binary pels (elements) of the BAB are then encoded in raster-scan order using the following steps:

1. Compute a context number based on the templates shown in Figure 3.13. This context number is given by $C = \sum_{k=0}^{N-1} c_k 2^k$, where $c_k = 0$ for a



(a) IntraCAE (b) InterCAE

Figure 3.13: CAE templates in MPEG-4

transparent pel, $c_k = 1$ for an opaque pel, $N=10$ pels for IntraCAE, and $N = 9$ pels for InterCAE.

2. Determine the probability of the pel being transparent (or opaque) by using the context number to index a table of probabilities defined by the standard.

3. Use the indexed probability to drive an arithmetic encoder for codeword assignment.

When all pels in the BAB have been encoded, the arithmetic encoder is terminated.

3.5.2.2 Gray-Scale Shape Coding

A gray-scale alpha plane has a similar representation to the binary alpha plane, with the difference that elements within the plane can take on a range of values, usually 0 to 255 with 8-bit representation, designating the degree of transparency of the corresponding pel. Gray-scale shape information consists of two parts. The first part is the support information. This is obtained by thresholding the gray-scale alpha plane at 0 (i.e., any value that is not equal to 0 is set to 255). Support information is encoded using the binary shape coding methods described previously. The second part of gray-scale shape information contains the gray-scale values of the alpha plane. This is encoded using methods similar to the texture encoding methods described later in this chapter (Section 3.5.4).

3.5.2.3 Scalable Shape Coding

Besides changing the coding mode of BABs, additional mechanisms are employed for controlling the quality and bit rate of binary shape information. One method is by reducing the resolution of the BAB by a factor of 2 or 4. The resulting 8×8 or 4×4 BAB is encoded using any of the available modes. At the decoder, the reduced-resolution BAB is first decoded and then upsampled. Another method for reducing the binary shape bit rate is by changing the orientation of the BAB. The efficiency of the CAE algorithm can depend on the orientation of the BAB. In some cases, transposing the BAB before coding it can increase coding efficiency. In this case, the decoder decodes the BAB and then transposes it back to its original orientation.

3.5.3 Motion Estimation and Compensation

Motion estimation and compensation methods in MPEG-4 are very similar to those employed by other standards. The main difference is that block-

based motion estimation and compensation are adapted to the arbitrary-shape VOP structure of MPEG-4. The standard has three modes for encoding a given VOP: intra-VOP (I-VOP), predicted-VOP (P-VOP), and bidirectionally-predicted-VOP (B-VOP).

Since the shape, size, and location of a VOP can change from one instance to another, the absolute (frame) coordinate system is used for referencing every VOP. Thus, the motion vector for a particular feature inside a VOP refers to the displacement of the feature in absolute coordinates. During motion estimation and compensation, no alignment of VOP bounding boxes at different time instances is performed.

Motion is estimated only for those MBs within the bounding box of the current VOP. If the current MB is an internal MB, then motion is estimated using the usual block-matching method. If, however, the current MB is a boundary MB, then motion is estimated using a modified block-matching method called *polygon matching*. In polygon matching, the distortion measure is calculated using only those pels in the current macroblock that belong to the VOP.

The motion estimation and compensation processes may require accessing pels outside the reference VOP. Padding is used to define the values of such pels. The luma component is padded per 16×16 samples, while the chroma components are padded per 8×8 samples. If the reference MB is a boundary MB, then it is padded using *repetitive padding*. This process starts by horizontal repetitive padding, where each sample at the boundary of a reference VOP is replicated horizontally in the left and/or right direction in order to fill the transparent region of the reference MB. If there are two boundary sample values for filling a sample, the two boundary samples are averaged. The remaining unfilled transparent samples are padded by a similar process as the horizontal repetitive padding but in the vertical direction, i.e., vertical repetitive padding. The remaining MBs within the reference VOP are exterior MBs. Such MBs are filled by *extended padding*. In this method, samples of an exterior MB are filled by replicating the samples at the border of the neighboring boundary MB. If an exterior MB is next to more than one boundary MB, then one of the boundary MBs is chosen according to a priority criterion defined by the standard. The remaining exterior MBs are filled with 128 (for an 8-bit luma component).

Motion vectors are estimated to half-pel accuracy. They are then predictively VLC coded in a similar fashion to the H.263 standard.

Similar to the H.263 standard, MPEG-4 has an advanced prediction mode (four motion vectors per MB and unrestricted motion vectors) and an overlapped motion compensation mode.

3.5.4 Texture Coding

For I-VOPs, texture refers to the luma and chroma values (i.e., the video signal). For motion-compensated VOPs, texture refers to the luma and chroma residual errors remaining after motion compensation (i.e., the DFD signal). The process of texture coding involves the following steps: padding, DCT, quantization, INTRA coefficient prediction, scanning, and variable-length encoding.

3.5.4.1 Padding

Like H.263 and most other video coding standards, MPEG-4 encodes texture information using a block-based 8×8 DCT. In this process, internal MBs are encoded directly, whereas boundary MBs must first be padded. The aim of this padding process is to remove abrupt transitions within the macroblock, thus reducing the number of significant DCT coefficients. Note that during texture coding, exterior MBs are not coded.

For motion-compensated boundary MBs, pels outside the VOP are padded with zero. For INTRA boundary MBs, pels outside the VOP are padded using the following *low-pass extrapolation* (LPE) procedure:

1. Calculate the mean value of the macroblock pels that lie within the VOP. Use this value for padding the macroblock pels that lie outside the VOP.
2. Starting at the top-left corner of the macroblock, proceed in scanning order to the bottom-right corner, replacing each pel $f(x, y)$ that lies outside the VOP with the average value of its four neighbors; i.e., $f(x, y) = (f(x-1, y) + f(x+1, y) + (f(x, y-1) + f(x, y+1))/4$. The neighboring pels should lie within the VOP; otherwise they are not considered in the averaging process and the equation is modified accordingly.

3.5.4.2 DCT

The internal MBs and the padded boundary MBs are then transformed using a 2-D 8×8 forward DCT.

3.5.4.3 Quantization

The resulting DCT coefficients are quantized using one of two methods. The first method is very similar to H.263 quantization and uses a fixed quantization step size for the whole macroblock. The second method, however, uses one of two default quantization matrices (or scaled versions of them) to modify the

quantizer step size depending on the spatial frequency of the coefficient. In MPEG-4, DC coefficients can also be quantized using a nonlinear quantizer.

3.5.4.4 Prediction of INTRA DCT Coefficients

To achieve more efficiency, the quantized coefficients of an INTRA block can be predicted from the colocated coefficients in either the block immediately to the left of or the block immediately above the current block, as shown in Figure 3.14. The direction of prediction is adapted depending on the horizontal and vertical DC gradients of neighboring blocks. Thus, if X is the current INTRA block, $\mathrm{QF}_A(0,0)$ is the quantized DC coefficient of block A immediately to the left of the current block, $\mathrm{QF}_B(0,0)$ is the quantized DC coefficient of block B above and to the left of X, and $\mathrm{QF}_C(0,0)$ is the quantized DC coefficient of block C immediately above X, then the direction of prediction is chosen as follows: If $|\mathrm{QF}_A(0,0)-\mathrm{QF}_B(0,0)| < |\mathrm{QF}_B(0,0)-\mathrm{QF}_C(0,0)|$, then predict from block C; otherwise predict from block A.

Having decided the direction of prediction, there are two types of prediction:

1. *DC prediction*: Depending on the direction of prediction, the DC coefficient of the current block X is predicted from the DC coefficient of either block A or block C. For example, when the horizontal direction is chosen, the prediction is given by $\mathrm{PQF}_X(0,0) = \mathrm{QF}_X(0,0) - \mathrm{QF}_A(0,0)$.

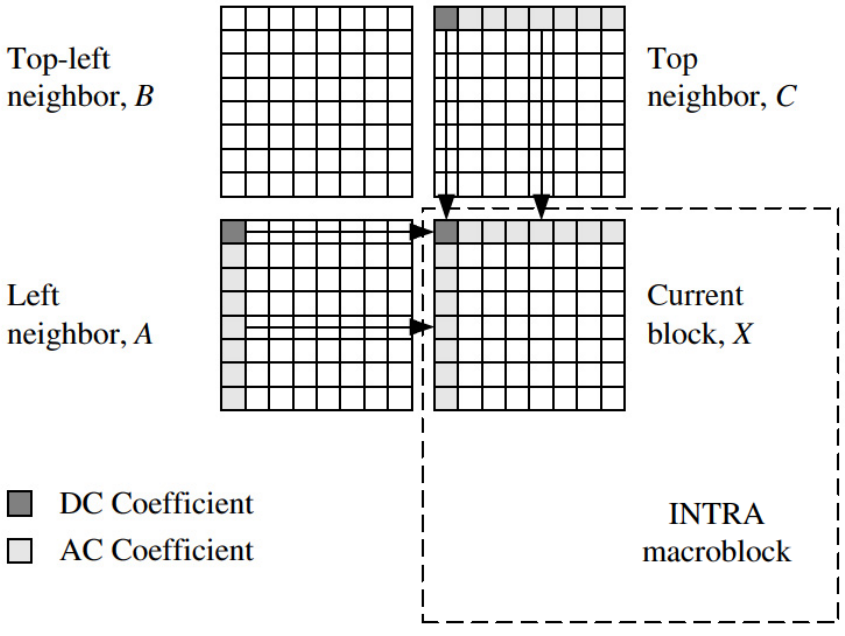


Figure 3.14: DC and AC coefficient adaptive prediction in MPEG-4

2. *AC prediction*: Depending on the direction of prediction, the AC coefficients of the first row of the current block X are predicted from the AC coefficients of the first row of block C, or the AC coefficients of the first column of the current block X are predicted from the AC coefficients of the first column of block A. To compensate for differences in the quantization parameters of adjacent blocks used in AC prediction, the prediction process is modified so that the predictor is scaled by the ratio of the current quantization parameter, QP_X, and the quantization parameter of the predictor block, QP_A or QP_C. For example, when the horizontal direction is chosen, the prediction is given by $\mathrm{PQF}_X(0,j) = \mathrm{QF}_X(0,j) - \frac{\mathrm{QF}_A(0,j)\mathrm{QP}_A}{\mathrm{QP}_X}$. The use of AC prediction can be enabled/disabled at the macroblock level.

If any of the neighboring blocks are outside of the VOP boundary or the video packet boundary, or if they do not belong to an INTRA coded macroblock, their DC values are assumed to take a value of $2^{\text{bits/pel}+2}$ and their AC values are assumed to take a value of 0. DC and AC predictions are performed similarly for the luma and each of the two chroma components.

3.5.4.5 Scanning

To prepare the coefficients for variable-length encoding, a scanning process is used to convert the 2-D matrix of coefficients into a 1-D vector.

There are three possible scanning patterns: zigzag, alternate-vertical, and alternate-horizontal. All non-INTRA blocks use the conventional zigzag scanning pattern. For INTRA blocks, however, the choice of the scanning pattern depends on the prediction process:

1. If AC prediction is *not* employed, then the conventional zigzag scanning pattern is used for all blocks within the macroblock.
2. If, however, AC prediction is employed, then the direction of the DC prediction is used to select a suitable scanning pattern on a block basis, as follows:
   (a) If the DC prediction employs the horizontal direction, then the alternate-vertical scanning pattern is used.
   (b) If, however, the DC prediction employs the vertical direction, then the alternate-horizontal scanning pattern is used.

3.5.4.6 Variable-Length Coding

The differential (predicted) DC coefficients in INTRA macroblocks are encoded using a concatenation of a VLC codeword and a FLC codeword. The possible range of encoded differential DC coefficients is divided into subranges

or categories. The VLC codeword indicates to which category the encoded difference belongs, whereas the FLC codeword, then, uniquely identifies the difference within that category. Instead of this special treatment, the INTRA DC coefficients can optionally be encoded using the same INTRA AC VLC table described next. To achieve compatibility with H.263, the INTRA DC coefficients can also optionally be encoded without prediction using an 8-bit FLC codeword.

All other coefficients are encoded using a procedure similar to that of H.263. Thus, the scanned quantized coefficients are converted into an intermediate set of EVENTS of the form (LAST, RUN, LEVEL). The most commonly occurring events are then encoded using standard VLC tables. There are two standard VLC tables: one for INTRA blocks and another for INTER blocks. To achieve compatibility with H.263, the VLC table for INTER blocks can optionally be used for both INTER and INTRA blocks. Less frequent EVENTS are encoded with the help of an ESCAPE codeword.

3.5.5 Still-Texture Coding

The MPEG-4 also supports coding of static textures (or still images). This mode uses subband coding based on the discrete wavelet transform (DWT).

As discussed in Section 2.6.3, the DWT is used in subband coding to apply a nonuniform decomposition (refer to Figure 2.12(b)) to the texture information. This results in a decomposition tree of subbands. The lowest subband (horizontal low/vertical low (LL)) is known as the DC subband, whereas the remaining subbands are known as the AC subbands.

In MPEG-4, the DWT can be either a floating-point or an integer transform, as signaled by the encoder in the bitstream. The encoder can also choose to use a set of default filters or to use its own filters and define them in the bitstream.

The quantized coefficients of the DC subband are encoded using DPCM followed by arithmetic coding. The choice of the predictor for a particular coefficient depends on the magnitude of the horizontal and vertical gradients of neighboring coefficients. If the horizontal gradient is smaller than the vertical gradient, then prediction is performed using the left neighboring coefficient; otherwise the top neighboring coefficient is employed.

The quantized coefficients of the AC subbands are encoded using a zero-tree algorithm followed by arithmetic coding.

3.5.6 Sprite Coding

An interesting mode supported by MPEG-4 is sprite coding. A *sprite* consists of those parts of an object that are present in the scene throughout a video

segment. For example, a background sprite (also referred to in the literature as a *background mosaic*) can be constructed by collecting all pels belonging to the background throughout a video segment. Note that in the case of camera panning, for example, the background sprite can be larger than the actual frames of the sequence. This still, and possibly large, image needs to be transmitted only once before transmitting the corresponding video segment. For each frame of the video segment, there is no need to encode a background VOP. Instead, a small number of parameters needs to be transmitted to allow the decoder to warp/crop the sprite and generate an appropriate background VOP. Thus, in such cases, sprite coding can achieve very high coding efficiency.

Sprite coding can operate in three modes: basic sprite coding, low-latency sprite coding, and scalable sprite coding. In *basic sprite coding* the whole sprite is encoded and transmitted to the decoder before transmitting the corresponding video segment. In *low-latency sprite coding* only part of the sprite is encoded and transmitted. This part is sufficient to be used for the first few frames of the video segment. The remaining part of the sprite is transmitted, piecewise, when required or as the bandwidth allows. In *scalable sprite coding* the sprite is encoded and transmitted progressively. In other words, a low-quality version of the sprite is encoded and transmitted first. This is then refined gradually by encoding and transmitting residuals.

3.5.7 Scalability

MPEG-4 supports both temporal and spatial scalability using multiple VOLs. For example, in the case of two VOLs, one VOL provides the base layer whereas the other VOL provides the enhancement layer.

MPEG4 uses a *generalized scalability* framework, as shown in Figure 3.15. In this framework the functionality of a block depends on the chosen type of scalability.

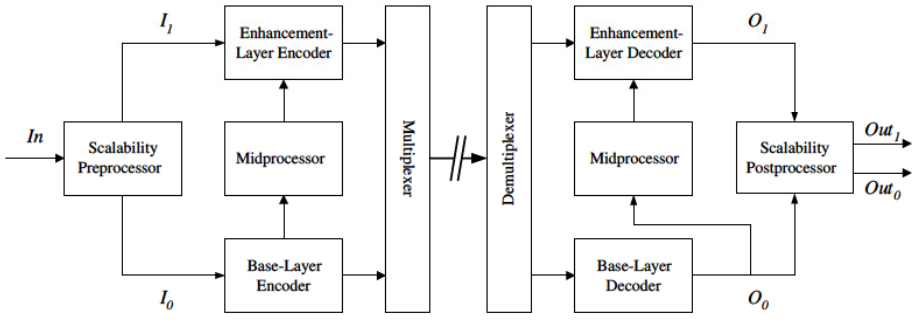


Figure 3.15: MPEG-4 generalized scalability

VOPs are input to the *scalability preprocessor*. If spatial scalability is to be performed, then this preprocessor downsamples the input VOPs to generate the base-layer VOPs forming the input to the *base-layer encoder*. The *midprocessor* takes the reconstructed base-layer VOPs and upsamples them. The difference between the original VOPs and the output of the *midprocessor* forms the enhancement-layer VOPs. Those are encoded using the *enhancement-layer encoder*. The *multiplexer* is then used to multiplex the base- and enhancement-layer bitstreams into a single bitstream. At the decoder, the *demultiplexer* is used to separate the incoming bitstream into base- and enhancement-layer bitstreams. The *scalability postprocessor* performs any necessary operations, such as upsampling the decoded base layer for display.

If, however, temporal scalability is to be performed, then the *scalability preprocessor* separates the stream of input VOPs into two substreams. One substream forms the input to the *base-layer encoder*, while the other forms the input to the *enhancement-layer encoder*. In this case, the *midprocessor* does not perform any spatial resolution conversion and simply allows the reconstructed base-layer VOPs to pass through to be used for the temporal prediction of enhancement-layer VOPs. In this case also, the *postprocessor* simply outputs the reconstructed base-layer VOPs without any conversion.

For spatial scalability, only rectangular VOPs are supported by MPEG-4. In the case of temporal scalability, however, both rectangular and arbitrary-shaped VOPs are supported. MPEG-4 provides two types of temporal scalability:

- *Type I*: The enhancement layer increases the temporal resolution of only a partial region of the base layer.
- *Type II*: The enhancement layer increases the temporal resolution of the entire region of the base layer.

3.5.8 Error Resilience

One of the main aims of MPEG-4 is to provide universal access through a wide range of environments, including error-prone environments. One of the important requirements of video communication over error-prone environments, like mobile networks, is robustness against errors. MPEG-4 provides three main tools for error resilience: resynchronization, data partitioning, and reversible VLCs.

3.5.8.1 Resynchronization

As is discussed in Chapter 9, one of the disadvantages of VLC coding is that errors in the bitstream can cause a loss of synchronization between the encoder and the decoder. One way to reduce this effect is to insert unique

markers called *resynchronization codewords* in the bitstream. When an error is detected, the decoder skips the remaining bits until it finds a resynchronization codeword. This reestablishes the synchronization with the encoder, and the decoder then proceeds to decode from that point on.

Version 1 of H.263 adopts a GOB-based resynchronization approach. This means that a resynchronization codeword is inserted every time a fixed number of macroblocks (which is equal to the size of the GOB) has been encoded. Since the number of bits can vary between macroblocks, the resynchronization codewords will most likely be unevenly spaced throughout the bitstream. Therefore, certain parts of the sequence, such as high-motion areas with high bit content, will be more susceptible to errors and will also be more difficult to conceal.

MPEG-4, however, adopts a more robust approach based on video packets, as illustrated in Figure 3.16(a). In this approach each packet contains approximately the same number of bits. This means that the resynchronization codewords are almost periodic in the bitstream. Note that the header of the packet contains the necessary information (e.g., the address of the first MB in the packet and the corresponding quantization parameter) to restart decoding after reestablishing synchronization. Following the packet header is the *header extension code* (HEC). When this bit is set to "1," then additional information (e.g., timing information and VOP coding type) are included in the header.

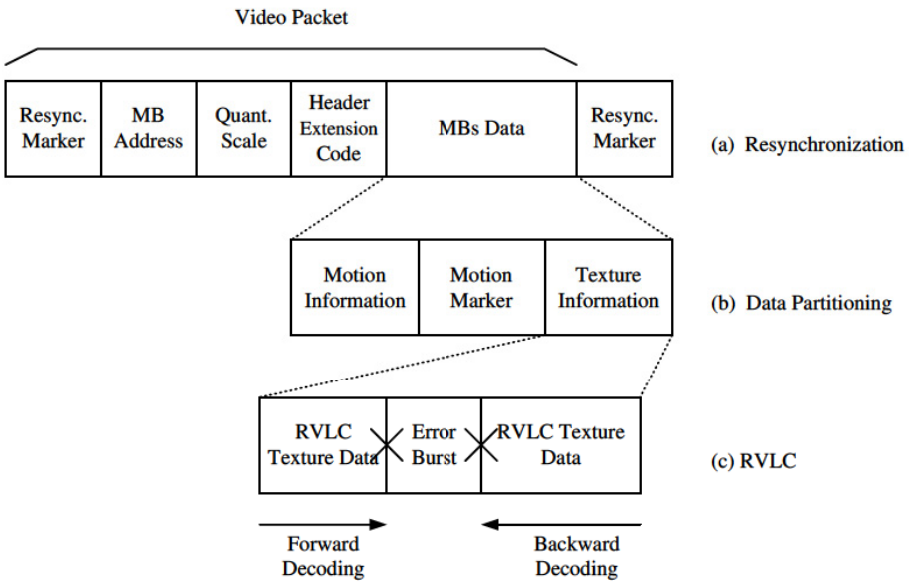


Figure 3.16: MPEG-4 error resilience tools

Such information was originally included in the VOP header. Its inclusion in the packet header as well enables the decoder to decode the packet without reference to the packet containing the VOP header. Such information can also help error detection, since it is supposed to be the same in all packets belonging to the same VOP.

Another problem with VLC coding is that errors can emulate the occurrence of start and resynchronization codewords. To reduce this effect, MPEG-4 provides a second resynchronization approach called *fixed-interval synchronization*. In this approach, VOP start codes and packet resynchronization codewords appear only at legal fixed-interval locations in the bitstream. Thus, only codewords at those legal locations will be used by the decoder to reestablish synchronization.

3.5.8.2 Data Partitioning

In some cases, an error occurs well before the point in the bitstream at which the error is detected. Therefore, when an error is detected, all bits between the resynchronization codeword prior to the error detection point and the resynchronization codeword where synchronization is reestablished are typically discarded. If the decoder can localize the error more effectively, then the performance of error concealment techniques (discussed in Chapter 9) can be improved.

MPEG-4 uses data partitioning to further improve the ability of the decoder to localize errors. In this approach the bitstream between two resynchronization codewords is divided into smaller logical units. Each logical unit contains one type of information for *all* MBs belonging to the same packet. For example, in Figure 3.16(b) the motion information for all the MBs in the packet is encoded first, followed by a motion marker and then the texture information for all the MBs in the packet. In the non-data-partitioned case, if an error occurs in the texture information, then the header, motion, and texture information will all be discarded. In the data-partitioned case, however, if an error occurs in the texture information, then only the texture information will be discarded, and the motion marker will be used to locate and recover the header and motion information. Temporal concealment (described in Chapters 9 and 10) can then use this recovered information to conceal the corrupted MBs from the reference VOP.

3.5.8.3 Reversible VLCs

As already discussed, when an error is detected in the bitstream, the bits between the surrounding resynchronization codewords are discarded, and the decoder skips to the next resynchronization codeword and proceeds decoding from there. In MPEG-4, however, texture information is encoded using

RVLCs, as illustrated in Figure 3.16(c). In this case, when the decoder jumps to the next resynchronization codeword, instead of discarding all preceding bits, the decoder can start decoding in the reverse direction to recover and utilize some of those bits.

3.5.9 Profiles and Levels

As already discussed, profiles and levels provide a means of defining subsets of the syntax and semantics of a standard. This in turn provides a means of defining the decoder capabilities required to decode a particular bitstream. Profiles and levels are used to define conformance points that facilitate bitstream interchange among different applications.

In MPEG-4, *object types* are used to define profiles. An object type defines a subset of MPEG-4 tools that provides a single or a group of functionalities. There are six natural video object types: simple, core, main, simple scalable, *N*-bit, and still scalable texture. For example, the *main object type* includes the following subset of tools: basic (I- and P-VOP, coefficient prediction, 4-MV, and unrestricted MV), error resilience, short header, B-VOP, Methods 1 and 2 for quantization, P-VOP-based temporal scalability, binary shape, gray shape, interlace, and sprite.

A *profile* is a defined subset of the entire bitstream syntax. MPEG-4 defines six natural video profiles: simple, core, main, simple scalable, *N*-bit, and scalable texture. Each profile is defined in terms of video object types. For example, the *main profile* includes the following object types: simple, core, main, and scalable still texture.

A *level* within a profile is a defined set of constraints imposed on parameters in the bitstream that relate to the tools of that profile. For example, *level 1* (L1) of the *simple profile* has a typical session size of QCIF, a maximum total number of objects of 4, and a maximum bitrate of 64 kbits/s.

Part II
Coding Efficiency

The radio spectrum is a limited and scarce resource. This puts very stringent limits on the bandwidth available for a mobile channel. Given the enormous amount of data generated by video, the use of efficient coding techniques is vital.

One of the most important factors that decide the coding efficiency of a video codec is the motion estimation and compensation technique. This part contains three chapters. Chapter 4 covers some basic motion estimation methods. It starts by introducing some of the fundamentals of motion estimation. It then reviews some basic motion estimation methods, with particular emphasis on the widely used block-matching methods. The chapter then presents the results of a comparative study between the different methods. The chapter also investigates the efficiency of block-matching motion estimation at very low bit rates, typical of mobile video communication. The aim is to decide if the added complexity of this process is justifiable, in terms of an improved coding efficiency, at such bit rates.

Chapter 5 investigates the performance of the more advanced warping-based motion estimation methods. The chapter starts by describing a general warping-based motion estimation method. It then considers some important parameters, like the shape of the patches, the spatial transformation used, and the node-tracking algorithm. The chapter then assesses the suitability of warping-based methods for mobile video communications. In particular, the chapter compares the coding efficiency and the computational complexity of such methods to those of block-matching methods.

Chapter 6 investigates the performance of another advanced motion estimation method, called multiple-reference motion-compensated prediction (MR-MCP). The chapter starts by briefly reviewing multiple-reference motion estimation methods. It then concentrates on the long-term memory motion-compensated prediction (LTM-MCP) technique. The chapter investigates the prediction gains and the coding efficiency of this technique at very low bit rates. The primary aim is to decide if the added complexity, increased motion overhead, and increased memory requirements of this technique are justifiable at such bit rates. The chapter also investigates the properties of multiple-reference block motion fields and compares them to those of single-reference fields.

Chapter 4

Basic Motion Estimation Techniques

4.1 Overview

Motion estimation is an important process in a wide range of disciplines and applications, such as image sequence analysis, computer vision, target tracking, and video coding. Different disciplines and applications have different requirements and may, therefore, use different motion estimation techniques.

This chapter reviews some basic motion estimation techniques developed specifically for video coding. It then carries out a comparative study between the different techniques. The chapter also presents the results of an investigation into the efficiency of block-matching motion estimation at very low bit rates. In particular, the investigation shows that the added complexity of this process is justifiable at such bit rates.

Section 4.2 gives a brief introduction to the basics of motion estimation. Sections 4.3–4.6 briefly review the differential, pel-recursive, frequency-domain, and block-matching motion estimation methods. Section 4.7 presents the results of a comparative study of the reviewed techniques, whereas Section 4.8 investigates the efficiency of motion estimation at very low bit rates. The chapter concludes with a discussion in Section 4.9.

4.2 Motion Estimation

As already discussed in Chapter 2 (Section 2.7.2), the most commonly used video coding method is motion-compensated coding. In the first stage of this method, called *motion estimation* (ME), the motion of objects between a reference frame and the current frame is estimated. This motion information is then used in the second stage, called *motion compensation* (MC), to move the objects of the reference frame to provide a prediction for the current