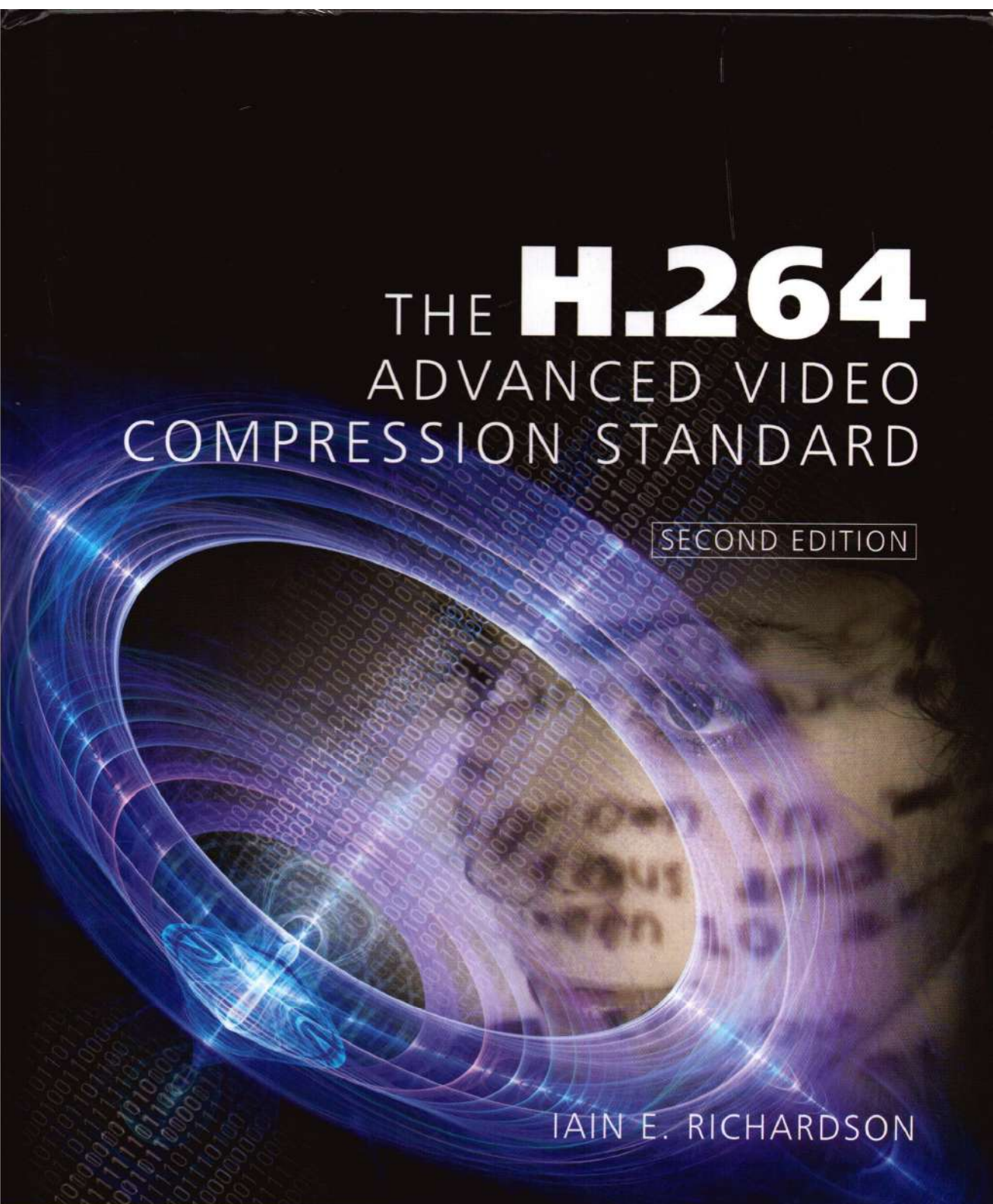




THE **H.264** ADVANCED VIDEO COMPRESSION STANDARD

SECOND EDITION

IAIN E. RICHARDSON

 **WILEY**

THE H.264 ADVANCED VIDEO COMPRESSION STANDARD

SECOND EDITION

IAIN E. RICHARDSON, *Vcodex Limited, UK*

H.264 Advanced Video Coding or MPEG-4 Part 10 is fundamental to a growing range of markets such as high definition broadcasting, internet video sharing, mobile video and digital surveillance. This second edition of *The H.264 Advanced Video Compression Standard* reflects the growing importance and implementation of H.264 video technology. Offering a detailed overview of the system, it explains the syntax, tools and features of H.264 and equips readers with practical advice on how to get the most out of the standard.

- Packed with clear examples and illustrations to explain H.264 technology in an accessible and practical way.
- Covers basic video coding concepts, video formats and visual quality.
- Explains how to measure and optimise the performance of H.264 and how to balance bitrate, computation and video quality.
- Analyses recent work on scalable and multi-view versions of H.264, case studies of H.264 codecs and new technological developments such as the popular High Profile extensions.
- An invaluable companion for developers, broadcasters, system integrators, academics and students who want to master this burgeoning state-of-the-art technology.

"[This book] unravels the mysteries behind the latest H.264 standard and delves deeper into each of the operations in the codec. The reader can implement (simulate, design, evaluate, optimize) the codec with all profiles and levels. The book ends with extensions and directions (such as SVC and MVC) for further research."

PROFESSOR K. R. RAO, THE UNIVERSITY OF TEXAS AT ARLINGTON,
CO-INVENTOR OF THE DISCRETE COSINE TRANSFORM



Enjoyed this book?

Why not tell others about it
and write a review on your
favourite online bookseller.



 **WILEY**
wiley.com



University Libraries
Carnegie Mellon University,
Pittsburgh, PA 15213-3890

THE H.264 ADVANCED VIDEO COMPRESSION STANDARD

Second Edition

Iain E. Richardson

Vcodex Limited, UK



A John Wiley and Sons, Ltd., Publication

2
This edition first published 2010
© 2010, John Wiley & Sons, Ltd

First Edition published in 2003

Registered office

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom

For details of our global editorial offices, for customer services and for information about how to apply for permission to reuse the copyright material in this book please see our website at www.wiley.com.

The right of the author to be identified as the author of this work has been asserted in accordance with the Copyright, Designs and Patents Act 1988.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, except as permitted by the UK Copyright, Designs and Patents Act 1988, without the prior permission of the publisher.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Designations used by companies to distinguish their products are often claimed as trademarks. All brand names and product names used in this book are trade names, service marks, trademarks or registered trademarks of their respective owners. The publisher is not associated with any product or vendor mentioned in this book. This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Library of Congress Cataloguing-in-Publication Data

Richardson, Iain E. G.

The H.264 advanced video compression standard / Iain E. Richardson. – 2nd ed.
p. cm.

Rev. ed. of: H.264 and MPEG-4 video compression. c2003.

Includes bibliographical references and index.

ISBN 978-0-470-51692-8 (pbk.)

1. Digital video—Standards. 2. Video compression—Standards. 3. MPEG (Video coding standard) 4. Multimedia systems. I. Richardson, Iain E. G. H.264 and MPEG-4 video compression. II. Title.

TK6680.5.R52 2010

006.6'96—dc22

2009054387

A catalogue record for this book is available from the British Library.

ISBN: 978-0-470-51692-8

Typeset in 10/12pt Times by Aptara Inc., New Delhi, India

Printed and bound in Great Britain by CPI Autony Rowe, Chippenham, Wiltshire

TK
6680.5
.R52
2010

To Pat

*Language is living,
but what is most important
goes deeper than words.*

Contents

About the Author	ix
Preface	xv
Glossary	xvii
List of Figures	xxi
List of Tables	xxiii
1 Introduction	1
1.1 A change of name	1
1.2 Getting the change	4
1.3 The role of standards	4
1.4 Why is 264 Advanced Video Coding so important?	4
1.5 An agenda for this book	5
1.6 Acknowledgments	5
2 Video format and quality	7
2.1 Introduction	7
2.2 Analog video format	7
2.3 Digital video	8
2.3.1 Spatial sampling	8
2.3.2 Temporal sampling	9
2.3.3 Color and fields	11
2.4 Bitstream space	15
2.4.1 RGB	15
2.4.2 YCbCr	15
2.4.3 TV-Chroma subsampling formats	16
2.5 Signal formats	16
2.5.1 Interlaced video formats	16
2.5.2 Standard Definition	17
2.5.3 High Definition	18
2.6 Quality	19
2.6.1 Subjective Quality measurement	19
2.6.2 Objective quality measurement	20

University Libraries
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Contents

About the Author

Preface

Glossary

List of Figures

List of Tables

1	Introduction	1
1.1	A change of scene	1
1.2	Driving the change	4
1.3	The role of standards	4
1.4	Why H.264 Advanced Video Coding is important	4
1.5	About this book	5
1.6	Reference	6
2	Video formats and quality	7
2.1	Introduction	7
2.2	Natural video scenes	7
2.3	Capture	8
2.3.1	Spatial sampling	9
2.3.2	Temporal sampling	9
2.3.3	Frames and fields	11
2.4	Colour spaces	12
2.4.1	RGB	12
2.4.2	YCrCb	13
2.4.3	YCrCb sampling formats	14
2.5	Video formats	16
2.5.1	Intermediate formats	16
2.5.2	Standard Definition	17
2.5.3	High Definition	18
2.6	Quality	19
2.6.1	Subjective quality measurement	20
2.6.2	Objective quality measurement	21

2.7	Summary	24
2.8	References	24
3	Video coding concepts	25
3.1	Introduction	25
3.2	Video CODEC	26
3.3	Prediction model	28
3.3.1	Temporal prediction	28
3.3.2	Spatial model: intra prediction	38
3.4	Image model	40
3.4.1	Predictive image coding	41
3.4.2	Transform coding	42
3.4.3	Quantization	50
3.4.4	Reordering and zero encoding	52
3.5	Entropy coder	57
3.5.1	Predictive coding	57
3.5.2	Variable-length coding	58
3.5.3	Arithmetic coding	65
3.6	The hybrid DPCM/DCT video CODEC model	68
3.7	Summary	79
3.8	References	79
4	What is H.264?	81
4.1	Introduction	81
4.2	What is H.264?	81
4.2.1	A video compression format	81
4.2.2	An industry standard	82
4.2.3	A toolkit for video compression	83
4.2.4	Better video compression	83
4.3	How does an H.264 codec work?	83
4.3.1	Encoder processes	85
4.3.2	Decoder processes	89
4.4	The H.264/AVC Standard	91
4.5	H.264 Profiles and Levels	92
4.6	The H.264 Syntax	94
4.7	H.264 in practice	97
4.7.1	Performance	97
4.7.2	Applications	98
4.8	Summary	98
4.9	References	98
5	H.264 syntax	99
5.1	Introduction	99
5.1.1	A note about syntax examples	99
5.2	H.264 syntax	100
5.3	Frames, fields and pictures	101

5.3.1	Decoding order	104
5.3.2	Display order	104
5.3.3	Reference picture lists	106
5.3.4	Frame and field coding	111
5.4	NAL unit	114
5.5	Parameter Sets	115
5.6	Slice layer	117
5.6.1	Slice types	117
5.6.2	Slice header	117
5.6.3	Slice data	118
5.7	Macroblock layer	119
5.7.1	Overview	119
5.7.2	The Intra PCM mode	121
5.7.3	Macroblock prediction	122
5.7.4	Residual data	124
5.7.5	Macroblock syntax examples	127
5.8	Summary	134
5.9	References	135
6	H.264 Prediction	137
6.1	Introduction	137
6.2	Macroblock prediction	137
6.3	Intra prediction	138
6.3.1	4 × 4 luma prediction modes	143
6.3.2	16 × 16 luma prediction modes	146
6.3.3	Chroma prediction modes	147
6.3.4	8 × 8 luma prediction, High profiles	148
6.3.5	Signalling intra prediction modes	148
6.4	Inter prediction	149
6.4.1	Reference pictures	151
6.4.2	Interpolating reference pictures	152
6.4.3	Macroblock partitions	157
6.4.4	Motion vector prediction	158
6.4.5	Motion compensated prediction	162
6.4.6	Inter prediction examples	164
6.4.7	Prediction structures	169
6.5	Loop filter	171
6.5.1	Boundary strength	172
6.5.2	Filter decision	173
6.5.3	Filter implementation	174
6.5.4	Loop filter example	174
6.6	Summary	177
6.7	References	177
7	H.264 transform and coding	179
7.1	Introduction	179

7.2	Transform and quantization	179
7.2.1	The H.264 transforms	179
7.2.2	Transform processes	180
7.2.3	Integer transform and quantization: 4×4 blocks	185
7.2.4	Integer transform and quantization: 8×8 blocks	198
7.2.5	DC transforms	203
7.2.6	Transform and quantization extensions in the High profiles	204
7.3	Block scan orders	206
7.4	Coding	207
7.4.1	Exp-Golomb Coding	208
7.4.2	Context Adaptive Variable Length Coding, CAVLC	210
7.4.3	Context Adaptive Binary Arithmetic Coding, CABAC	217
7.5	Summary	220
7.6	References	221
8	H.264 conformance, transport and licensing	223
8.1	Introduction	223
8.2	Conforming to the Standard	223
8.2.1	Profiles	224
8.2.2	Levels	226
8.2.3	Hypothetical Reference Decoder	230
8.2.4	Conformance testing	236
8.3	H.264 coding tools for transport support	237
8.3.1	Redundant slices	237
8.3.2	Arbitrary Slice Order (ASO)	238
8.3.3	Slice Groups/Flexible Macroblock Order (FMO)	238
8.3.4	SP and SI slices	240
8.3.5	Data partitioned slices	243
8.4	Transport of H.264 data	244
8.4.1	Encapsulation in RBSPs, NALUs and packets	244
8.4.2	Transport protocols	245
8.4.3	File formats	247
8.4.4	Coding and transport issues	247
8.5	Supplemental Information	248
8.5.1	Supplemental Enhancement Information (SEI)	248
8.5.2	Video Usability Information (VUI)	248
8.6	Licensing H.264/AVC	248
8.6.1	Video coding patents	250
8.6.2	Video coding standards and patents	252
8.6.3	Licensing H.264/AVC patents	252
8.7	Summary	253
8.8	References	253
9	H.264 performance	255
9.1	Introduction	255
9.2	Experimenting with H.264	256

9.2.1	The JM Reference Software	256
9.2.2	Other software encoders/decoders	261
9.2.3	H.264 stream analysis	263
9.3	Performance comparisons	265
9.3.1	Performance criteria	265
9.3.2	Performance examples: Foreman sequence, QCIF resolution	265
9.3.3	Performance examples: Foreman and Container sequences	269
9.3.4	Performance examples: Inter prediction structures	271
9.3.5	Performance example: H.264 vs. MPEG-4 Visual	273
9.4	Rate control	274
9.4.1	Rate control in the JM reference encoder	276
9.5	Mode selection	279
9.5.1	Rate Distortion Optimized mode selection	281
9.6	Low complexity coding	283
9.6.1	Approximating the cost function	283
9.6.2	Reducing the set of tested modes	284
9.6.3	Early termination	285
9.7	Summary	285
9.8	References	285
10	Extensions and directions	287
10.1	Introduction	287
10.2	Scalable Video Coding	288
10.2.1	Simulcast transmission	288
10.2.2	Scalable transmission	289
10.2.3	Applications of Scalable Video Coding	290
10.2.4	Scalable Video Coding in H.264	290
10.2.5	Temporal scalability	292
10.2.6	Quality scalability: overview	294
10.2.7	Spatial scalability: overview	294
10.2.8	Spatial scalability in detail	294
10.2.9	Quality scalability in detail	298
10.2.10	Combined scalability	299
10.2.11	SVC performance	299
10.3	Multiview Video Coding	302
10.3.1	H.264 Multiview Video Coding	304
10.4	Configurable Video Coding	306
10.4.1	MPEG Reconfigurable Video Coding	307
10.4.2	Fully Configurable Video Coding	308
10.5	Beyond H.264/AVC	310
10.6	Summary	310
10.7	References	311
Index		313

It should be emphasized that the syntax examples, diagrams and explanations in this chapter are for illustration only. Complete and accurate explanations of the syntax and the semantics or meaning of each syntax element are given in the H.264/AVC standard.

5.2 H.264 syntax

The hierarchical organization of the H.264/AVC syntax can be seen in Figure 5.1 [ii]. The **Network Abstraction Layer** (NAL) consists of a series of **NAL Units** (section 5.4). Sequence Parameter Sets (SPS) and Picture Parameter Sets (PPS) are NAL units that signal certain common control parameters to the decoder (section 5.5). Coded video data is communicated in **Video Coding Layer** (VCL) NAL units, known as coded **slices**. An **access unit**, a coded

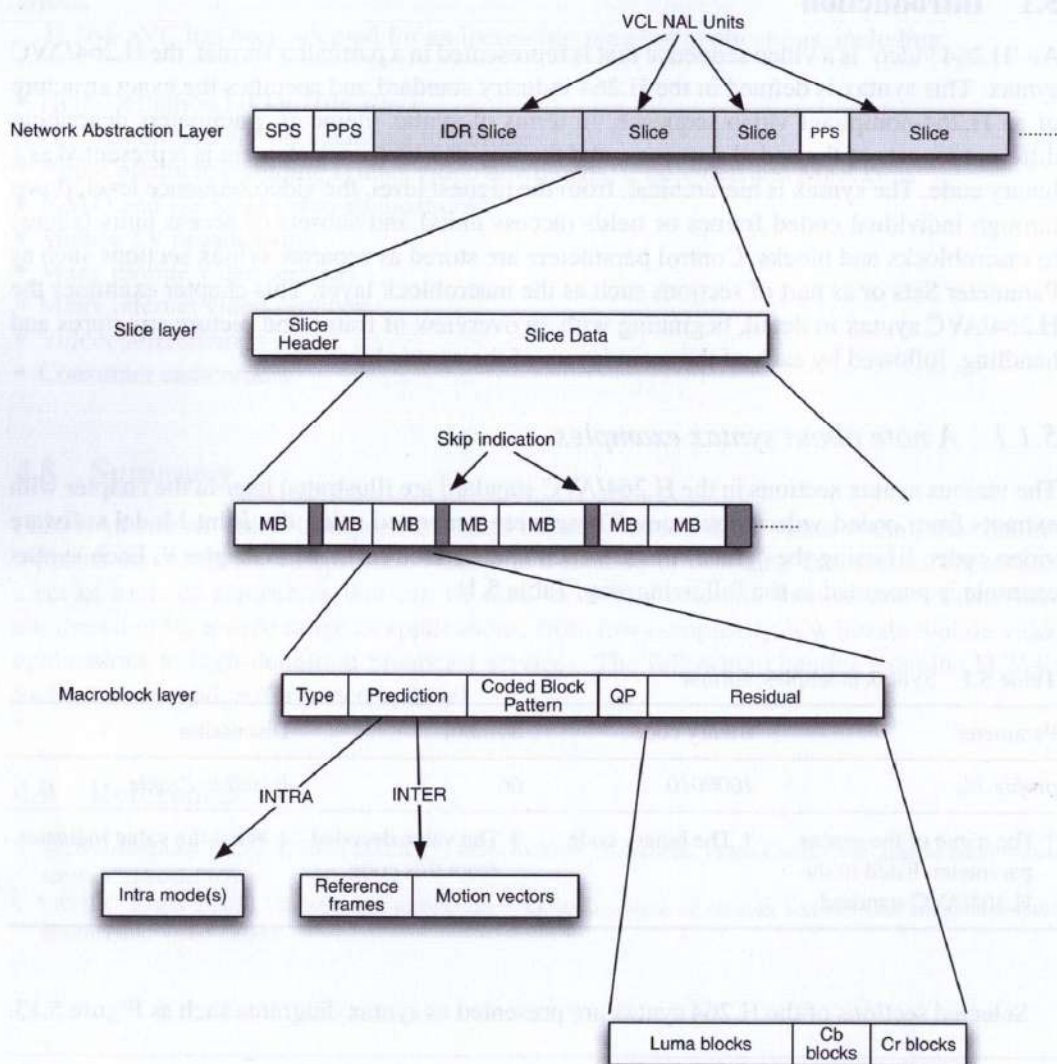


Figure 5.1 Syntax overview

frame or field, is made up of one or more slices. At the **slice layer** (section 5.6), each slice consists of a Slice Header and Slice Data. The Slice Data is a series of coded **macroblocks** (MB) and skip macroblock indicators which signal that certain macroblock positions contain no data. Each coded macroblock (section 5.7) contains the following syntax elements:

- MB type : I/intra coded, P/inter coded from one reference frame, B/inter coded from one or two reference frames.
- Prediction information : prediction mode(s) for an I macroblock, choice of reference frame(s) and motion vectors for a P or B macroblock.
- Coded Block Pattern CBP : indicates which luma and chroma blocks contain non-zero residual coefficients.
- Quantization Parameter QP, for macroblocks with CBP $\neq 0$.
- Residual data, for blocks containing non-zero residual coefficients.

A **coded video sequence** begins with an Instantaneous Decoder Refresh (IDR) Access Unit, made up of one or more IDR slices, a special type of Intra coded slice. Subsequent video frames or fields, described as Access Units, are coded as slices. The video sequence ends when a new IDR slice is received, signalling a new coded sequence, or when the transmission is complete.

A list of all the main sections of the syntax is given in Table 5.2. A NAL unit contains a **Raw Byte Sequence Payload (RBSP)**, a sequence of bytes containing syntax elements. H.264 syntax elements are binary codes with varying length and so a sequence of syntax elements within a NAL unit will not necessarily fit into an integral number of bytes. Zero-valued bits are added to the end of the RBSP contents in order to ensure that the length is an integral number of bytes (**RBSP trailing bits**). An RBSP syntax element can be transmitted or stored as a complete packet. Some syntax sections contain sub-sections, e.g. the Sequence Parameter Set may contain the sections *scaling_list* and *vui_parameters* and always contains the section *rbsp_trailing_bits*. Other sections, e.g. *scaling_list*, do not contain any sub-sections.

Before discussing the syntax sections in detail, it is necessary to examine the way in which H.264/AVC handles frames, fields and pictures.

5.3 Frames, fields and pictures

Chapter 2 introduced the concept of digital video and typical formats for storing and handling digital video information. An H.264 encoder converts video frames into compressed or coded pictures. H.264/AVC defines a **frame** as an array of luma samples and two corresponding arrays of chroma samples. The two **fields**, top field and bottom field, that make up a frame may be sampled at the same time instant, progressive scan, or at different time instants, interlaced scan. The term **picture** refers collectively to a frame or a field. Figure 5.2 is an overview of the way pictures are handled during H.264 encoding and decoding. Frames or fields are encoded to form coded pictures, each of which is composed of one or more slices. Slices are decoded to produce decoded pictures which are stored in a Decoded Picture Buffer (DPB). Pictures in the DPB may be used for inter prediction of further coded pictures and/or output for display.

It is important to distinguish between three different orders of pictures: decoding order, the order in which pictures are decoded from the bitstream, display order, the order in which pictures are output for display, and reference order, the order in which pictures are arranged for inter prediction of other pictures (Figure 5.3).

5.4 NAL unit

Coded H.264 data is stored or transmitted as a series of packets known as Network Abstraction Layer Units, NAL Units or NALUs. Each NAL Unit consists of a 1-byte NALU header followed by a byte stream containing control information or coded video data. The header indicates the NALU type, some of which are listed in Table 5.6, and the ‘importance’ of the NALU. Parameter Sets and slices that are used for reference, i.e. used to predict further frames, are considered important or high priority, since their loss could make it difficult to decode subsequent coded slices. Non-reference slices are considered to be less important to the decoder, since their loss will not affect any further decoding. This information can optionally be used to prioritise certain NALUs during transmission (Chapter 8).

Coded slices are described as Video Coding Layer (VCL) NAL units. A coded sequence begins with an Instantaneous Decoder Refresh (IDR) Access Unit, made up of one or more IDR slices, each of which is an Intra coded slice. This is followed by the default slice type, i.e. a non-IDR coded slice, and/or by Data Partitioned slices. Data Partitioned slices carry different components of coded video data in separate NAL units, which may be useful in error-prone transport situations. Non-VCL NAL units include Parameter Sets, Supplemental Enhancement Information parameters that may be useful for decoding and displaying video data, but are not essential for correct decoding, and delimiters that define boundaries between coded sections.

Each NALU may be transported in one or more network packets, streamed sequentially as a byte stream or encapsulated in a media file (Chapter 8).

Table 5.6 Selected NAL unit types

NAL Unit	Description	VCL
Coded slice, non-IDR	A typical coded slice, see section 5.6	Yes
Coded slice data partition A	Part of a data partitioned slice, see Chapter 8	Yes
Coded slice data partition B	Part of a data partitioned slice, see Chapter 8	Yes
Coded slice data partition C	Part of a data partitioned slice, see Chapter 8	Yes
Coded slice, IDR	Part of the Access Unit at the start of a coded video sequence	Yes
SEI	Supplemental Enhancement Information, see Chapter 8	No
SPS	Sequence Parameter Set, one per sequence, see section 5.5	No
PPS	Picture Parameter Set, see section 5.5	No
Access unit delimiter	Indicates type of the slices in the next coded picture (optional)	No
End of sequence	Indicates that the next access unit is IDR, i.e. the start of a new coded video sequence (optional)	No
End of stream	Indicates the end of the coded bitstream (optional)	No
Filler	Filler bytes (optional), e.g. to prevent buffer under-run, see Chapter 8.	No
.....		

5.5 Parameter Sets

Parameter Sets are NAL units that carry decoding parameters common to a number of coded slices. Sending these parameters independently of coded slices can improve efficiency, since common parameters need only be sent once. Parameter Sets are essential to the correct decoding of a sequence. In a lossy transmission scenario, where bits or packets may be lost or corrupted during transmission, Parameter Sets may be sent with a higher quality of service using e.g. Forward Error Correction or a priority mechanism.

A Sequence Parameter Set (SPS) contains parameters common to an entire video sequence or programme, such as the Profile and Level (Chapter 8), the size of a video frame and certain decoder constraints such as the maximum number of reference frames (section 5.3.3). Table 5.7 shows an example of a SPS for a Baseline Profile, QCIF coded sequence. Note that each SPS has a unique identifier, *seq_parameter_set_id*, which is 0 in this case.

A Picture Parameter Set (PPS) contains common parameters that may apply to a sequence or subset of coded frames, such as entropy coding type, number of active reference pictures and initialization parameters. See Table 5.8 for an example. The PPS has its own identifier, *pic_parameter_set_id*, and 'points' to a SPS identifier. In this example, the PPS 'inherits' the parameters of SPS 0. Each coded slice 'points' to a PPS and inherits its parameters. In this way, there is considerable flexibility to set up common coding parameters via SPS and PPS NAL units and then to refer to these in subsequent coded slices.

Table 5.7 Sequence Parameter Set example

Parameter	Binary code	Symbol	Discussion
profile_idc	1000010	66	Baseline Profile
constrained_set0_flag	0	0	Bistream might not obey all the constraints of the Baseline Profile
constrained_set1_flag	0	0	As above, Main Profile
constrained_set2_flag	0	0	As above, Extended Profile
constrained_set3_flag	0	0	Used to specify the special case of Level 1b
reserved_zero_4bits	0	0	Not used
level_idc	11110	30	Level 3
seq_parameter_set_id	1	0	Sequence Parameter Set 0
log2_max_frame_num_minus4	1	0	frame_num will not exceed 16.
pic_order_cnt_type	1	0	Default POC
log2_max_pic_order_cnt_lsb_minus4	1	0	LSB of POC will not exceed 16.
num_ref_frames	1011	10	Up to 10 reference frames.
gaps_in_frame_num_value_allowed_flag	0	0	No gaps in frame_num.
pic_width_in_mbs_minus1	1011	10	11 macroblocks wide = QCIF
pic_height_in_map_units_minus1	1001	8	9 MBs high = QCIF
frame_mbs_only_flag	1	1	No field slices or field MBs
direct_8 × 8_inference_flag	1	1	Specifies how certain B macroblock motion vectors are derived
frame_cropping_flag	0	0	Frames are not cropped
vui_parameters_present_flag	0	0	VUI parameters not present

Table 5.8 Picture Parameter Set example

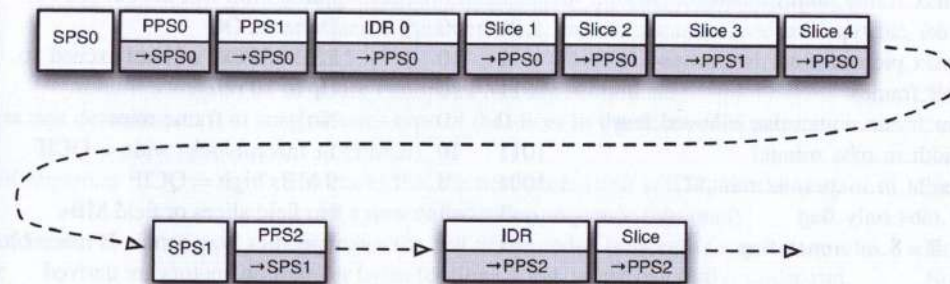
Parameter	Binary code	Symbol	Discussion
pic_parameter_set_id	1	0	Picture Parameter Set 0
seq_parameter_set_id	1	0	Use SPS 0
entropy_coding_mode_flag	0	0	CAVLC entropy coding
pic_order_present_flag	0	0	POC not present
num_slice_groups_minus1	1	0	One slice group
num_ref_idx_l0_active_minus1	1010	9	10 reference pictures in list0
num_ref_idx_l1_active_minus1	1010	9	10 reference pictures in list1
weighted_pred_flag	0	0	Weighted prediction not used
weighted_bipred_idc	0	0	Weighted biprediction not used
pic_init_qp_minus26	1	0	Initial QP (luma) = 26
pic_init_qs_minus26	1	0	Initial SI/SP QP=26
chroma_qp_index_offset	1	0	No chroma QP offset
deblocking_filter_control_present_flag	0	0	Use default filter parameters
constrained_intra_pred_flag	0	0	Intra prediction is not constrained
redundant_pic_cnt_present_flag	0	0	Redundant picture count parameter is not used

Activation of Parameter Sets

A Parameter Set is 'inactive' until it is activated, i.e. its parameters are not used by the decoder. A PPS, previously transmitted to the decoder, is activated when it is referred to in a slice header and remains active until a different PPS is activated. A SPS is activated when a PPS that refers to it, e.g. the PPS in Table 5.8, is activated. A single SPS remains active for an entire coded video sequence, which must start with an IDR Access Unit, hence a SPS is effectively activated by an IDR slice. Each coded slice in a video sequence may refer to one of a number of Picture Parameter Sets.

Using parameter sets: Example

SPS0 is sent at the start of a sequence, followed by PPS0 and PPS1, both of which 'inherit' SPS0 (Figure 5.12). PPS0 is activated by IDR slice 0, which means that SPS0 becomes active at the same time. Slices 1 and 2 use the parameters of PPS0 and SPS0. PPS1 is activated by slice 3, making PPS0 inactive, hence slice 3 uses the parameters of PPS1 and SPS0; PPS0 is activated again by slice 4, making PPS1 inactive.

**Figure 5.12** Example: Sequence and Picture Parameter Sets

At a later point, SPS1 and PPS2 are sent; PPS2 inherits SPS1. Both SPS1 and PPS2 are inactive until a new IDR slice is sent, activating both SPS1 and PPS2 and signalling the start of a new coded video sequence. Subsequent slices use the parameters of SPS1 and PPS2.

5.6 Slice layer

5.6.1 Slice types

Each coded picture, a coded frame or field, is composed of one or more slices, each containing a **slice header** followed an integral number of macroblocks. The number of macroblocks in a slice need not be constant. There are minimal data inter-dependencies between coded slices, which can help to limit the propagation of errors between slices. Possible scenarios for choosing slice sizes include:

- One slice per coded picture, common practice in many H.264 encoding applications.
- N slices per picture, each containing M macroblocks, N and M integer. The number of bytes in each coded slice will tend to vary depending on the amount of motion and detail in the picture area.
- N slices per picture, containing a varying number of macroblocks, chosen to keep the number of bytes per slice roughly constant. This may be useful if, for example, each slice is mapped to a fixed-size network packet.

The available slice types are listed in Table 5.9, together with the macroblock types that may be present in each slice. Note that a B slice (for example) may contain B, P and/or I macroblock types and an encoder may choose between these types depending on the content of the sequence. Macroblock types are explained further in section 5.7.1.

5.6.2 Slice header

The slice header conveys information common to all macroblocks in the slice, such as the slice type which determines which macroblock types are allowed, the frame number that the slice corresponds to, reference picture settings and default quantization parameter (QP). Table 5.10

Table 5.9 Slice types in H.264

Slice type	Contains macroblock types	Notes
I (including IDR)	I only	Intra prediction only.
P	I and/or P	Intra prediction (I) and/or prediction from one reference per macroblock partition (P).
B	I, P and/or B	Intra prediction (I), prediction from one reference (P) or biprediction, i.e. prediction from two references (B)
SP	P and/or I	Switching P slice, see Chapter 8
SI	SI	Switching I slice, see Chapter 8

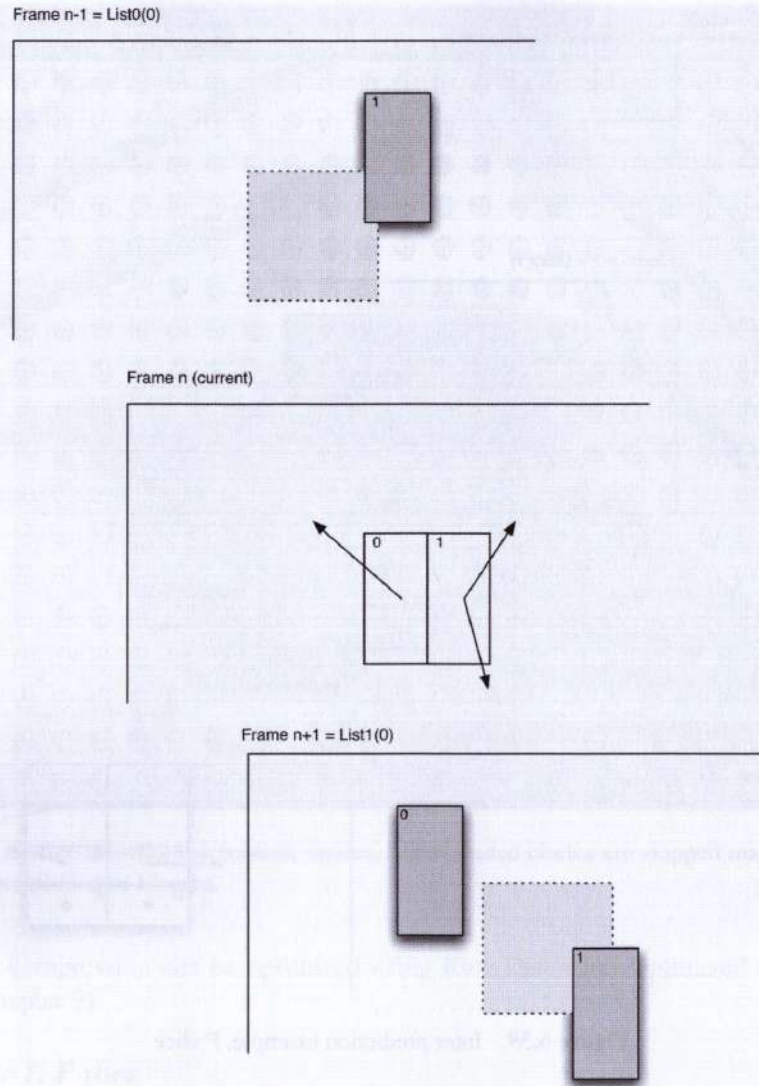


Figure 6.40 Inter prediction example, B slice

Partition 0

Prediction reference: List1(0), which points to frame $n + 1$, the next frame in display order.

Motion vector: $(-10, -9.5)$

Partition 1

Prediction references: List0(0), which points to frame $n - 1$, the previous frame in display order, and List1(0), which points to frame $n + 1$.

Motion vectors: $(+7.25, -9)$ (List0) and $(+6, +8.25)$ (List1).

Hence Partition 0 is predicted from a corresponding area in frame $(n + 1)$ whereas Partition 1 is bipredicted or interpolated between areas in frame $(n - 1)$ and frame $(n + 1)$.

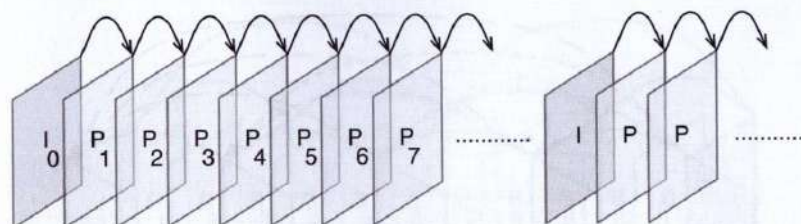


Figure 6.41 Low delay prediction structure

6.4.7 Prediction structures

H.264 offers many different options for choosing reference pictures for inter prediction. An encoder will typically use reference pictures in a structured way. Some examples of reference picture structures are presented here.

6.4.7.1 Low delay, minimal storage

The prediction structure shown in Figure 6.41 uses only I and P slices. It is compatible with the Baseline Profile or Constrained Baseline Profile of H.264, which do not allow B slices, and would be suitable for an application requiring low delay and/or minimal storage memory at the decoder. The first frame is coded as an I slice and subsequent frames are coded as P slices, predicted from the previous frame. Prediction efficiency is relatively low, because only one prediction direction and one reference is allowed for each frame. However, the decoder can display each frame as soon as it is decoded, minimising delay, and the decoder only needs to store one reference frame, minimising storage requirements. This type of structure would be suitable for applications such as videoconferencing where latency must be kept to a minimum. Note that I slices may be inserted in the stream at intervals to limit the propagation of transmission errors and to enable random access to the coded sequence.

6.4.7.2 'Classic' Group of Pictures structure

Earlier standards such as MPEG-1 and MPEG-2 supported the Group of Pictures structure shown in Figure 6.42. The Group of Pictures (GOP) starts with an I slice. P slices are inserted at intervals, with B slices between the I and P slices. The I and P slices are used for reference;

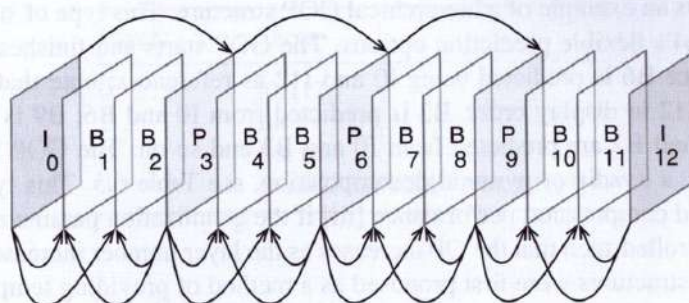


Figure 6.42 'Classic' Group of Pictures prediction structure

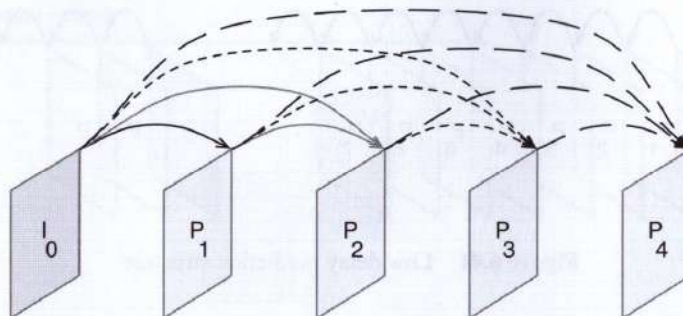


Figure 6.43 IPPP ... with multiple reference pictures

the B slices are not used for reference, i.e. all pictures are predicted from I or P slices. Each P slice is predicted from the preceding I or P slice and each B slice is predicted from the I and/or P slices on either side of it.

This structure provides higher compression efficiency than the structure shown in Figure 6.41 because prediction in a B slice tends to be more efficient than in a P slice due to the availability of two reference pictures. The disadvantage is increased delay and larger frame storage requirements. The decoder must receive slices I0 and P3 before slices 1 and 2 can be decoded, i.e. the delay is a minimum of three frames in this example. Slices 0 and 3 must be stored by the decoder because they are reference frames for slices 1 and 2.

6.4.7.3 Multiple reference frames

H.264 makes it possible to create other prediction structures, for example using multiple reference frames for prediction. Figure 6.43 shows an IPPP ... prediction structure in which all the previously coded slices are available as reference frames. Slice P1 is predicted from slice I0; slice P2 is predicted from slices I0 and P1; slice P3 is predicted from slices I0, P1 and P2; and so on. This means that the encoder can search up to N reference frames to find the best match for each P macroblock. This can improve compression efficiency at the expense of (a) increased computational expense at the encoder and (b) increased storage at encoder and decoder, since N reference frames must be stored.

6.4.7.4 Hierarchical prediction structures

Figure 6.44 shows an example of a hierarchical GOP structure. This type of structure is made possible by H.264's flexible prediction options. The GOP starts and finishes with I slices 0 and 12. Next, slice B6 is predicted using I0 and I12 as references; note that B6 is halfway between I0 and I12 in display order. B3 is predicted from I0 and B6; B9 is predicted from B6 and I12. B1 and B2 are predicted from I0 and B3 and so on. The GOP is composed of a series of layers, a dyadic or pyramid decomposition, see Table 6.5. This type of structure can give improved compression performance [iii] if the quantization parameters of the layers are carefully controlled such that the QP increases as the layer number increases. Hierarchical or pyramid GOP structures were first proposed as a method of providing temporal scalability (Chapter 10) but they may be used in any H.264 codec that supports B slices.

Chapter 9 compares the compression performance of several prediction structures.

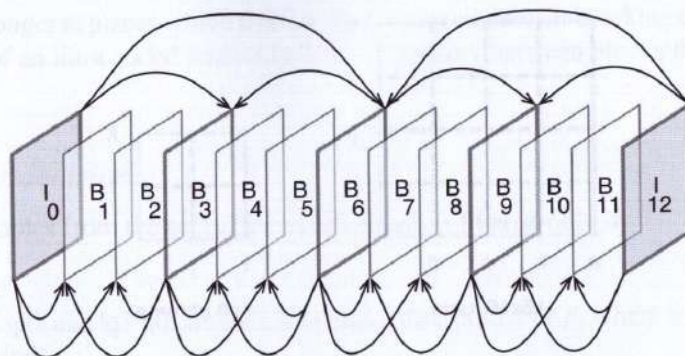


Figure 6.44 Hierarchical GOP structure

6.5 Loop filter

A filter is applied to every decoded macroblock to reduce blocking distortion [iv]. The deblocking filter is applied after the inverse transform in the encoder before reconstructing and storing the macroblock for future predictions and in the decoder before reconstructing and displaying the macroblock. The filter smooths block edges, improving the appearance of decoded frames. The filtered image is used for motion-compensated of future frames and this generally improves compression performance because the filtered image is a more faithful reproduction of the original frame than a blocky, unfiltered image.¹

Filtering is applied to vertical or horizontal edges of 4×4 blocks in a macroblock excluding edges on slice boundaries, in the following order:

1. Filter 4 vertical boundaries of the luma component in order a,b,c,d in Figure 6.45
2. Filter 4 horizontal boundaries of the luma component in order e,f,g,h, Figure 6.45
3. Filter 2 vertical boundaries of each chroma component (i,j)
4. Filter 2 horizontal boundaries of each chroma component (k,l)

Each filtering operation affects up to **three** pixels on either side of the boundary. Figure 6.46 shows four pixels on either side of a vertical or horizontal boundary in adjacent blocks p and q, p0,p1,p2,p3 and q0,q1,q2,q3. The 'strength' of the filter, i.e. the amount

Table 6.5 Layers in hierarchical GOP example

Layer	Slices
0	I0, I12
1	B6
2	B3, B9
3	B1, B2, B4, B5, B7, B8, B10, B11

¹ Intra-coded macroblocks are filtered, but intra prediction (section 6.3) is carried out using **unfiltered** reconstructed macroblocks to form the prediction.

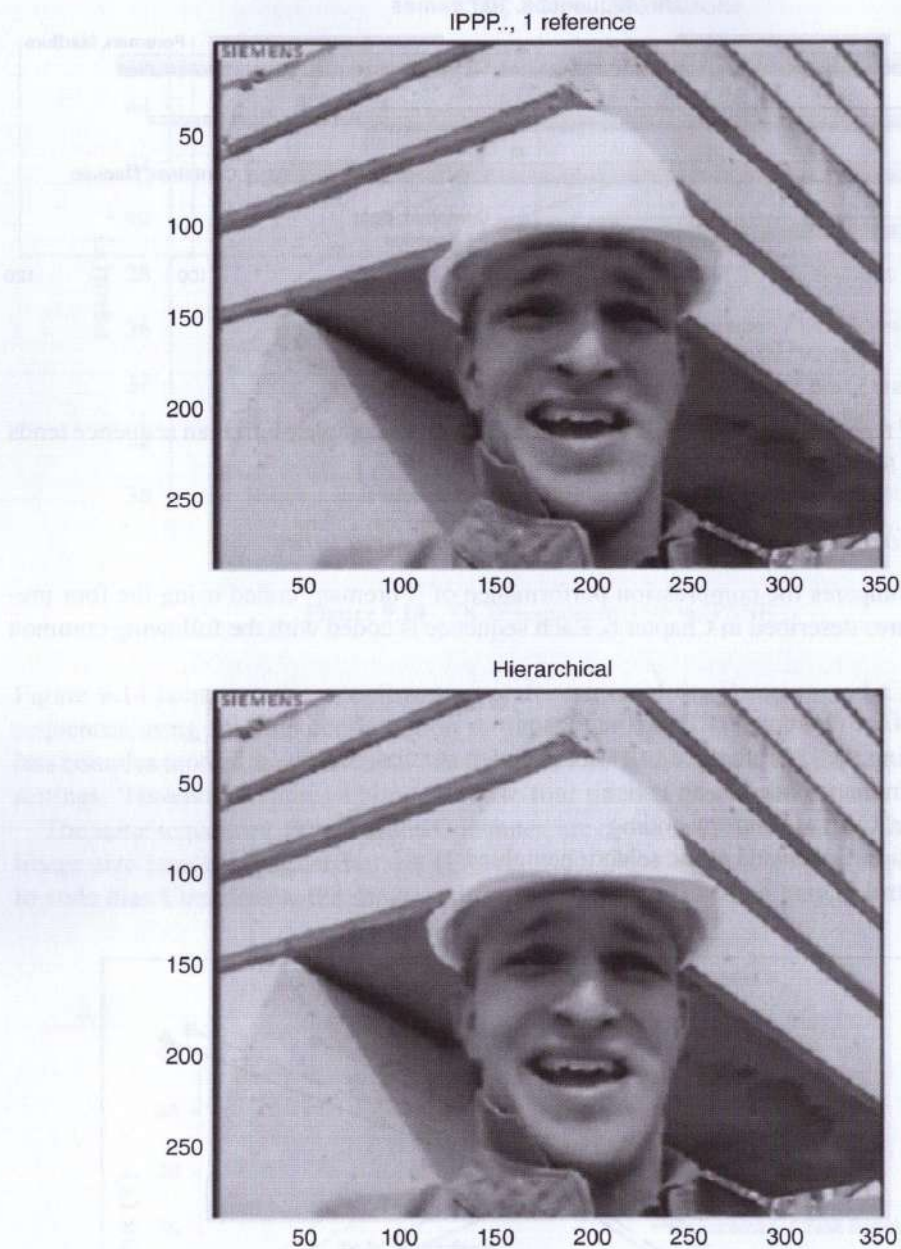


Figure 9.18 Sample frames from sequences using different prediction structures, coded at 280kbps

The prediction structures are as follows (see Chapter 6):

- (i) IPPPPPPPPPPPPPPPP..., i.e. one I-slice followed by 11 P-slices, with one reference frame
- (ii) As (i) but with five reference frames

- (iii) IBBPBBPBBPBBIBBP..., i.e. 12-frame GOP with one I-slice, three P-slices and eight B-slices in each GOP.
- (iv) IBBBBBBBBBBBIBB..., 12-frame GOP, hierarchical prediction.

The basic IPPPP... structure with one reference frame has the worst performance, i.e. the lowest rate-distortion curve. Enabling five reference frames with the same structure improves the performance slightly. The 'Classic' IBBPBBP... GOP structure, with 12 pictures in a GOP, improves the compression performance further and a Hierarchical GOP structure, with 12 pictures, gives the best performance. Sample frames from IPPP... and Hierarchical structures are shown in Figure 9.18, each coded at the same bitrate (280kbps). The luminance PSNR of the Hierarchical sequence is around 0.8dB higher than that of the IPPP... sequence; note that there is only a slight subjective difference in the frames. The Hierarchical sequence takes approximately 50 per cent longer to encode using the JM software.

9.3.5 Performance example: H.264 vs. MPEG-4 Visual

Figure 9.19 compares the rate-distortion performance of H.264 and the earlier MPEG-4 Part 2 ('Visual') standard. Similarly to H.264, MPEG-4 Visual has Profiles that define subsets of coding tools. Results for the 'Carphone' QCIF sequence coded using Simple Profile (SP) and Advanced Simple Profile (ASP) are shown. The SP encoder uses I- and P-frame coding and the ASP encoder adds B-frame coding for better performance. An H.264 Baseline Profile encoder using UVLC/CAVLC and one reference frame performs considerably better than MPEG-4 Visual ASP and H.264 Main Profile with CABAC and five reference frames performs better

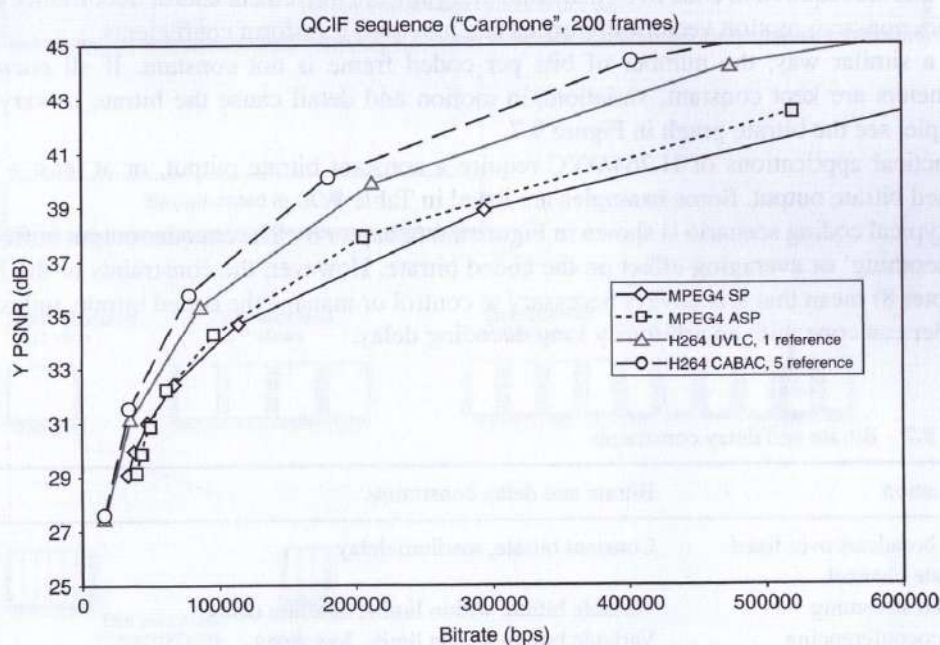


Figure 9.19 Carphone, QCIF: H.264 vs. MPEG-4 Visual

Good results can be obtained by calculating λ as a function of QP (9.4, from [iv]).

$$\lambda = 0.852^{(QP-12)/3} \quad (9.4)$$

Distortion (D) is calculated as the Sum of Squared Distortion (SSD, (9.5):

$$D_{SSD} = \sum_{x,y} (b(x,y) - b'(x,y))^2 \quad (9.5)$$

Where x,y are the sample positions in a block, $b(x,y)$ are the original sample values and $b'(x,y)$ are the decoded sample values at each sample position. Other distortion metrics, such as Sum of Absolute Differences or Sum of Absolute Transformed Differences, SAD and SATD, may be used in processes such as selecting the best motion vector for a block [iv]. A different distortion metric typically requires a different λ calculation.

A typical mode selection algorithm might proceed as follows:

- For every macroblock
 - For every available coding mode m
 - Code the macroblock using mode m and calculate R , the number of bits required to code the macroblock
 - Reconstruct the macroblock and calculate D , the distortion between the original and decoded macroblocks
 - Calculate the mode cost J_m using (9.3), with appropriate choice of λ
 - Choose the mode that gives the minimum J_m

This is clearly a computationally intensive process, since there are hundreds of possible mode combinations and therefore it is necessary to code the macroblock hundreds of times to find the 'best' mode in a rate-distortion sense. The problem becomes larger when considering the choice of prediction, for example:

1. Each 4×4 block in an Intra- 4×4 macroblock can be coded using one of nine modes. The choice of mode for one 4×4 block affects the coding cost and therefore the choice of mode for all the other 4×4 blocks (see Chapter 6, Intra mode signalling).
2. Each partition in an Inter macroblock may be predicted from any one of the available reference frames or from one or two reference frames in a B macroblock.
3. Each partition or sub-macroblock partition in an Inter macroblock has a separate motion vector or two in the case of a bi-predicted partition in a B macroblock; each motion vector may point to any of hundreds of positions within a defined search window.
4. The choice of motion vector or prediction is further complicated by the fact that more likely predictions, e.g. small motion vectors or intra modes similar to recently-selected modes, generally require fewer bits to code than predictions such as large motion vectors that are less likely to generate the best result.

In the case of a B macroblock, for example, an encoder has a huge potential 'space' of coding options, including all the intra modes, all the inter partition sizes, all the possible motion vectors, all the available reference frames and the choice of one-directional or bi-predicted

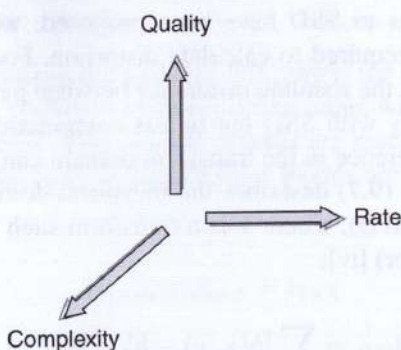


Figure 9.28 Rate, quality and complexity

motion compensation. Exhaustively searching this space to find the best combination of mode and prediction type is a highly computationally intensive task.

9.6 Low complexity coding

Many practical H.264/AVC codecs simply do not have the computational resources to carry out the full rate-distortion optimized mode selection process described above. This practical constraint, together with the desire to maximize compression performance, has led the development and proposal of hundreds of low complexity coding algorithms and approaches. The general goal is to maximize performance in the rate-distortion-complexity space (Figure 9.28). A low-complexity encoder will tend to have poor or average rate-distortion performance; applying more computation to the problem of choosing the best coding mode, hence moving along the complexity axis, will tend to increase rate-distortion performance. Hence compressed bitrate, decoded quality and codec complexity can be traded off. The goal of a good low-complexity coding method is to achieve good rate-distortion performance at a reduced computational cost. A suitable comparison point is a 'full complexity' H.264 encoder, which evaluates every possible coding mode, every prediction type and every motion vector when coding each macroblock.

In general, every low-complexity coding algorithm has a similar effect on performance, i.e. rate-distortion performance tends to be lower than the benchmark 'full complexity' codec and computational complexity is also lower. Some of the more sophisticated low-complexity algorithms, however, are capable of delivering coding performance very close to the benchmark with a significant reduction in computational complexity.

9.6.1 Approximating the cost function

The basic cost function (9.3) requires calculation of D and R . In a 'full' implementation this means it is necessary to:

- (i) Code the block B to obtain R
- (ii) Decode the block to obtain B' , the reconstructed block
- (iii) Calculate the distortion (e.g. SSD) between B and B'

A number of approximations to SSD have been proposed, with the aim of reducing the number of processing steps required to calculate distortion. For example, Sum of Absolute Differences (SAD) calculates the absolute difference between pairs of samples b and b' (9.6). SAD increases monotonically with SSD but is less computationally intensive to calculate. Calculating the absolute difference in the transform domain can improve the accuracy of the cost function approximation. (9.7) describes the transform-domain metric Sum of Absolute Transformed Differences (SATD), where T is a transform such as the Hadamard Transform and α is a normalization factor) [iv].

$$D_{SAD} = \sum_{x,y} |b(x, y) - b'(x, y)| \quad (9.6)$$

$$D_{SATD} = \alpha \sum_{x,y} |T(b(x, y) - b'(x, y))| \quad (9.7)$$

Further fast approximations to these and other distortion metrics have been proposed in the literature, e.g. [xiii, xiv]. An alternative is to sub-sample the block, i.e. reduce the number of sample positions (x, y) to be evaluated [xvi, xv].

A simple approach to approximating the rate of a coded block is to predict blocks with zero rate, i.e. all-zero coefficients [xvii]. As all-zero blocks occur very frequently in a typical coded sequence, predicting the occurrence of these blocks without actually coding the data can save a significant amount of computation. A more sophisticated rate estimation model is presented in [xviii].

Finally, the entire cost function (9.3) may be estimated, for example based on the R-D cost of the same macroblock position in a previously coded frame [xix, xx].

9.6.2 Reducing the set of tested modes

A second approach to reducing the complexity of mode selection is to cut the number of modes that are tested for a given macroblock or block.

The 'Skip' mode tends to occur very frequently in P and B slices, especially when (i) the scene activity is relatively low and/or (ii) the quantization parameter is relatively high. Many methods have been developed that incorporate early skip detection based on modelling previous macroblock statistics, for example [xix, xxi, xx].

More generally, a number of algorithms attempt to reduce the cost of inter mode selection by grouping modes and only evaluating or coding certain groups of inter modes. Groupings can be determined by the statistics of previously coded macroblocks and/or according to homogeneity or features of the current macroblock [xxii, xxiii].

The number of intra modes tested for a MB / block may be reduced by examining the structure of the image data. The best intra mode tends to depend on the characteristics of the block or macroblock. For example, a macroblock containing smooth texture is likely to be effectively predicted using an Intra-16 × 16 mode (see Chapter 6) [xxiv]. Alternatively, the best Intra-4 × 4 mode for a particular 4 × 4 block is likely to be closely correlated to the block texture. For example, the dominant edge direction in a 4 × 4 block may be used to predict the most likely prediction direction [xxv].

9.6.3 Early termination

Related to the strategy of reducing the number of modes is the concept of early termination, which involves (i) evaluating coding modes in a certain order, which may be fixed or variable, and (ii) terminating the process when certain criteria are reached. For example, a number of algorithms assume that the costs of certain inter modes are monotonically increasing or decreasing, i.e.:

$$J_{16 \times 16} > J_{8 \times 8} > J_{4 \times 4}$$

or

$$J_{16 \times 16} < J_{8 \times 8} < J_{4 \times 4}$$

Evaluating the modes in order of expected monotonicity, the encoder terminates the mode selection process if a particular mode cost is not monotonic [xxvi, xxvii].

9.7 Summary

A good way to understand and evaluate the capabilities of a video coding method such as H.264/AVC is to experiment with it. The availability of public-domain encoders and decoders such as the 'official' Joint Model (JM) and the open-source $\times 264$, makes it possible to test out every aspect of the standard. H.264/AVC has the potential to deliver coding performance as good as, or better than, other standards-based and proprietary codecs that are currently available. However, the performance of an H.264 codec depends very much on the coding parameters and the source video material. There is a fundamental trade-off between good coding performance - high quality and low bitrate - and computational complexity, particularly in a video encoder. A key aspect of this trade-off is the challenge of selecting the best coding mode for a macroblock, out of a very large number of possible options. Practical H.264/AVC encoders typically simplify this process using fast, low-complexity approximations to the full mode selection process.

9.8 References

- i. JM reference software version 16.0, <http://iphone.hhi.de/suehring/tml/>, July 2009.
- ii. ITU-T Recommendation H.264.2 : Reference software for H.264 advanced video coding, June 2008.
- iii. A. M. Tourapis, A. Leontaris, K. Suehring and G. Sullivan, 'H.264/MPEG-4 AVC Reference Software Manual', Joint Video Team Document JVT-AD010, January 2009.
- iv. K. P. Lim, G. Sullivan and T. Wiegand, 'Text Description of Joint Model Reference Encoding Methods and Decoding Concealment Methods', Joint Video Team Document JVT-O079, April 2005.
- v. G. Sullivan, 'Adaptive quantization encoding technique using an equal expected-value rule', Joint Video Team Document JVT-N011, Jan 2005.
- vi. $\times 264$ public domain H.264 encoder/decoder, <http://www.videolan.org/developers/x264.html>.
- vii. Doom9 Codec Shoot-Out 2005, <http://www.doom9.org/index.html/?codecs-final-105-1.htm>, December 2005.
- viii. Fourth Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison, <http://compression.ru/video/codecs-comparison/mpeg-4-avc-h264-2007-en.html>, December 2007.
- ix. Z. Li, F. Pan, K. P. Lim, G. Feng, X. Lin and S. Rahardja, 'Adaptive Basic Unit Layer Rate Control for JVT', Joint Video Team Document JVT-G012, March 2003.

- x. A. Leontaris and A. M. Tourapis, 'Rate Control reorganization in the Joint Model (JM) reference software', Joint Video Team Document JVT-W042, April 2007.
- xi. A. Ortega and K. Ramchandran, 'Rate-distortion methods for image and video compression', *IEEE Signal Processing Magazine*, November 1998.
- xii. G. Sullivan and T. Wiegand, 'Rate-distortion optimization for video compression', *IEEE Signal Processing Magazine*, November 1998.
- xiii. C. Tseng, H. Wang and J. Yang, 'Enhanced intra-4 × 4 mode decision for H.264/AVC coders', *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 8, August 2006, pp. 1027–1032.
- xiv. J. Xin, A. Vetro and H. Sun, 'Efficient macroblock coding-mode decision for H.264/AVC video coding', Proceedings of the Picture Coding Symposium, 2004.
- xv. Y. Ivanov and C. Bleakley, 'Skip prediction and early termination for fast mode decision in H.264/AVC', Proceedings of the International Conference on Digital Telecommunications, 2006.
- xvi. Y. Huang, B. Hsieh, T. Chen and L. Chen, 'Analysis, fast algorithm and VLSI architecture design for H.264/AVC intra frame coder', *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 3, pp. 378–401, March 2005.
- xvii. Y. Moon, G. Kim and J. Kim, 'An improved early detection algorithm for all-zero blocks in H.264 video encoding', *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 8, pp. 1053–1057, August 2005.
- xviii. H. Kim and Y. Altunbasak, 'Low complexity macroblock mode selection for H.264/AVC encoders', Proc. IEEE International Conference on Image Processing, October 2004.
- xix. C. Kannangara, I. Richardson, M. Bystrom, J. Solera, Y. Zhao, A. MacLennan and R. Cooney, 'Low complexity skip prediction for H.264 through Lagrangian cost estimation', *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16 no. 2, pp. 202–208, February 2006.
- xx. M. Bystrom, I. Richardson and Y. Zhao, 'Efficient mode selection for H.264 complexity reduction in a Bayesian framework', *Signal Processing: Image Communication*, vol. 23, no. 2, pp. 71–86, February 2008.
- xxi. B. Jeon and J. Lee, 'Fast mode decision for H.264', Joint Video Team document JVT-J033, ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, Hawaii, December 2003.
- xxii. P. Yin, H. Tourapis, A. Tourapis and J. Boyce, 'Fast mode decision and motion estimation for JVT-H.264', Proc. IEEE International Conference on Image Processing, September 2003.
- xxiii. A.C. Yu, 'Efficient Block-Size Selection Algorithm for Inter-Frame Coding in H.264/MPEG-4 AVC,' Proc. IEEE International Conference on Acoustic, Speech, Signal Processing, May 2004.
- xxiv. C. Yang, L. Po and W. Lam, 'A fast H.264 intra prediction algorithm using macroblock properties', Proc. IEEE International Conference on Image Processing, October 2004.
- xxv. F. Pan, X. Lin, S. Rahardja, K. Lim, Z. Li, D. Wu and S. Wu, 'Fast mode decision algorithm for intra prediction in H.264/AVC video coding', *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 7, pp. 813–822, July 2005.
- xxvi. Z. Zhou, M. Sun, and Y. Hsu, 'Fast Variable Block-Size Motion Estimation Algorithms Based on Merge and Split Procedures for H.264/MPEG-4 AVC,' Proc. International Symposium on Circuits and Systems, 2004.
- xxvii. L. Salgado and M. Nieto, 'Sequence Independent Very Fast Mode Decision Algorithm on H.264/AVC Baseline Profile,' Proc. IEEE International Conference on Image Processing, 2006.

10

Extensions and directions

10.1 Introduction

Since the first version of H.264/AVC was published in 2003, the video coding industry has continued to evolve. The range of platforms and delivery mechanisms for video continues to grow, with an increasing expectation that video content should be available on any platform from mobile to HD and 3D displays, over any network including broadcast, internet, mobile, etc. The standard itself has evolved since 2003. This chapter summarizes recent extensions to the standard and looks at what might come after H.264.

The so-called 'Professional' or 'Fidelity Range' extensions became the High Profiles of H.264, tools for coding High Definition and studio content with very high reproduction fidelity, described in earlier chapters.

The increasing need for coding the same original content at different bandwidths and display resolutions led to the development of the Scalable Video Coding (SVC) extension to H.264, standardised as H.264 SVC. SVC supports efficient coding of video in such a way that multiple versions of the video signal can be decoded at a range of bitrates, spatial resolutions and/or temporal resolutions or frame rates. By jointly coding multiple versions, it should be possible to deliver them in a more efficient way than the alternative of coding and transmitting each version separately.

There is a trend towards creating and delivering multiple views of the same video scene. Stereoscopic video, with suitable display technology, gives the impression of a three-dimensional (3D) image. Multiple views of a scene can give the user the option of choosing their viewpoint. 'Free viewpoint' video can potentially deliver any view of a scene, by synthesising intermediate views between actual camera positions. These 'multiview' applications generally require coding of multiple, closely related video signals or views. Similarly to SVC, Multiview Video Coding (MVC) exploits the correlation between these views to deliver efficient compression. Tools for multiview video coding have been standardized as H.264 MVC.

The number of video compression formats continues to increase, with more and more content being produced and coded into many different, incompatible compression formats. Recent initiatives in configurable video coding address the problem of efficiently supporting an increasing range of compression formats. MPEG's Reconfigurable Video Coding (RVC)

sub-group has defined a Video Tool Library, a set of standard Functional Units, building blocks for video compression. A particular video codec may be specified by defining a subset of Functional Units together with their parameters and interconnections. This should make it possible to flexibly re-configure a video codec to support multiple standard and proprietary formats. Going further, Fully Configurable Video Coding (FCVC) makes it possible to completely define and implement an arbitrary video codec using a set of low-level primitive operations and to change this definition during a video communication session. Potential benefits of this approach include the ability to rapidly implement any new video coding algorithm and to adapt the compression algorithm dynamically to suit the characteristics of the current video scene.

H.264 has proved to be a useful and successful technical standard and continues to increase its share of the video coding market. As processor capabilities continue to develop, the standards community are considering what should come after H.264/AVC. At the time of writing (early 2010), the MPEG and VCEG groups are examining proposals for a next generation video coding standard that is expected to offer better performance than H.264, probably at a higher computational cost.

10.2 Scalable Video Coding

10.2.1 *Simulcast transmission*

A challenge for many video compression applications is to deliver multiple versions of a video sequence at different operational points, i.e. different qualities, spatial resolutions and frame rates. This can be done using conventional video codecs such as H.264/AVC by coding each stream independently. This is **simulcast**. In a typical scenario (Figure 10.1), a single source video is required to be transmitted to multiple decoders or clients, each with different capabilities. In this example, the original video clip is coded three times to produce three independent AVC streams, each of which is transmitted and decoded. The problem with the simulcast scenario is that the three bitstreams contain significant redundancy, since the same video sequence is coded in each bitstream at different resolutions and/or qualities. In theory, a

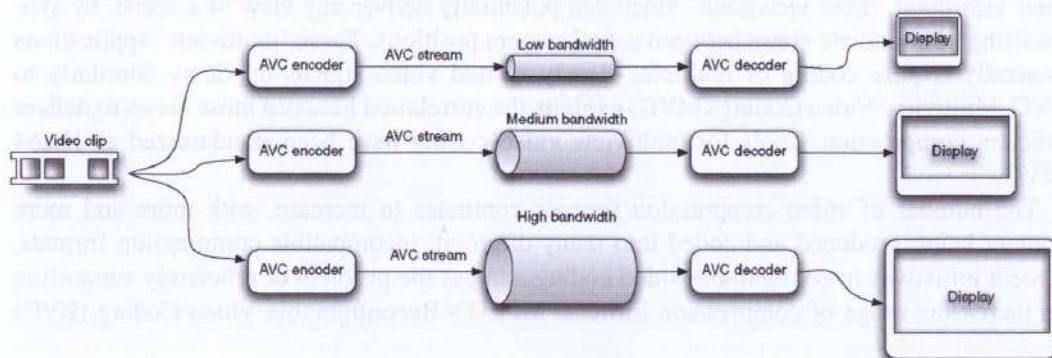


Figure 10.1 Multiple streams / simulcast

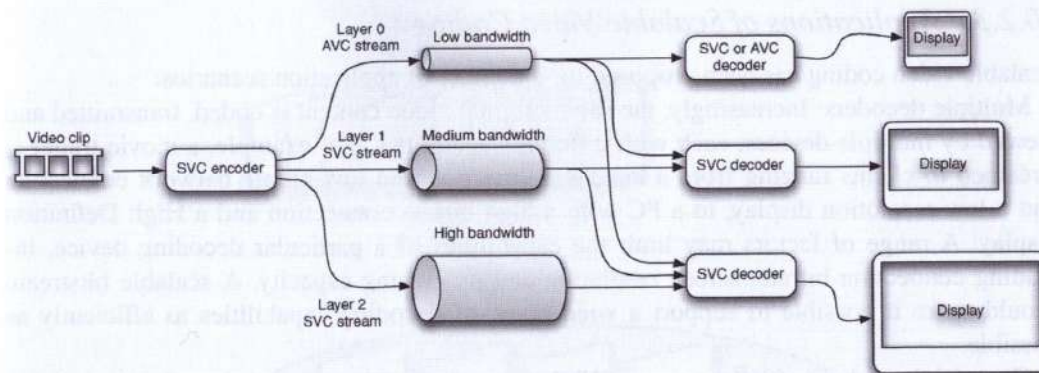


Figure 10.2 Multiple streams / scalable

smaller transmission bandwidth could be utilized by exploiting this redundancy between the three streams.

10.2.2 Scalable transmission

Scalable Video Coding (SVC) attempts to deliver multiple coded versions of a sequence using a lower overall bitrate than the simulcast scenario above. It does this by exploiting the redundancies between the different versions, i.e. the correlation between different versions of the same sequence coded at different operating points.

The same three sequences delivered using SVC are shown in Figure 10.2. A single SVC encoder produces three coded bitstreams, described as **layers**. The lowest or **base layer**, layer 0 in the figure, is a stream decodeable using a standard single-layer decoder, e.g. an H.264 decoder, to produce a video sequence at the lowest of the available quality/resolution operating points. One or more **enhancement layers**, layers 1 and 2 in this example, are coded as SVC bitstreams. To decode a sequence at a higher quality or resolution, an SVC decoder decodes the base layer **and** one or more enhancement layers. In this example, decoding layer 0 using a standard AVC decoder produces the lowest quality output; decoding layers 0 and 1 using an SVC decoder produces a higher quality output; decoding layers 0, 1 and 2 using an SVC decoder produces the highest-quality output. The SVC coding process exploits redundancy between sequences coded at different resolutions or qualities, by predicting successive enhancement layers from the base layer and lower enhancement layers. In this way, it should be possible to achieve the same displayed result as the simulcast system (Figure 10.1) at a reduced bandwidth cost.

The general concept of a scalable coded bitstream is that 'parts of the stream can be removed in such a way that the resulting sub-stream forms another valid bit stream for some target decoder' [i]. Considering Figure 10.2, the scalable bitstream consists of the coded Layer 0, Layer 1 and Layer 2 streams. Decoding all three streams produces a high-quality output; removing Layer 2 and decoding layers 0 and 1 produces a medium-quality output; removing layers 1 and 2 and decoding just the base layer produces a low-quality output.

10.2.3 Applications of Scalable Video Coding

Scalable video coding has been proposed for a number of application scenarios.

Multiple decoders: Increasingly, the same original video content is coded, transmitted and viewed by multiple devices, each with different capabilities. For example, a movie trailer is streamed to clients ranging from a handheld device with a low bitrate network connection and a low-resolution display, to a PC with a high bitrate connection and a High Definition display. A range of factors may limit the capabilities of a particular decoding device, including connection bitrate, screen resolution and processing capacity. A scalable bitstream should make it possible to support a wide range of decoding capabilities as efficiently as possible.

Graceful degradation / enhancement: Whilst some applications such as broadcast television tend to have a clearly-defined and consistent channel for video transmission, many other applications use a channel that may change significantly during a communication session. For example, IP-based applications such as video streaming or internet conferencing will experience a varying channel throughput that depends on factors such as the amount of traffic and congestion in the network. Scalable coding offers a mechanism for maximising the quality at a particular point in time for a specific decoder. For example, a streaming server transmits base and enhancement layers for a video source. A decoder attempts to receive each of the available layers. If all layers are successfully received, the decoder extracts a sequence at the maximum available quality. If the connection throughput drops, the decoder 'drops back' to a lower-quality sequence by only receiving selected layers. As long as the base layer is successfully decoded, a basic-quality video sequence can be displayed at all times. This implies that the base layer is very important, i.e. a higher priority than the enhancement layer(s).

Archiving: Storing a video sequence as a scalable coded bitstream can make it possible to rapidly decode a low-quality 'preview' of the video sequence. For example, a HD sequence is coded as a number of scalable layers. Extracting only the base layer gives a low quality version that is quick to decode and display, suitable as a preview of the entire sequence.

10.2.4 Scalable video coding in H.264

Scalable Video Coding (SVC) is incorporated as Annex G of recent versions of the H.264/AVC standard [i, ii] and extends the capabilities of the original standard. A software implementation, the Joint Scalable Video Model, JSVM, is available for download and experimentation [iii].

H.264 SVC supports three main types or classes of scalability (Figure 10.3):

1. **Temporal scalability:** The base layer is coded at a low temporal resolution or low frame rate; adding enhancement layers increases the frame rate of the decoded sequence.
2. **Spatial scalability:** The base layer is coded at a low spatial resolution; adding enhancement layers increases the spatial resolution of the decoded sequence.
3. **Quality scalability:** The base layer is coded at a low visual quality using a high QP; adding enhancement layers increases the visual quality of the decoded sequence.

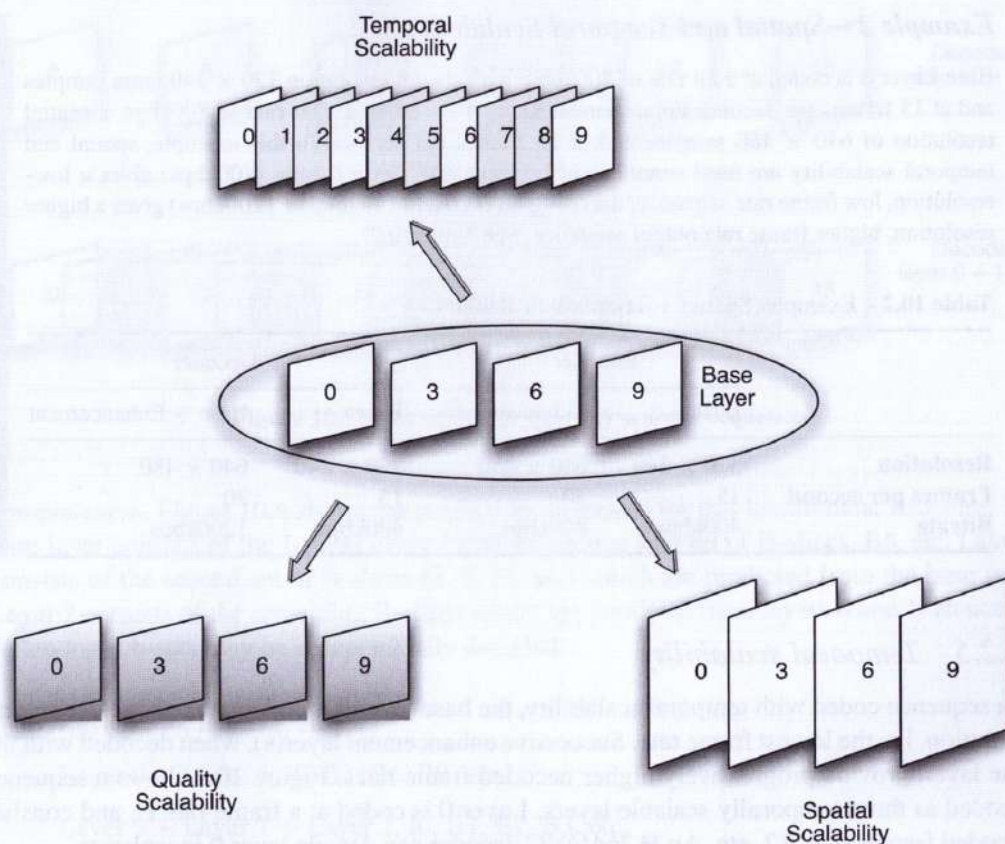


Figure 10.3 Overview of scalability types

Example 1 – Quality Scalability:

Layer 0, base layer, is coded at a bit rate of 200kbps, at CIF resolution, 352×288 luma samples per frame, and at 30 frames per second. Enhancement layer 1 is coded at a bit rate of 520 kbps using quality scalability, i.e. the frame rate and resolution stay the same. A low-quality sequence is obtained by sending the base layer over a 200kbps channel and decoding it. A high-quality sequence is obtained by sending base + enhancement layers over a 720kbps channel. (See Table 10.1).

Table 10.1 Example: Quality Scalability

	Encoder		Decoder	
	Base	Enhancement	Base	Base + Enhancement
Resolution	352×288	352×288	352×288	352×288
Frames per second	30	30	30	30
Bitrate	200kbps	520kbps	200kbps	720kbps

Example 2 – Spatial and Temporal Scalability:

Base Layer 0 is coded at a bit rate of 400kbps, with spatial resolution 320×240 luma samples and at 15 frames per second. Enhancement Layer 1 is coded at a bit rate of 800kbps, a spatial resolution of 640×480 samples and at 30 frames per second. In this example, spatial and temporal scalability are used simultaneously. Decoding Layer 0 only (400kbps) gives a low-resolution, low frame rate sequence; decoding layers 0 and 1 (a total of 1200kbps) gives a higher resolution, higher frame rate output sequence. See Table 10.2.

Table 10.2 Example: Spatial + Temporal Scalability

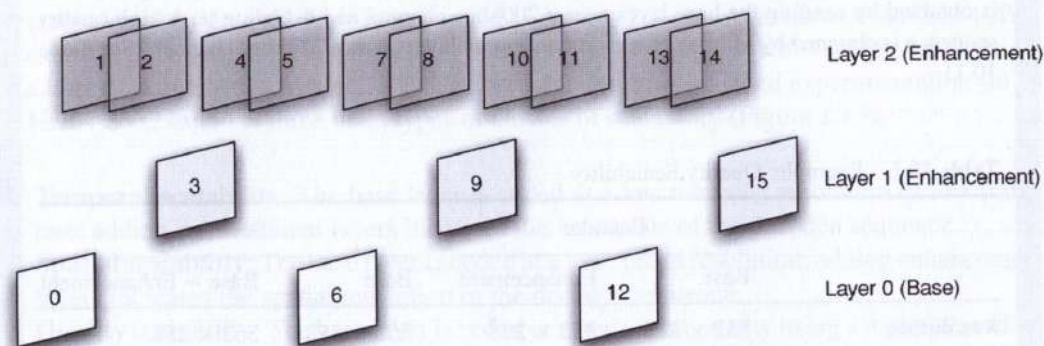
	Encoder		Decoder	
	Base	Enhancement	Base	Base + Enhancement
Resolution	320×240	640×480	320×240	640×480
Frames per second	15	30	15	30
Bitrate	400kbps	800kbps	400kbps	1200kbps

10.2.5 Temporal scalability

In a sequence coded with temporal scalability, the base layer 0 is coded at the lowest temporal resolution, i.e. the lowest frame rate. Successive enhancement layer(s), when decoded with the base layer, provide progressively higher decoded frame rates. Figure 10.4 shows a sequence encoded as three temporally scalable layers. Layer 0 is coded at a frame rate F_0 and consists of coded frames 0, 6, 12, etc. An H.264/AVC decoder can decode layer 0 in isolation.

Layer 1 consists of frames 3, 9, 15, ... etc. A decoder may decode layers 0 and 1 to produce a higher-rate sequence (Figure 10.5) at $2F_0$ frames per second. Layer 2 consists of frames 1, 2, 4, 5, 7, 8, ... etc and a decoder that decodes layers 0, 1 and 2 can deliver an output sequence (Figure 10.5) at $6F_0$ frames per second.

Temporal scalability can be achieved using the P- and/or B-slice coding tools available in H.264/AVC. The examples in Figure 10.4 and Figure 10.5 are developed using the 'Hierarchical' or 'Pyramid' Group of Pictures structure discussed in Chapter 6 (section 6.4.7). For

**Figure 10.4** Temporally scalable sequence, 3 layers

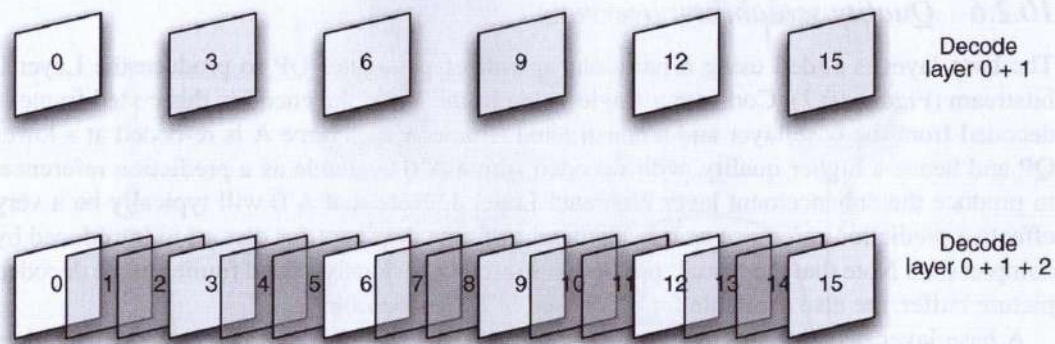


Figure 10.5 Decoding a temporally scalable sequence

completeness, Figure 10.6 shows the prediction directions for this hierarchical structure. The base layer consists of the I-slices every 12 slices and the first set of B-slices, B6, etc. Layer 1 consists of the second set of B-slices (3, 9, 15, etc) which are predicted from the base layer. Layer 2 consists of the remaining B-slices which are predicted from layers 0 and 1. Hence the following sub-sets may be independently decoded:

Layer 0, I0, B6, I12, etc

Layer 0 + Layer 1, I0, B3, B6, B9, etc.

Layer 0 + Layer 1 + Layer 2, I0, B1, B2, B3, etc.

Because the necessary prediction tools, i.e. P or B pictures used for reference, are supported in the Main and High Profiles of H.264/AVC, temporal scalability can be achieved **without** the need for any extensions to the core H.264/AVC standard.

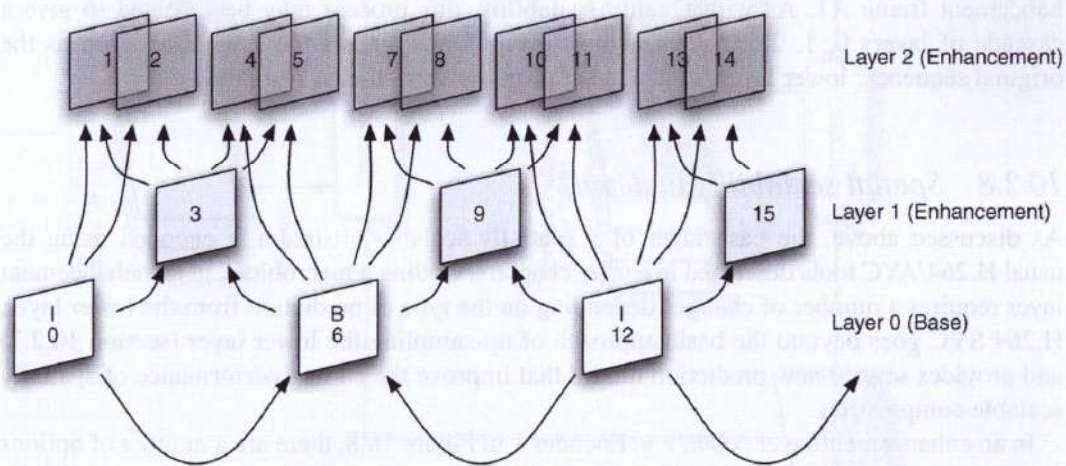


Figure 10.6 Hierarchical prediction structure