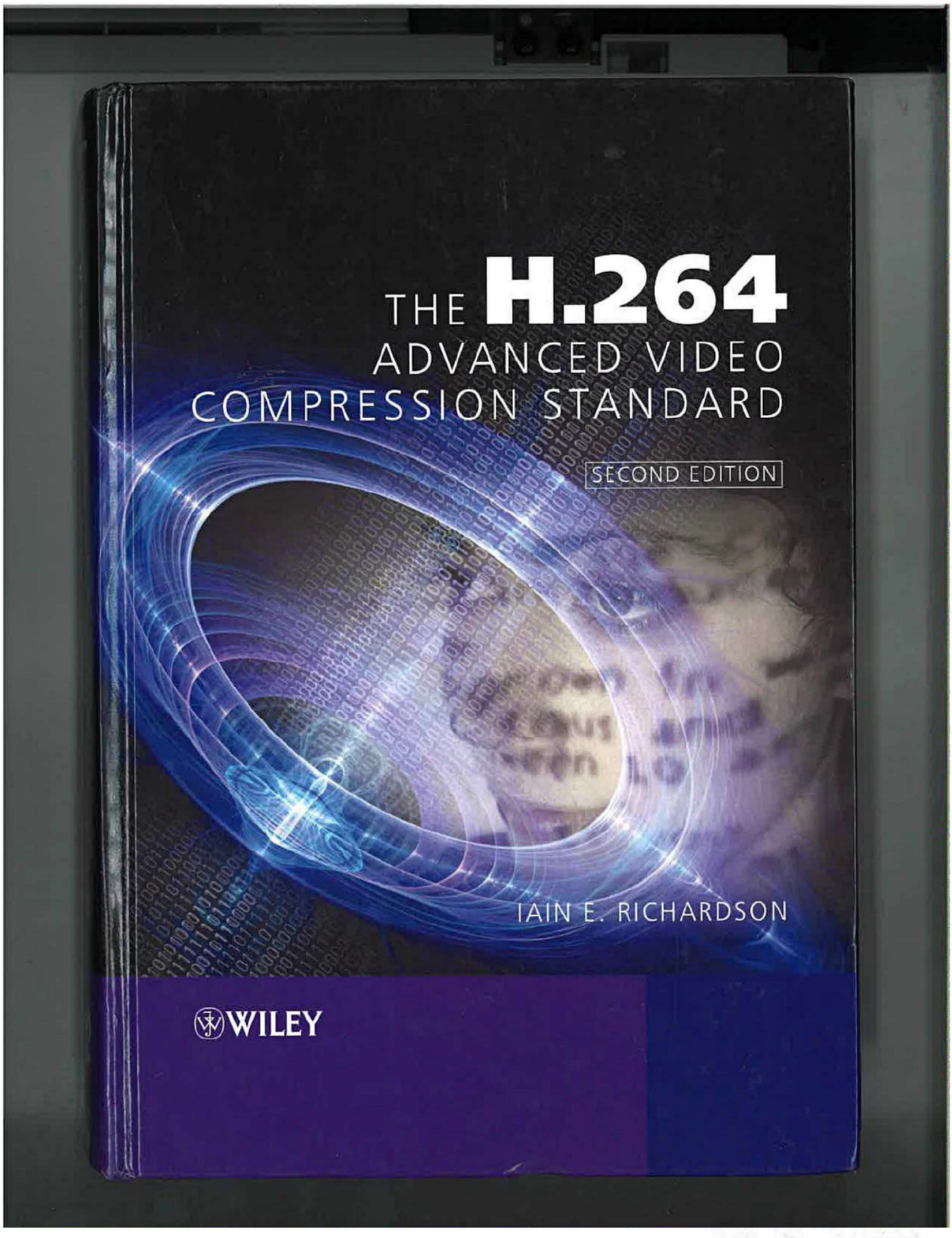
EXHIBIT 2003



IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00001

THE H.264 ADVANCED VIDEO COMPRESSION STANDARD

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00002

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

To Pat

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

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00003

Contents

About the Author xiii

Preface xv

Glossary xvii

List of Figures xxi

List of Tables xxix

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

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00004

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

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00005

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

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00006

About the Author

Professor Iain Richardson is an internationally known expert on the MPEG and H.264 video compression standards.

The author of *H.264 and MPEG-4 Video Compression*, a widely cited work in the research literature, Professor Richardson has written two further books and over 70 journal and conference papers on video compression. He regularly advises companies on video compression technology, video coding patents and company acquisitions in the video coding industry. Professor Richardson leads an internationally renowned video coding research team, contributes to the MPEG industry standards group and is sought after as an expert witness. Based in Aberdeen, Scotland, he regularly travels to the US and Europe.

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00007

The improved compression performance of H.264 comes at the price of greater computational cost. H.264 is more sophisticated than earlier compression methods and this means that it can take significantly more processing power to compress and decompress H.264 video.

4.7.2 *Applications*

As well as its improved compression performance, H.264 offers greater flexibility in terms of compression options and transmission support. An H.264 encoder can select from a wide variety of compression tools, making it suitable for applications ranging from low-bitrate, low-delay mobile transmission through high definition consumer TV to professional television production. The standard provides integrated support for transmission or storage, including a packetized compressed format and features that help to minimize the effect of transmission errors.

H.264/AVC has been adopted for an increasing range of applications, including:

- High Definition DVDs (Blu-Ray)
- High Definition TV broadcasting in Europe
- Apple products including iTunes video downloads, iPod video and MacOS
- NATO and US DoD video applications
- Mobile TV broadcasting
- Many mobile video services
- Many internet video services
- Videoconferencing
- Consumer camcorders.

4.8 Summary

H.264 Advanced Video Compression is an industry standard for video coding that defines a format or syntax for compressed video and a method of decoding this syntax. It provides a set of tools or algorithms that can be used to deliver efficient, flexible and robust video compression for a wide range of applications, from low-complexity, low bitrate mobile video applications to high-definition broadcast services. The following chapters examine H.264's tools, features and performance in detail.

4.9 References

i. Recommendation ITU-T H.264 | ISO/IEC 14496-10:2009, 'Advanced Video Coding for generic audio-visual services', March 2009.

ii. ISO/IEC 14496 2:2000, 'Information technology – Generic coding of moving pictures and associated audio information: Video', 2000.

5

H.264 syntax

5.1 Introduction

An 'H.264 video' is a video sequence that is represented in a particular format, the H.264/AVC syntax. This syntax is defined in the H.264 industry standard and specifies the exact structure of an H.264-compliant video sequence, in terms of syntax elements, parameters describing different aspects of the coded sequence, and the way in which each element is represented as a binary code. The syntax is hierarchical, from the highest level, the video sequence level, down through individual coded frames or fields (access units) and subsets of access units (slices) to macroblocks and blocks. Control parameters are stored as separate syntax sections such as Parameter Sets or as part of sections such as the macroblock layer. This chapter examines the H.264/AVC syntax in detail, beginning with an overview of frame and picture structures and handling, followed by each of the main layers of the syntax hierarchy.

5.1.1 *A note about syntax examples*

The various syntax sections in the H.264/AVC standard are illustrated later in the chapter with extracts from coded video sequences. These were generated using the Joint Model software video codec [i] using the 'Trace' mode which is described further in Chapter 9. Each syntax example is presented in the following way, Table 5.1:

Table 5.1 Syntax examples: format

Parameter	Binary code	Symbol	Discussion
profile_idc	*1000010*	*66*	*Baseline Profile*
↑ The name of the syntax parameter, listed in the H.264/AVC standard	↑ The binary code	↑ The value decoded from this code	↑ What the value indicates.

Selected sections of the H.264 syntax are presented as syntax diagrams such as Figure 5.13.

The H.264 Advanced Video Compression Standard, Second Edition Iain E. Richardson
© 2010 John Wiley & Sons, Ltd

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00008

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

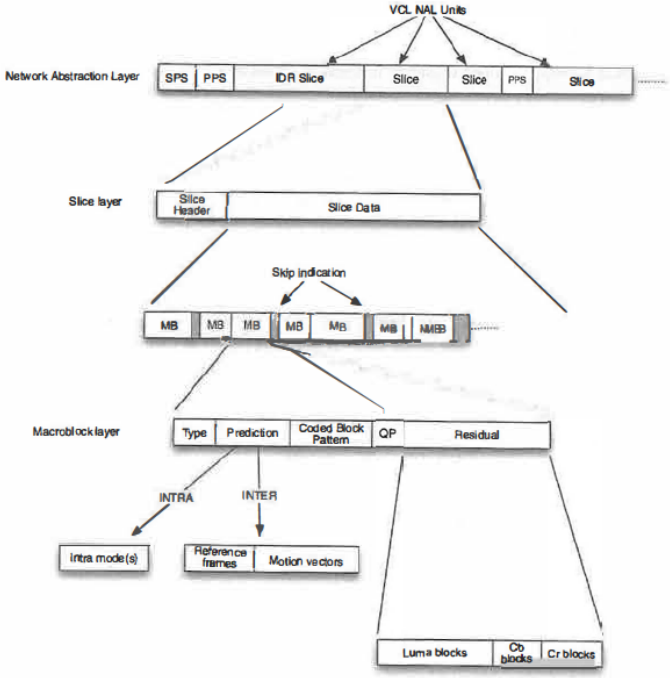


Figure 5.1 Syntax overview

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).

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00009

Table 5.2 H.264 Syntax Sections

Syntax section	Description	Contains section(s)
NAL unit	Network abstraction layer unit.	Raw Byte Sequence Payload (RBSP)
Sequence Parameter Set (RBSP)	Parameters common to a video sequence.	Scaling List (optional) VUI Parameters (optional) RBSP trailing bits
Scaling List	Encoder-specified scaling matrix for inverse quantization.	None
SPS extension (RBSP)	Auxiliary picture information for alpha blending.	RBSP trailing bits
Picture Parameter Set (RBSP)	Parameters common to one or more coded pictures.	Scaling List (optional) RBSP trailing bits
Supplement Enhancement Information (RBSP)	Container for SEI message, see below.	SEI message RBSP trailing bits
Supplemental Enhancement Information Message	SEI messages contain information that may assist decoding or display but is not essential for constructing decoded video frames.	SEI payload
Access Unit Delimiter (RBSP)	Optional delimiter that indicates the slice type in the next coded picure.	RBSP trailing bits
End of Sequence (RBSP)	Optional delimiter that indicates the next slice is an IDR.	None
End of Stream (RBSP)	Optional delimiter that indicates the end of the coded bitstream.	None
Filler Data (RBSP)	Optional series of filler bytes.	RBSP trailing bits
Slice layer without partitioning (RBSP)	Coded slice that does not use data partitioning.	Slice header Slice data RBSP slice trailing bits
Slice data partition A layer (RBSP)	Partition A of a Data Partitioned Slice, see Chapter 8.	Slice header Slice data RBSP slice trailing bits
Slice data partition B layer (RBSP)	Partition B of a Data Partitioned Slice, see Chapter 8.	Slice header Slice data RBSP slice trailing bits
Slice data partition C layer (RBSP)	Partition C of a Data Partitioned Slice, see Chapter 8.	Slice header Slice data RBSP slice trailing bits
RBSP slice trailing bits	Contains RBSP trailing bits and occurs at end of coded slice or slice data partition.	RBSP trailing bits
RBSP trailing bits	Padding bits to ensure RBSP is byte aligned.	None
Slice header	Parameters common to a coded slice.	Reference picture list reordering Prediction weight table (optional) Decoded reference picture marking (optional)

Table 5.2 *(Continued)*

Syntax section	Description	Contains section(s)
Reference picture list reordering	Series of commands to change default reference picture list order(s).	None
Prediction weight table	Luma and chroma weight offsets to change effect of motion compensated prediction.	None
Decoded reference picture marking	Series of commands to mark reference pictures as long term references, etc.	None
Slice data	Contains series of coded macroblocks.	Macroblock layer
Macroblock layer	PCM samples or macroblock headers, prediction and transform coefficients.	Macroblock prediction (optional) Sub-macroblock prediction (optional) Residual data (optional)
Macroblock prediction	Intra prediction modes or reference indices and motion vectors.	None
Sub-macroblock prediction	Reference indices and motion vectors.	None
Residual data	Contains series of residual blocks, as indicated by Coded Block Pattern.	Residual block CAVLC or residual block CABAC
Residual block CAVLC	Block of transform coefficients coded using CAVLC.	None
Residual block CABAC	Block of transform coefficients coded using CABAC.	None

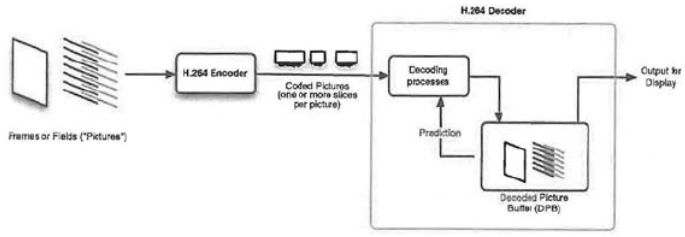


Figure 5.2 Picture handling in H.264, overview

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00010

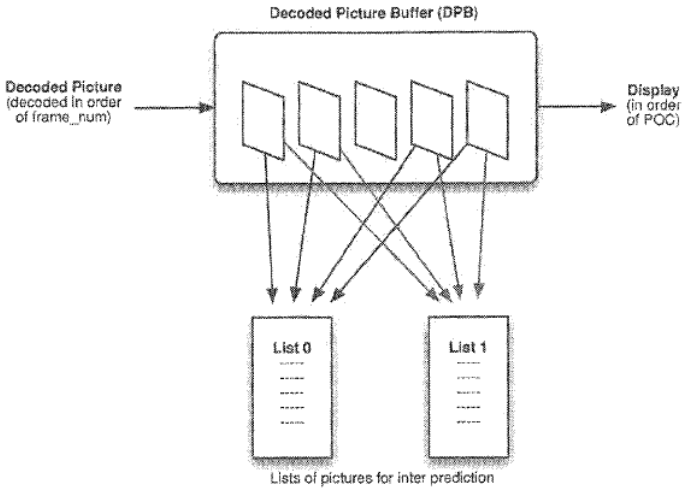


Figure 5.3 Decoded Picture Buffer and picture orders

As shown in Figure 5.3, the **decoding order** of pictures, i.e. the order in which a video decoder should process the coded pictures, is indicated by the parameter *frame_num*.

The **display order** of pictures is determined by the parameters *TopFieldOrderCount* and *BottomFieldOrderCount*, collectively described as Picture Order Count, POC.

The **reference order** of pictures is determined by one or two lists, each of which is an ordered list of all the available decoded pictures. A P slice uses a single list, list0, and a B slice uses two lists, list0 and list1, each containing available reference pictures in different orders.

5.3.1 *Decoding order*

The parameter *frame_num*, decoded from the slice header, determines the decoding order of pictures, coded frames or fields. Except in certain special cases, *frame_num* for each decoded picture increases by one compared with the previous **reference** frame in decoding order.

5.3.2 *Display order*

Display order is determined by the POC parameters *TopFieldOrderCount* and *BottomFieldOrderCount* which are derived from the slice header using one of three methods:

- Type 0: The least significant bits of POC are sent in every slice header. This allows maximum flexibility but typically requires more bits than the other methods.

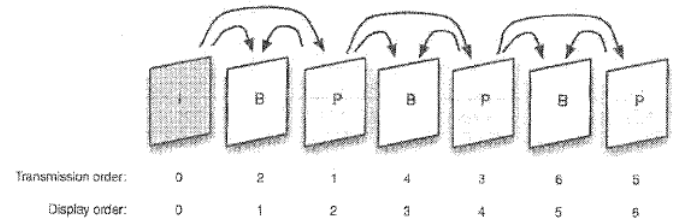


Figure 5.4 Display order: Type 0 example

- Type 1: A 'cycle' of POC increments is set up in the sequence parameter set and POC changes according to this cycle unless otherwise signalled in the slice header using a Delta offset. The cycle defines the interval between frames used for reference, plus a POC offset to frames not used for reference.
- Type 2: POC is derived directly from *frame_num* and display order is the same as decoding order.

Display Order Example, Type 0:

Frame pictures, one frame per slice, display order IBPBPBPBPB... (Figure 5.4).

In this example, the B slices, bipredicted slices in which each macroblock or partition can be predicted from up to two reference pictures, are not used for reference prediction of any other pictures. POC increments by two for every complete frame, i.e. every two fields. Note that *frame_num* increments **after** each reference picture is transmitted. In this example, only the I and P pictures are reference pictures. Increments in *frame_num* are indicated in bold type.

Slice	Type	Used for reference	*frame_num*	POC LSBs	Display order
1st	I	Yes	**0**	0	0
2nd	P	Yes	**1**	4	2
3rd	B	No	**2**	2	1
4th	P	Yes	2	8	4
5th	B	No	**3**	6	3
6th	P	Yes	3	12	6
7th	B	No	**4**	10	5
8th	P	Yes	4	16	8
...	...				

Display Order Example, Type 1:

Frame pictures, one frame per slice, display order: IBBPBBPBBPB... (Figure 5.4)

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.0011

In this example, the B slices are not used for reference. The POC cycle consists of one reference frame, with an offset from the previous reference frame to the next reference frame of +6 and an offset from the previous reference frame to the next non-reference frame of −4.

Slice	Type	Used for reference	*frame_num*	Delta POC	Offset	POC	Display order
1st	I	Yes	**0**	0	0	0	0
2nd	P	Yes	**1**	0	+6	6	3
3rd	B	No	**2**	0	−4	2	1
4th	B	No	2	2	−4+2	4	2
5th	P	Yes	2	0	+6	12	6
6th	B	No	**3**	0	−4	8	4
7th	B	No	3	2	−4+2	10	5
8th	P	Yes	3	0	+6	18	9
...	...						

The 4th and 7th slices have a delta POC, signalled in the slice header, of +2. The POC for these slices is calculated as: POC = expected POC + 2.

5.3.3 *Reference picture lists*

A picture that is coded and available for reference is stored in the Decoded Picture Buffer (DPB) and marked as one of the following:

(a) a short term reference picture, indexed according to *frame_num* or POC or,
(b) a long term reference picture, indexed according to *LongTermPicNum*, a reference number derived from the parameter *LongTermFrameIdx* which is assigned when a picture is marked as a long term reference picture.

Short term reference pictures may be assigned a *LongTermFrameIdx*, i.e. changed to a long term reference picture, at a later time.

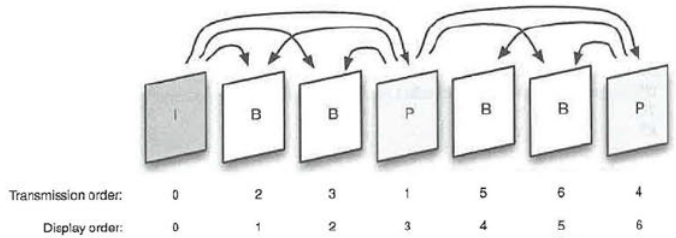


Figure 5.5 Display order: Type 1 example

Short term reference pictures are removed from the DPB (a) by an explicit command in the bitstream or (b) when the DPB is full and an automatic mode of DPB handling is in use, in which case the oldest short term picture is removed. Long term pictures are removed by an explicit command in the bitstream.

5.3.3.1 Default reference picture list order

Reference pictures are ordered in one or two lists prior to encoding or decoding a slice. A P slice uses a single list of reference pictures, list0 and a B slice uses two lists, list0 and list1. In each list, short term reference pictures are listed first followed by long term reference pictures, in increasing order of *LongTermPicNum*. The default short term reference picture order depends on **decoding order** when the current slice is a P slice and on **display order** when the current slice is a B slice. The list orders are important, since indices to reference pictures earlier in the list require fewer bits to signal. Hence the default orders are organized so that reference pictures temporally 'nearer' to the current picture occur earlier in the list, since these are likely to contain the best prediction match for the current picture.

List0 (P slice) : The default order is in decreasing order of *PicNum*. *PicNum* is a version of *frame_num*, 'wrapped around' modulo *MaxFrameNum*.

List0 (B slice) : The default order is (1) in decreasing order of POC, for pictures with POC earlier than current picture, then (2) in increasing order of POC, for pictures with POC later than the current picture.

List1 (B slice) : The default order is (1) in increasing order of POC, for pictures with POC later than the current picture, then (2) in decreasing order of POC, for pictures with POC earlier than current picture.

Reference Picture Order Example: P slices (list0)

The reference picture list is initially empty. The current *frame_num* is 150 and the maximum size of the DPB is 5 frames. Italics indicate a *LongTermPicNum*. Notice that by default, the first entry in list0 is the most recently coded frame. This is likely to be the 'best' prediction reference for the majority of macroblocks.

Operation	list0(0)	list0(1)	list0(2)	list0(3)	list0(4)
Initial state	-	-	-	-	-
Encode frame 150	150	-	-	-	-
Encode 151	151	150	-	-	-
Encode 152	152	151	150	-	-
Encode 153	153	152	151	150	-
Encode 154	154	153	152	151	150
Encode 155	155	154	153	152	151
Assign 154 to LongTermPicNum *3*	155	153	152	151	*3*
Encode 156 and mark it as LongTermPicNum *1*	155	153	152	*1*	*3*
Encode 157	157	155	153	*1*	*3*
...					

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00012

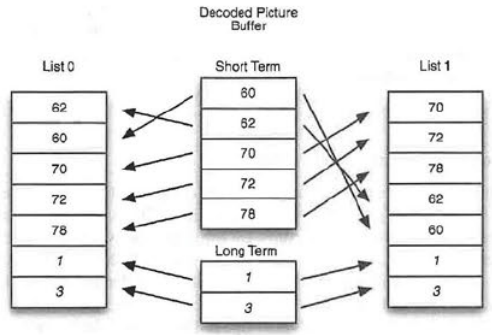


Figure 5.6 List 0 and List 1 ordering: example

Reference Picture Order Example: B slice (list0 and list1)

The DPB contains short term reference pictures with POC 60, 62, 70, 72 and 78 and long term reference pictures with LongTermFrameIdx *1* and *3*. The current POC is 68. The default order of list0 is:

62, 60, 70, 72, 78, *1*, *3*

The default order of list1 is:

70, 72, 78, 62, 60, *1*, *3*

Figure 5.6 shows how the two lists are created from the sequence of pictures in the DPB. List0 defaults to prediction from 'past' pictures in display order whilst List1 defaults to 'future' pictures in display order. This is shown graphically in Figure 5.7. Note that the first entry in

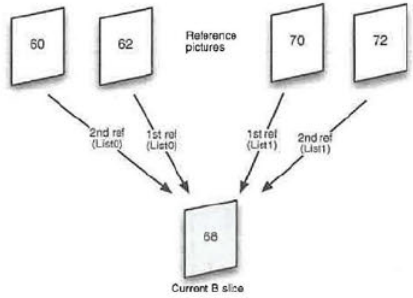


Figure 5.7 Default reference pictures: example

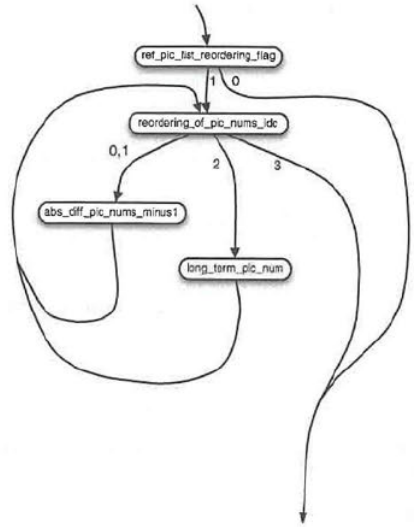


Figure 5.8 Reference picture re-ordering syntax, simplified overview

list0 is the most recent **past** picture (62) and the first entry in list1 is the nearest **future** picture (70). These are likely to be the best 'past' and 'future' prediction references respectively.

5.3.3.2 Changing the reference picture list order

The default order of list0, and of list1 in the case of a B slice, may be modified by explicit commands in the slice header. These commands are initiated by the syntax element *ref_pic_list_reordering_flag* and change the list order **for the current slice only**. This may be used, for example, to enable an encoder to place a particularly useful reference frame earlier in the list than its usual position, since earlier reference indices cost fewer bits to signal.

The reference picture list re-ordering process is illustrated in Figure 5.8 and proceeds as follows for list 0. This is a simplified overview.

- Initialize a pointer (refIdxL0) to point to the first reference picture index in the list (0)
- While *reordering_of_pic_nums_idc* $\neq 3$
  - Select a reference picture, either short term, indicated by *abs_diff_pic_num_minus1*, or long term, indicated by *long_term_pic_num*
  - Move all other pictures from position refIdxL0 onwards in the list to one position later
  - Put this picture in the list at the position indicated by refIdxL0
  - Increment pointer refIdxL0.

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00013

Table 5.3 Selecting a reference picture to re-map

reordering_of_pic_nums_idc	remappedPicNum is calculated as:
0	predictedPicNum – (*abs_diff_pic_num_minus_1* +1)
1	predictedPicNum + (*abs_diff_pic_num_minus_1* +1)
2	*long_term_pic_num*
3	(Exit the re-ordering process)

The value *abs_diff_pic_num_minus1*+1 signals a positive or negative offset from a **predicted** reference picture number. For the first reordering or remapping instruction, the predicted picture number predictedPicNum is the current picture number. For subsequent reordering instructions, the predicted picture number is the picture number of the last remapped picture. The picture to be remapped or moved, remappedPicNum, is selected according to the following rules shown in Table 5.3.

Reference Picture Re-Ordering Example:
The current slice is a P slice. The DPB contains five reference frames, indexed in list0. Three of these are short-term frames with PicNum equal to 153, 155, and 157, respectively. The other two are long-term frames with LongTermPicNum equal to *1* and *3*, respectively. The current *frame_num* is 158. The default reference list order (list0) is as follows (Table 5.4).

Table 5.4 Initial reference picture list

Index	0	1	2	3	4
Picture	157	155	153	*1*	*3*

The following series of four reference picture reordering commands are received.
Initial predictedPicNum = 158, the current picture; initial refIdxL0 = 0

1. *reordering_of_pic_nums_idc = 0, abs_diff_pic_num_minus_1 = 4*
   remappedPicNum = 158 – 5 = 153
   Put picture 153 at position refIdxL0=0 (the start of the list).
   New list: **153**, 157, 155, *1*, *3*
   New predictedPicNum = 153; new refIdxL0 = 1.
2. *reordering_of_pic_nums_idc = 1, abs_diff_pic_num_minus_1 = 1*
   remappedPicNum = 153 + 2 = 155
   Put picture 155 at position refIdxL0=1, second in the list.
   New list: 153, **155**, 157, *1*, *3*
   New predictedPicNum = 155; new refIdxL0 = 2.
3. *reordering_of_pic_nums_idc = 2, long_term_pic_num* = 3
   remappedPicNum= *3*
   Put long term picture *3* at position refIdxL0=2, third in the list.
   New list: 153, 155, **3**, 157, *1*
4. *reordering_of_pic_nums_idc* = 3
   End of reordering process.

The final list0 order is as follows (Table 5.5).

Table 5.5 Final reference picture list

Index	0	1	2	3	4
Picture	153	155	*3*	157	*1*

5.3.4 Frame and field coding

A video sequence may be coded in Frame mode only, with no special coding of fields or interlaced video, or in Frame/Field mode, using special coding tools for interlaced video, signalled by the syntax element *frame_mbs_only_flag* in the Sequence Parameter Set (section 5.5). If this flag is set to zero, special coding of fields or interlaced video is enabled.

5.3.4.1 Coding pictures in frame or field mode

If frame/field coding is enabled, each frame of video, a pair of fields, may be coded as a complete frame, i.e. one picture = one frame, or as two separate fields, i.e. one picture = one field. The syntax element *field_pic_flag* is signalled in each Slice Header (section 5.6.2) and indicates whether the current coded picture is a frame or a field. This gives rise to the following two cases:

1. Picture is coded as a complete frame: The complete frame, which happens to include two fields, is coded as one or more slices, each containing an integral number of macroblocks. Reference picture list(s), used for motion compensated prediction of P- or B-macroblocks, are constructed from previously coded **frames**, i.e. each reference picture is a complete coded frame. Block coefficients are scanned in a zig-zag order (Chapter 7). Macroblocks are optionally coded as macroblock pairs in Macroblock Adaptive Frame Field Mode.
2. Picture contains one field: The current top or bottom field is coded as one or more slices, each containing a number of macroblocks. Reference picture list(s) are constructed from previously coded **fields**, i.e. each reference picture is a coded field. Each field of a stored reference frame is identified as a separate reference picture with a unique index[1]. An inter-coded macroblock is predicted from region(s) in previously coded field(s). Block coefficients are coded in a modified or field scan order (Chapter 7).

The coded frames in the Decoded Picture Buffer (DPB) may therefore be accessed as complete frames, containing top + bottom field, or as individual fields, depending on whether the current picture is coded in frame or field mode.

An encoder may choose to switch adaptively between frame and field coding during a sequence, for example to improve the compression performance for interlaced video or for progressive video that was converted from an interlaced source. For an interlaced video

[1] When decoding a field, there are effectively twice as many reference pictures available as there would be when decoding a frame at the same position.

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00014

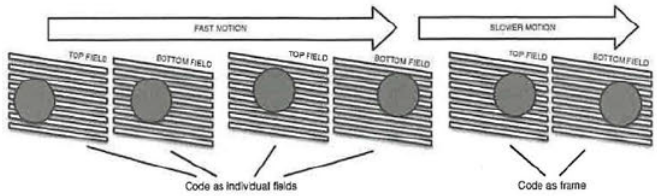


Figure 5.9 Picture Adaptive Frame Field Coding example

sequence, field coding tends to be more efficient when there is significant motion in the scene which tends to give larger changes between successive fields, whereas frame coding tends to be more efficient in more static areas of the scene where there is less change between successive fields. Switching between frame and field coding at the picture level is known as Picture Adaptive Frame Field Coding (PAFF).

Example:

Figure 5.9 shows a sequence of six fields from an interlaced sequence or from a progressive sequence that was converted from interlaced content. The first four contain rapid motion, a fast-moving ball, and so the encoder chooses to code each field as an individual picture using field coding mode. Each field is coded as a unit; reference picture list(s) consist of previously-coded fields; block scanning uses the modified field scan described in Chapter 7. The last two fields contain slower motion and the encoder chooses to code these as a complete frame. The combined frame is coded as a single picture; reference picture list(s) consist of previously-coded frames; block scanning uses the zig-zag progressive scan described in Chapter 7.

5.3.4.2 Coding macroblocks in frame or field mode (MBAFF)

Macroblock Adaptive Frame Field Coding (MBAFF) [iii] is a coding mode that is enabled in the Sequence Parameter Set (section 5.5) and is available thereafter in any slice coded in Frame mode. MBAFF makes it possible to adaptively switch between frame and field coding within a single frame, for example to maximise coding efficiency. Macroblocks are handled in pairs consisting of two vertically adjacent macroblocks (Figure 5.10).

For each macroblock pair, *mb_field_decoding_flag*, signaled in the header of the upper macroblock, indicates whether the pair is coded as a frame macroblock pair or a field macroblock pair (Figure 5.11). If it is a **frame MB pair**, then the top and bottom macroblocks are coded separately as frame macroblocks. Blocks are scanned using zig-zag scan (Chapter 7). If it is a **field MB pair**, alternate lines of the MB pair are mapped to top and bottom field macroblocks as shown in the figure. Each field macroblock is predicted from the corresponding field in reference picture(s) (Chapter 6). Blocks in a field MB are scanned using field scan (Chapter 7).

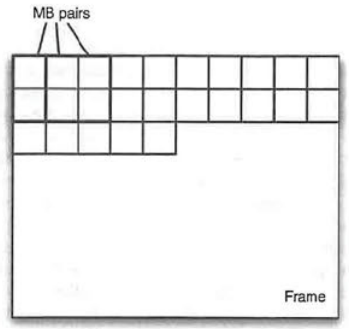


Figure 5.10 Frame with macroblock pairs

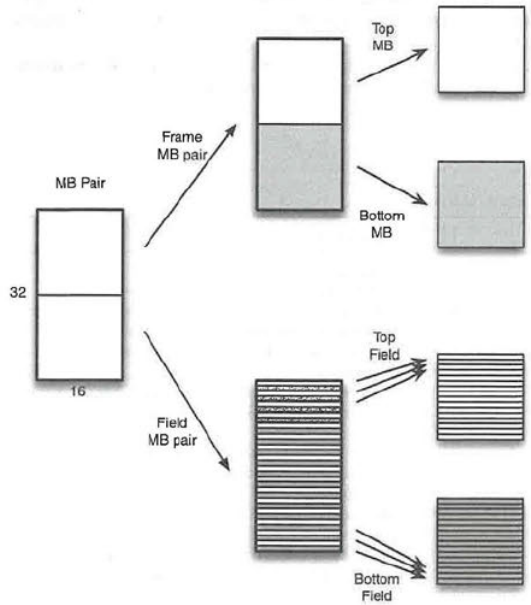


Figure 5.11 MB pair coded using Macroblock Adaptive Frame Field Coding

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00015

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

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00016

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.

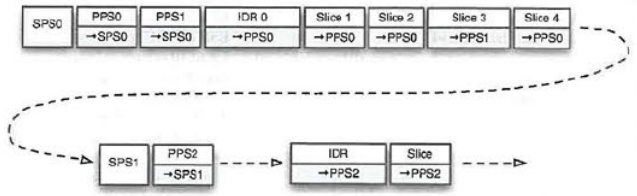


Figure 5.12 Example: Sequence and Picture Parameter Sets

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

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00017

Table 5.10 Slice Header, IDR/Intra, Frame 0

Parameter	Binary code	Symbol	Discussion
first_mb_in_slice	1	0	First MB is at position 0, the top-left position in the slice.
slice_type	1000	7	I slice, contains only I MBs
pic_parameter_set_id	1	0	Use PPS 0
frame_num	0	0	Slice is in frame 0
idr_pic_id	1	0	IDR #0 : only present in IDR picture
pic_order_cnt_lsb	0	0	Picture order count = 0
no_output_of_prior_pics_flag	0	0	Not used
long_term_reference_flag	0	0	Not used
slice_qp_delta	1000	4	QP = initial QP + 4 = 30

shows an example of a slice header from an I slice with frame_mbs_only_flag set, i.e. no field coding. This is the first I slice and is therefore an IDR slice. The default QP is set to the initial sequence value (26) plus 4.

The header of a P slice is shown in Table 5.11. This is frame 1; the picture order count (POC) is 2. Note that POC increases by 2 for every complete frame, see section 5.3.2. There is one active reference picture which happens to be frame 0.

5.6.3 Slice data

The slice data section consists of a series of macroblocks that make up a slice. A macroblock containing no data, a skip macroblock, is a very common occurrence in many coded sequences. It is signalled by the parameters mb_skip_run, a count of a sequence of skipped macroblocks, used with CAVLC entropy coding mode, or mb_skip_flag, indicating a single skipped macroblock, used with CABAC entropy coding mode. A skip macroblock can only

Table 5.11 Slice Header, Inter, Frame 1

Parameter	Binary code	Symbol	Discussion
first_mb_in_slice	1	0	First MB at position 0
slice_type	110	5	P slice, can contain I or P MBs
pic_parameter_set_id	1	0	Use PPS 0
frame_num	1	1	Frame 1
pic_order_cnt_lsb	10	2	Picture 2
num_ref_idx_active_override_flag	1	1	Default number of reference pictures overridden by following parameter
num_ref_idx_l0_active_minus1	1	0	One reference picture in List 0
ref_pic_list_reordering_flag_l0	0	0	No re-ordering of reference pictures
adaptive_ref_pic_buffering_flag	0	0	Reference pictures handled as 'first in / first out' (default)
slice_qp_delta	1000	4	QP = 26 + 4 = 30

occur in a P, SP or B slice. A complete slice data section consists of coded and skipped macroblocks that comprise a single coded slice (Figure 5.1).

5.7 Macroblock layer

5.7.1 Overview

The Macroblock Layer contains all the syntax elements necessary to decode a single macroblock (Figure 5.13).

mb_type indicates the macroblock coding type, I, SI, P or B, and further information about macroblock prediction and coding. An I macroblock (Intra) can occur in any slice type and is coded without reference to any other slices. An SIntra MB (SI) occurs only in switching or SI

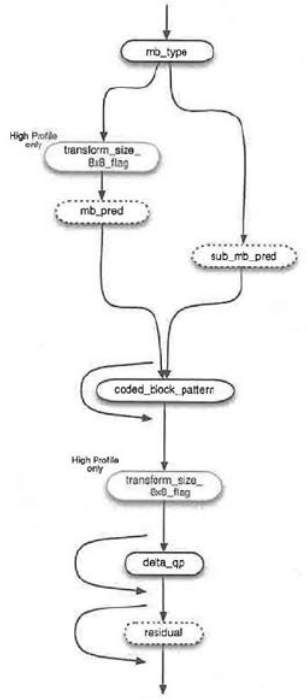


Figure 5.13 Macroblock layer syntax overview

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00018

Table 5.12 Macroblock types

Macroblock coding type	Further information signalled in mb_type codeword
I	4 × 4 : indicates 4 × 4 Intra prediction mode. 8 × 8 : indicates 8 × 8 Intra prediction mode. 16 × 16 : indicates 16 × 16 Intra prediction mode; luma and chroma coded block pattern is also indicated by mb_type. PCM: indicates that the special PCM mode is used (section 5.7.2).
SI	None.
P	16 × 16, 16 × 8, 8 × 16: indicates partition size. 8 × 8: indicates 8 × 8 partition size, a further syntax element sub_mb_type and optionally a reference index are also transmitted. P_Skip (inferred): indicates Skip mode, no further data transmitted.
B	16 × 16: indicates 16 × 16 partition size. Prediction mode may be Direct, no motion vectors or references transmitted, Pred, prediction from one reference picture in list 0 or list 1, or BiPred, prediction from two references, one from list 0 and one from list 1. 16 × 8: indicates 16 × 8 partition size plus prediction modes of each partition, either Pred from L0, Pred from L1 or BiPred. 8 × 16: indicates 8 × 16 partition size plus prediction modes of each partition, Pred from L0 or L1, or BiPred. 8 × 8: indicates 8 × 8 partition size. A further syntax element sub_mb_type is sent for each 8 × 8 partition. B_Skip (inferred): indicates Skip mode, no further data transmitted.

slices (Chapter 8). A P macroblock (Predicted) can occur in P or SP slices and is inter-coded with one prediction reference. A B macroblock (BiPredicted) can occur in B slices and is inter-coded with one or two prediction references. Table 5.12 summarizes the information signalled by the codeword mb_type. Note that mb_type is not actually transmitted for skip MBs in P or B slices – these macroblock types are inferred whenever mb_skip is signalled in the slice data layer.

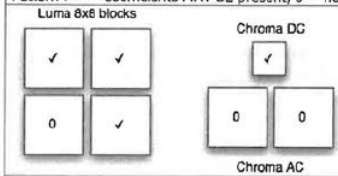
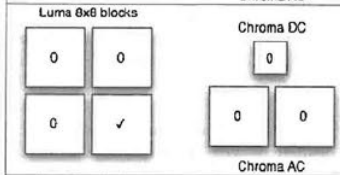
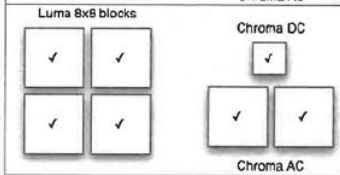
transform_size_8 × 8_flag is only present in High Profile bitstreams. It occurs in one of two places (Figure 5.13) depending on the macroblock type and does not occur in an Intra 16 × 16 macroblock. This flag indicates that an optional 8 × 8 integer transform is used for luma blocks, instead of the default 4 × 4 integer transform (Chapter 7).

mb_pred, for all macroblock types except P/B with 8 × 8 partition size, or **sub_mb_pred**, for P/B macroblocks with 8 × 8 partition size, indicates the type of intra or inter prediction used for the macroblock (section 5.7.3 and Chapter 6).

coded_block_pattern is a separate syntax element for macroblocks other than Intra_16 × 16_ types. It can take values between 0 and 47 and is constructed as follows:

(i) The four LSBs of coded_block_pattern, $b_3b_2b_1b_0$, indicate whether there are one or more non-zero transform coefficients within each of the four 8 × 8 luma blocks. A 1 = non-zero coefficient(s) present, 0 = no coefficients present.

Table 5.13 coded_block_pattern examples.

Pattern : ✓ = coefficients MAY BE present, 0 = no coefficients	Coded_block_pattern
	$\mathbf{01}1011_2 = 27_{10}$
	$\mathbf{00}1000_2 = 8_{10}$
	$\mathbf{10}1111_2 = 47_{10}$

(ii) The two MSBs, b_5b_4, can take the values 00_2, indicating no chroma coefficients present, 01_2, indicating chroma DC coefficients are present, no chroma AC coefficients, or 10_2, indicating chroma DC coefficients may be present, chroma AC coefficients are present.

Table 5.13 shows three examples; a tick indicates that coefficients may be present in the block, a 0 indicates that no non-zero coefficients are present.

delta_qp indicates a change in quantization parameter (QP), either positive or negative, from its previous value. If there is no change in QP from the previous value, delta_qp = 0.

If residual coefficient data is present, i.e. if there are non-zero coefficients in the macroblock, as indicated by a non-zero CBP, **residual_data** is then transmitted (section 5.7.4).

5.7.2 *The Intra PCM mode*

The Intra PCM (Pulse Code Modulation) mode is an optional mode of coding a macroblock, signalled by mb_type I_PCM. If I_PCM mode is chosen, the usual prediction, transform and

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00019

coding processes are bypassed. Instead, each luma and chroma sample is transmitted directly as follows:

1. Send zero bits until the current position in the bitstream is byte aligned.
2. Send every luma sample, 256 samples in total, as an individual BitDepth_Y-bit value.
3. Send every chroma sample, 64 samples for 4:2:0 format, as an individual BitDepth_C-bit value.

BitDepth_Y and BitDepth_C are the number of bits used to represent an uncompressed luma or chroma sample respectively. Each BitDepth parameter defaults to 8, i.e. 1 byte per sample, but higher bit depths may be set in the SPS for certain Profiles (Chapter 8).

Clearly, I_PCM mode does not provide compression, as the 'raw' luma and chroma samples are simply inserted into the H.264/AVC bitstream. However, there are situations when I_PCM mode may be useful, for example:

1. When video is coded at a very high perceived quality / very low QP setting, for example, for applications such as content archiving or distribution where high quality is required, there may occasionally be macroblocks which, when coded using the usual processes of prediction, transform and entropy coding, generate **more** bits than the original 'raw' data.
2. If I_PCM mode is selected for every macroblock, the H.264/AVC bitstream becomes a 'container' for uncompressed video. This makes it possible for an application to make use of all the other features of H.264/AVC such as packet handling, frame numbering and transport support, whilst retaining the original video samples.

5.7.3 *Macroblock prediction*

The macroblock prediction syntax elements indicate how intra or inter prediction is carried out for the current macroblock. Sub_mb_pred is used for P or B macroblocks coded with 8×8 partition size and mb_pred is used for all other cases (Figure 5.14). For a detailed description of intra and inter prediction, see Chapter 6.

I macroblock: If the prediction type is 4×4 or 8×8, the prediction modes for each 4×4 or 8×8 luma block are signalled. Note that if the prediction type is 16×16, the mode has already been signalled as part of mb_type (section 5.7.1). The chroma prediction mode is signalled. Section 5.2.5 explains how these modes are coded.

B macroblock, Direct 16×16 mode: No further prediction information is sent; List0 and List1 reference frame indices and List 0 and List 1 motion vectors are derived from previously coded macroblocks (Chapter 6).

P or B macroblock, 8×8 partition size: For each of the four 8×8 partitions or sub-macroblocks, sub_mb_type is sent indicating for the sub-macroblock:

(a) Direct mode or
(b) Sub-macroblock partition size, 8×8, 8×4, 4×8 or 4×4, and prediction sources, List 0, List 1 and/or BiPredicted. This is followed by some or all of the following elements, depending on sub_mb_type:

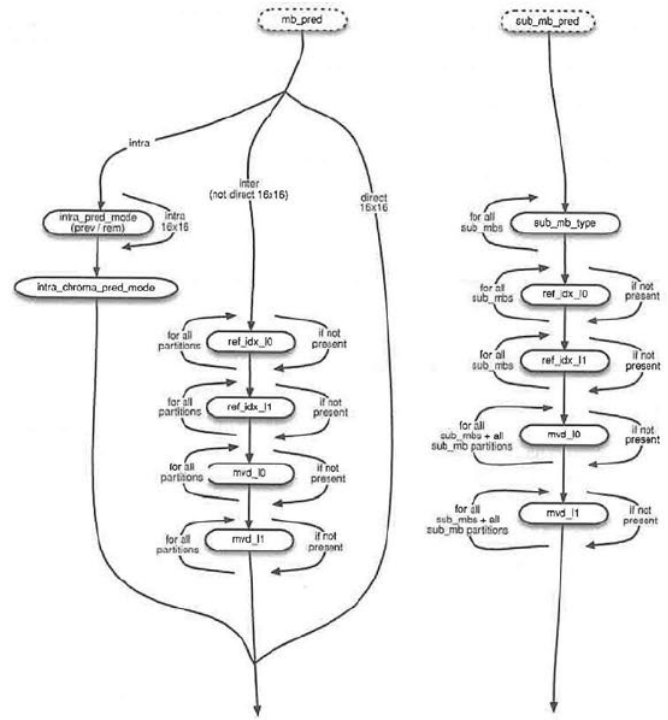


Figure 5.14 mb_pred and sub_mb_pred syntax overview

(i) Reference picture index/indices for List 0 and/or List 1, one index or one pair of indices for each sub-macroblock
(ii) Motion vector differences (mvd) for List 0 and/or List 1.

Note that all sub-macroblock partitions in a sub-macroblock share the same reference picture(s), whereas each sub-macroblock partition has its own motion vector (x,y) pair.

All other P or B macroblocks: Some or all of the following are sent for each macroblock partition, i.e. one 16×16 partition, two 8×16 partitions or two 16×8 partitions:

(a) Reference picture index/indices for List 0 and/or List 1
(b) Motion vector differences for List 0 and/or List 1.

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00020

Table 5.14 Type and prediction elements for B macroblocks, excluding Direct or 8 × 8 partitions

Partition	Type (part 0)	Type (part 1)	Number of reference indices	Number of mvd (x,y) pairs
16 × 16	L0 or L1	N/A	1	1
16 × 16	Bipred	N/A	2 : one for each List	2
16 × 8	L0 or L1	L0 or L1	2 : one for each partition	2
"	L0 or L1	Bipred	3 : one for partition 0, two for partition 1	3
"	Bipred	L0 or L1	3 : two for partition 0, one for partition 1	3
"	Bipred	Bipred	4 : two for each partition	4
8 × 16	Same options as for 16 × 8			

The motion vector differences (mvd) are added to a predicted motion vector to create the x and y components of each motion vector (Chapter 6).

As an example, Table 5.14 lists the type and prediction elements, reference indices and motion vector difference (mvd) pairs, for B macroblocks excluding the special cases of Direct and 8 × 8 partition types. Each partition has one or two reference frames, one from List 0, one from List 1 or one from each List, i.e. bipredíction, and a corresponding number of mvd(x,y) pairs signalled in the bitstream.

Note that each syntax element is only sent if it is needed. For example, if there is only one reference frame in a List, it is not necessary to send a reference index for that List and so the decoder does not expect to receive one.

5.7.4 *Residual data*

Residual data for a complete macroblock, if present, is sent according to the syntax summarised in Figure 5.15. Figure 5.16 shows the sequence in which residual blocks are coded. First, a 4 × 4 block of transformed luma DC coefficients is sent if Intra 16 × 16 mode was chosen for this macroblock. This optional first block is labelled '-1' in the Figure 5.16. Then, the luma component is sent in 8 × 8 block order. If coded_block_pattern indicates the 8 × 8 block contains no coefficients (section 5.7.1), that block is bypassed. Each 8 × 8 luma block is processed using four 4 × 4 transforms (Figure 5.16) or one 8 × 8 transform (Figure 5.17, High Profiles only).

After all the coded luma transform coefficients are sent, two blocks of transformed chroma DC coefficients are transmitted, if coded_block_pattern indicates they are present, e.g. for blocks 16 and 17 in Figure 5.16, blocks 4 and 5 in Figure 5.17. Finally, the chroma AC blocks are transmitted if present. If the sampling format is 4:2:0 (Chapter 2), each chroma DC block contains 2 × 2 samples and each chroma AC block contains 8 × 8 samples. These sizes change if 4:2:2 or 4:4:4 sampling is employed (Chapter 7).

Each residual block is coded using CAVLC or CABAC. The CAVLC and CABAC block syntax is summarized in Figure 5.18 and Figure 5.19 and described in detail in Chapter 7.

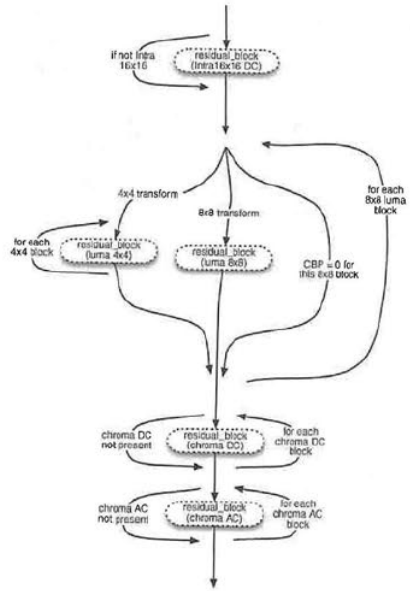


Figure 5.15 Residual data syntax overview

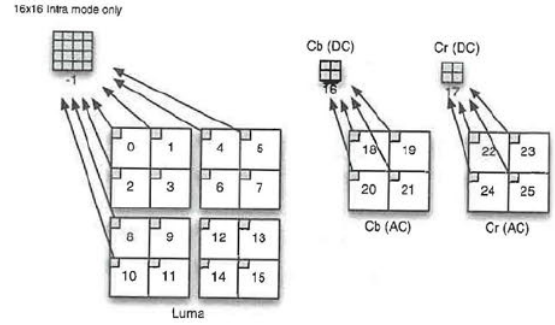


Figure 5.16 Block scanning order, 4 × 4 transform, 4:2:0 sampling

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00021

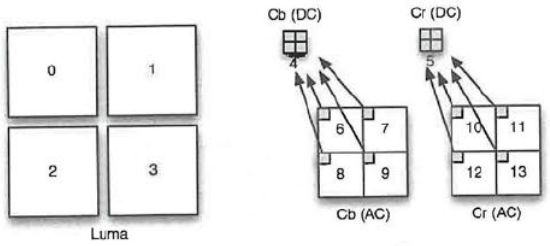


Figure 5.17 Block scanning order, 8 × 8 luma transform, 4:2:0 sampling

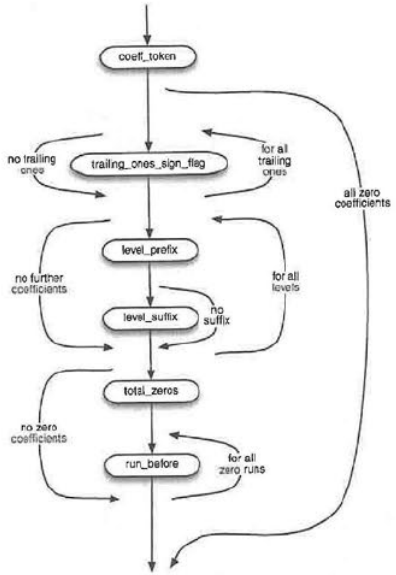


Figure 5.18 Residual CAVLC block syntax overview

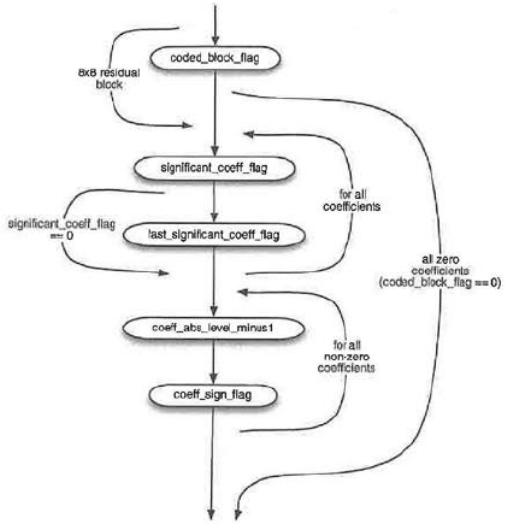


Figure 5.19 Residual CABAC block syntax overview

5.7.5 *Macroblock syntax examples*

The following examples are taken from trace files generated with the JM H.264 reference software codec [i].

1. Intra coded macroblock
   Table 5.15 shows the first part of a trace for an Intra coded macroblock. The first element, mb_type, indicates that this is an Intra macroblock using 4 × 4 prediction. The prediction modes of each 4 × 4 block are signalled, followed by the prediction mode of the chroma blocks. See Chapter 6 for a detailed description of how the intra mode is signalled using Most Probable and Rem parameters. Coded_block_pattern is $31_{10} = \mathbf{01}1111_2$, i.e. all luma blocks and the chroma DC blocks contain non-zero coefficients, but there are no coefficients present for chroma AC.
2. P macroblock, one reference frame
   In this example, illustrated in Table 5.16 and Figure 5.20, a P macroblock is coded using 16 × 16 mode, i.e. one motion vector per macroblock. There is only one reference frame and so the choice of reference is not signalled. MVDx = 0 and MVDy = −1, indicating

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00022

Table 5.15 I macroblock, example 1

Parameter	Binary code	Symbol	Discussion
mb_type (I_SLICE) (0, 1) = 9	1	0	Intra coding, 4 × 4 prediction of each luma block.
Intra mode = 3 0	0011	0,3	Rem = 3
Intra mode = −1 1	1	1	Most probable (MP)
Intra mode = 2 2	0010	0,2	Rem = 2
Intra mode = −1 3	1	1	MP
Intra mode = −1 4	1	1	MP
Intra mode = 2 5	0010	0,2	Rem = 2
Intra mode = 5 6	0101	0,5	Rem = 5
Intra mode = 5 7	0101	0,5	Rem = 5
Intra mode = 5 8	0101	0,5	Rem = 5
Intra mode = −1 9	1	1	MP
Intra mode = −1 10	1	1	MP
Intra mode = 1 11	0001	0,1	Rem = 1
Intra mode = −1 12	1	1	MP
Intra mode = 2 13	0010	0,2	Rem = 2
Intra mode = −1 14	1	1	MP
Intra mode = 3 15	0011	0,3	Rem = 3
Chroma intra pred mode	01	0	Mode 0
CBP (0, 1) = 31	10	31	Coefficients present for all luma blocks and for chroma DC blocks.
Delta QP (0, 1) = 0	1	0	No change in QP
Luma # c & tr.1s(0,0) vlc=2 #c=8 #t1=3	1101	8	Coefficient token (see Chapter 7)
Luma trailing ones sign (0,0)	0	0	Trailing ones
Luma lev (0,0) k=4 vlc=0 lev= 1	1	1	Level
Luma lev (0,0) k=3 vlc=1 lev= −1	11	−1	Level
Luma lev (0,0) k=2 vlc=1 lev= −2	11	−2	Level
Luma lev (0,0) k=1 vlc=1 lev= −4	11	−4	Level
Luma lev (0,0) k=0 vlc=2 lev= 3	100	3	Level
Luma totalrun (0,0) vlc=7 totzeros= 1	1	1	Total number of zeros
Luma run (0,0) k=7 vlc=0 run= 0	1	0	Zero run
Luma run (0,0) k=6 vlc=0 run= 1	0	1	Zero run
... (etc)			

Table 5.16 P macroblock, example 2

Parameter	Binary code	Symbol	Discussion
mb_skip_run	1	0	No preceding skip
mb_type (P_SLICE) (0, 2) = 1	1	0	16 × 16 mode
mvd_l0 (0) = 0 (org_mv 0 pred_mv 0)	1	0	X vector difference
mvd_l0 (1) = −1 (org_mv 0 pred_mv 1)	11	−1	Y vector difference
CBP (0, 2) = 1	11	1	000001_2: Only first block has coefficient data
Delta QP (0, 2) = 0	1	0	No change in QP
Luma # c & tr.1s(0,0) vlc=0 #c=3 #t1=2	101	3	Coefficient token
Luma trailing ones sign (0,0)	10	2	Trailing ones
Luma lev (0,0) k=0 vlc=0 lev= −2	1	−1	Level
Luma totalrun (0,0) vlc=2 totzeros= 2	110	2	Total zeros
Luma run (0,0) k=2 vlc=1 run= 1	1	1	Zero run
Luma run (0,0) k=1 vlc=0 run= 0	1	0	Zero run
Luma # c & tr.1s(1,0) vlc=1 #c=0 #t1=0	11	0	Zero 4 × 4 block
Luma # c & tr.1s(0,1) vlc=1 #c=0 #t1=0	11	0	"
Luma # c & tr.1s(1,1) vlc=0 #c=0 #t1=0	1	0	"

an offset of $-1/4$ sample in the y direction from the predicted motion vector (Chapter 6). The predicted vector is $(0,+1/4)$ so the actual motion vector is (0, 0). Only one 8 × 8 block, the top-left, contains non-zero coefficients. Within this 8 × 8 block, only the top-left 4 × 4 block contains non-zero coefficients. CBP indicates that coefficient information is only sent for the first 8 × 8 luma quadrant; non-zero coefficients are signalled for the first 4 × 4 block in this quadrant. The decoder fills the remaining 4 × 4 blocks with zeros.

3. P macroblock, one reference frame
Another P macroblock example is shown in Table 5.17. The two preceding macroblocks are skipped. The macroblock uses 8 × 16 prediction mode with one reference frame. The MVD for the first or left-hand partition is (0, 1/4) and MVD for the second or right-hand

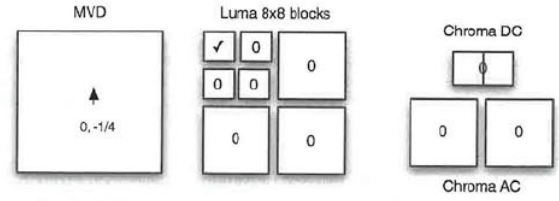


Figure 5.20 P macroblock, example 2

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00023

Table 5.17 P macroblock, example 3

Parameter	Binary code	Symbol	Discussion
mb_skip_run	11	2	Two previous MBs are skipped
mb_type (P_SLICE) (0, 3) = 3	11	2	8 × 16 mode
mvd_l0 (0) = 0	1	0	Partition 0 MVDx
mvd_l0 (1) = 1	10	1	Partition 0 MVDy
mvd_l0 (0) = −1	11	−1	Partition 1 MVDx
mvd_l0 (1) = 0	1	0	Partition 1 MVDy
CBP (0, 3) = 1	11	1	00001_2: Only first 8 × 8 block non zero
Delta QP (0, 3) = 0	1	0	No change in QP
Luma # c & tr.1s(0,0) vlc=0 #c=0 #t1=0	1	0	Zero 4 × 4 block
Luma # c & tr.1s(1,0) vlc=0 #c=1 #t1=1	1	1	Coefficient token
Luma trailing ones sign (1,0)	1	1	Trailing ones
Luma totalrun (1,0) vlc=0 totzeros= 1	11	1	Total zeros
Luma # c & tr.1s(0,1) vlc=0 #c=2 #t1=2	1	2	Coefficient token
Luma trailing ones sign (0,1)	10	2	Trailing ones
Luma totalrun (0,1) vlc=1 totzeros= 3	100	3	Total zeros
Luma run (0,1) k=1 vlc=2 run= 1	10	1	Zero run
Luma # c & tr.1s(1,1) vlc=1 #c=0 #t1=0	11	0	Zero 4 × 4 block

partition is ($-1/4$, 0). Only the first, top-left 8 × 8 luma quadrant contains non-zero coefficients and the second and third 4 × 4 blocks within this quadrant contain non-zero coefficients.

4. P macroblock, multiple reference frames
   In this example (Table 5.18, Figure 5.21) the macroblock is coded using 8 × 8 partitions. A further syntax element sub_mb_type is sent for each of the four 8 × 8 luma partitions to indicate the sub-macroblock partition types. Next, the reference frame for each macroblock partition is signalled. Note that all sub-macroblock partitions in a partition share the same reference frame. Partition 0 uses reference index 2, three frames in the past, partition 1 uses reference index 1, two frames in the past, and partitions 2 and 3 use reference index 0, one frame in the past, the default case. MVDx and MVDy are sent for each partition or sub-macroblock partition; the only non-zero values are (−3, 0) and (−3, 1). Note that these are signalled as (−12,0) and (−12, 4) respectively, in units of $1/4$ sample. CBP indicates that the only non-zero coefficients occur in the Chroma DC blocks.

5. B macroblock
   This B macroblock is coded using two 8 × 16 partitions (Table 5.19, Figure 5.22). Partition 0 is predicted from the first picture in List 1, the next frame in display order, and partition 1 is bi-predicted from the first pictures in List 0 and List 1, the previous

Table 5.18 P macroblock, example 4

Parameter	Binary code	Symbol	Discussion
mb_skip_run	1	0	No preceding skipped MBs
mb_type (P_SLICE) (5, 2) = 8	100	3	8 × 8 partitions with multiple reference frames
sub_mb_type(0)	1	0	Partition 0 mode = 8 × 8
sub_mb_type(1)	1	0	Partition 1 mode = 8 × 8
sub_mb_type(2)	10	1	Partition 2 mode = 8 × 4
sub_mb_type(3)	10	1	Partition 3 mode = 8 × 4
ref_idx_l0 = 2	11	2	Part 0 L0 ref = 2 (3 prev)
ref_idx_l0 = 1	10	1	Part 1 L0 ref = 1 (2 prev)
ref_idx_l0 = 0	1	0	Part 2 L0 ref = 0 (prev)
ref_idx_l0 = 0	1	0	Part 3 L0 ref = 0 (prev)
mvd_l0 (0) = 0	1	0	Part 0 MVDx
mvd_l0 (1) = 0	1	0	Part 0 MVDy
mvd_l0 (0) = 0	1	0	Part 1 MVDx
mvd_l0 (1) = 0	1	0	Part 1 MVDy
mvd_l0 (0) = −12	11001	−12	Part 2 sub 0 MVDx
mvd_l0 (1) = 0	1	0	Part 2 sub 0 MVDy
mvd_l0 (0) = 0	1	0	Part 2 sub 1 MVDx
mvd_l0 (1) = 0	1	0	Part 2 sub 1 MVDy
mvd_l0 (0) = 0	1	0	Part 3 sub 0 MVDx
mvd_l0 (1) = 0	1	0	Part 3 sub 0 MVDy
mvd_l0 (0) = −12	11001	−12	Part 3 sub 1 MVDx
mvd_l0 (1) = 4	1000	4	Part 3 sub 1 MVDy
CBP (5, 2) = 16	10	16	010000_2 : Only Chroma DC present
Delta QP (5, 2) = 0	1	0	No change in QP
ChrDC # c & tr.1s(0,0) vlc=0 #c=1 #t1=1	1	1	Cb DC coefficients
ChrDC trailing ones sign (0,0)	0	0	Just a single trailing one
ChrDC totalrun (0,0) vlc=0 totzeros= 2	1	2	Zero run
ChrDC # c & tr.1s(1,1) vlc=0 #c=0 #t1=0	1	0	No Cr DC coefficients

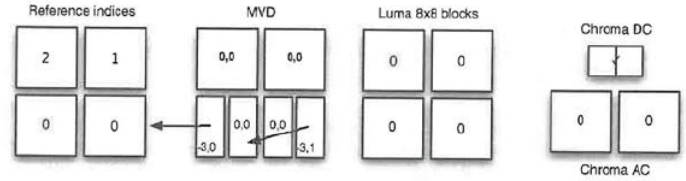


Figure 5.21 P macroblock example 4

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00024

Table 5.19 B macroblock, example 5

Parameter	Binary code	Symbol	Discussion
mb_skip_run	1	0	No preceding skipped MBs
mb_type (B_SLICE) (5, 4) = 3	10000	15	B_L1_Bi_8 × 16 : partition 0 from L1, partition 1 is bipred
ref_idx_l0 = 0	1	0	Part 1 L0 ref, order: all L0s then all L1s in partition order
ref_idx_l1 = 0	1	0	Part 0 L1 ref
ref_idx_l1 = 0	1	0	Part 1 L1 ref
mvd_l0 (0) = 11	10110	11	Part 1 L0 MVDx
mvd_l0 (1) = 2	100	2	Part 1 L0 MVDy
mvd_l1 (0) = −3	111	−3	Part 0 L1 MVDx
mvd_l1 (1) = 0	1	0	Part 0 L1 MVDy
mvd_l1 (0) = 1	10	1	Part 1 L1 MVDx
mvd_l1 (1) = −2	101	−2	Part 1 L1 MVDy
CBP (5, 4) = 0	1	0	000000_2: No coefficient data

+ next frames in display order. Partition 0 has MVD = ($-^3/_4$, 0) which, when added to the predicted vector, indicates an offset of ($7^3/_4$, $-1^1/_4$) from the reference frame, the next frame in display order. Partition 1 has two MVD pairs, ($2^3/_4$,$^1/_2$) with respect to the previous frame and ($^1/_4$,$-^1/_2$) with respect to the next frame in display order. The final motion vectors and reference regions are shown in Figure 5.22. There are no coded coefficients; this is quite common for B macroblocks, especially at lower bitrates, because inter prediction in a B macroblock tends to be very efficient, leading to minimal or zero energy in the quantized residual.

6. B macroblock
   Another B macroblock example is shown in Table 5.20. Similarly to the previous example, there are two 8 × 16 partitions. This time, the first is predicted from L0, the previous frame, and the second is bipredicted using L0, the previous frame, and L1, the next frame. Compare the order of the reference indices and motion vector differences with the previous example.

7. B macroblock
   This macroblock (Table 5.21) has a single 16 × 16 partition, bipredicted from the previous (L0) and next (L1) reference frames. One MVD pair (MVDx, MVDy) is sent for each prediction reference.

8. B macroblock
   This example is a macroblock coded using four 8 × 8 macroblock partitions (Table 5.22). Partitions 0 and 2 are predicted using Direct Prediction (Chapter 6) and so no MVD or reference index is sent for these partitions. Partition 1 is a single 8 × 8 sub-macroblock partition, predicted from L1, the next reference frame. Partition 3 contains four 4 × 4 sub-macroblock partitions, all predicted from L1. Each of these 4 × 4 sub MB partitions has a separate MVDx, MVDy pair.

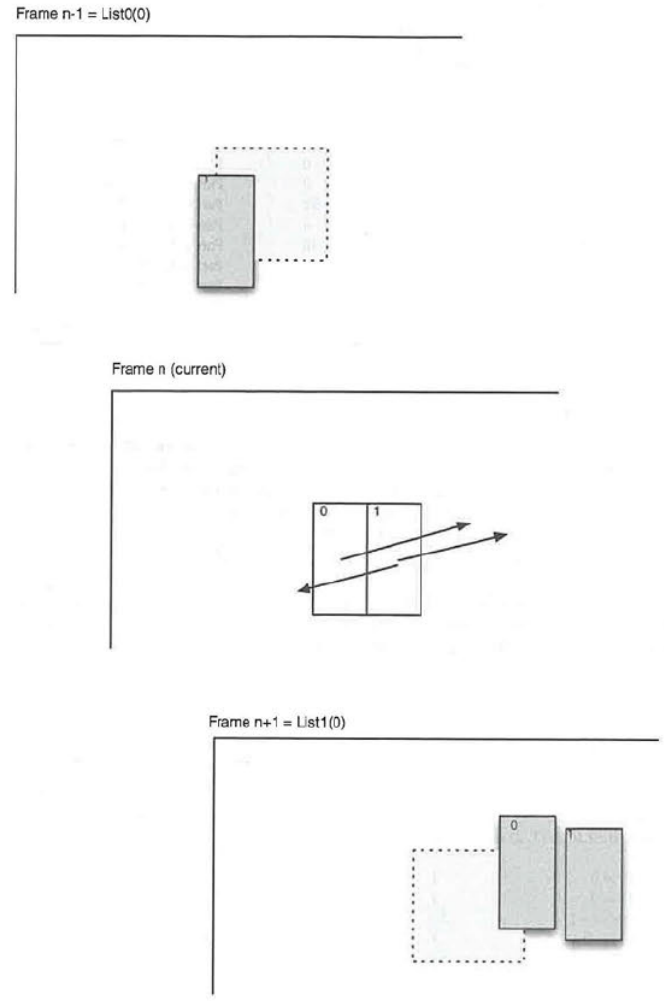


Figure 5.22 B macroblock, example 5

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00025

Table 5.20 B macroblock, example 6

Parameter	Binary code	Symbol	Discussion
mb_skip_run	1	0	No previous skip
mb_type (B_SLICE) (7, 2) = 3	1110	13	B_L0_Bi_8 × 16 : part 0 from L0, part 1 is bipred
ref_idx_l0 = 0	1	0	Part 0, L0 reference
ref_idx_l0 = 0	1	0	Part 1, L0 reference
ref_idx_l1 = 0	1	0	Part 1, L1 reference
mvd_l0 (0) = −22	101101	−22	Part 0, L0 MVDx
mvd_l0 (1) = 4	1000	4	Part 0, L0 MVDy
mvd_l0 (0) = 10	10100	10	Part 1, L0 MVDx
mvd_l0 (1) = 0	1	0	Part 1, L0 MVDy
mvd_l1 (0) = 11	10110	11	Part 1, L1 MVDx
mvd_l1 (1) = −1	11	−1	Part 1, L1 MVDy
CBP (7, 2) = 0	1	0	000000_2 : No coefficient data

5.8 Summary

At the heart of the H.264/AVC standard is the syntax, a specific format for representing a coded video sequence, together with ancillary information. A decoder parses the syntax, extracts the parameters and data elements and can then proceed to decode and display video. The syntax is organized hierarchically, from a complete video sequence at the highest level, down to coded macroblocks and blocks. A key feature of H.264/AVC is its highly flexible handling of pictures and predictions. Picture management procedures make it possible to construct and use a very large variety of prediction structures for intra and inter prediction. This flexibility makes it possible to accurately predict or estimate the current section of an image, which in turn leads to high compression efficiency. In the following chapters we will examine the prediction of macroblocks, transform and quantization of residuals and coding of syntax elements.

Table 5.21 B macroblock, example 7

Parameter	Binary code	Symbol	Discussion
mb_skip_run	110	5	5 previous MBs are skipped.
mb_type (B_SLICE) (7, 2) = 1	100	3	B_Bi_16 × 16 : one partition, bipredicted
ref_idx_l0 = 0	1	0	L0 reference
ref_idx_l1 = 0	1	0	L1 reference
mvd_l0 (0) = −1	11	−1	L0 MVDx
mvd_l0 (1) = 0	1	0	L0 MVDy
mvd_l1 (0) = −1	11	−1	L1 MVDx
mvd_l1 (1) = −1	11	−1	L1 MVDy
CBP (7, 2) = 0	1	0	No coefficient data

Table 5.22 B macroblock, example 8

Parameter	Binary code	Symbol	Discussion
mb_skip_run	10	1	1 previous MB is skipped
mb_type (B_SLICE) (6, 6) = 8	10111	22	B_8 × 8
sub_mb_pred(0)	1	0	Direct, no MVD or ref_ix
sub_mb_pred(1)	11	2	B_L1_8 × 8, 8 × 8 predicted from L1
sub_mb_pred(2)	1	0	Direct, no MVD or ref_ix
sub_mb_pred(3)	1100	11	B_L1_4 × 4, 4 × 4 predicted from L1
ref_idx_l1 = 0	1	0	L1 reference for 8 × 8 block 1
ref_idx_l1 = 0	1	0	L1 reference for 8 × 8 block 3
mvd_l1 (0) = −6	1101	−6	Block 1 MVDx
mvd_l1 (1) = 1	10	1	Block 1 MVDy
mvd_l1 (0) = 0	1	0	Block 3, subblock 0, MVDx
mvd_l1 (1) = 0	1	0	Block 3, subblock 0, MVDy
mvd_l1 (0) = 0	1	0	Block 3, subblock 1, MVDx
mvd_l1 (1) = 0	1	0	Block 3, subblock 1, MVDy
mvd_l1 (0) = 0	1	0	Block 3, subblock 2, MVDx
mvd_l1 (1) = −3	111	−3	Block 3, subblock 2, MVDy
mvd_l1 (0) = 0	1	0	Block 3, subblock 3, MVDx
mvd_l1 (1) = 0	1	0	Block 3, subblock 3, MVDy
CBP (6, 6) = 22	101101	22	010110_2 : Luma coefficients in 8 × 8 blocks 1 and 2. Chroma DC coefficients present, no chroma AC coefficients.
Delta QP (6, 6) = 0	1	0	No change in QP.
Luma # c & tr.1s(2,0) vlc=0 #c=0 #t1=0	1	0	First 4 × 4 block of luma 8 × 8 block 1...
....further data for luma 8 × 8 blocks 1 and 2 and chroma DC coefficients.			

5.9 References

i. JM reference software version 16.0, http://iphome.hhi.de/suehring/tml/, July 2009.

ii. T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, 'Overview of the H.264/AVC video coding standard', *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7. (4 August 2003), pp. 560–576.

iii. L. Wang, R. Gandhi, K. Panusopone, Y. Yu and A. Luthra, 'MB-Level Adaptive Frame/Field Coding for JVT', Joint Video Team Document JVT-B106, January 2002.

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00026

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, K. 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)

The H.264 Advanced Video Compression Standard, Second Edition Iain E. Richardson
© 2010 John Wiley & Sons, Ltd

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00027

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

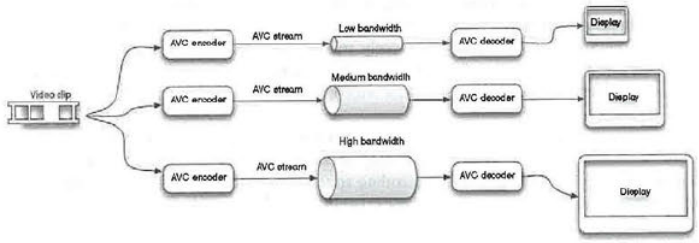


Figure 10.1 Multiple streams / simulcast

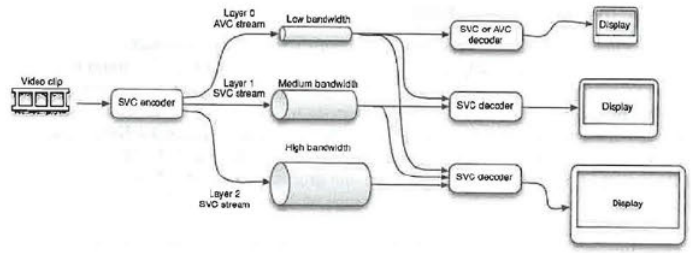


Figure 10.2 Multiple streams / scalable

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.

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00028

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.

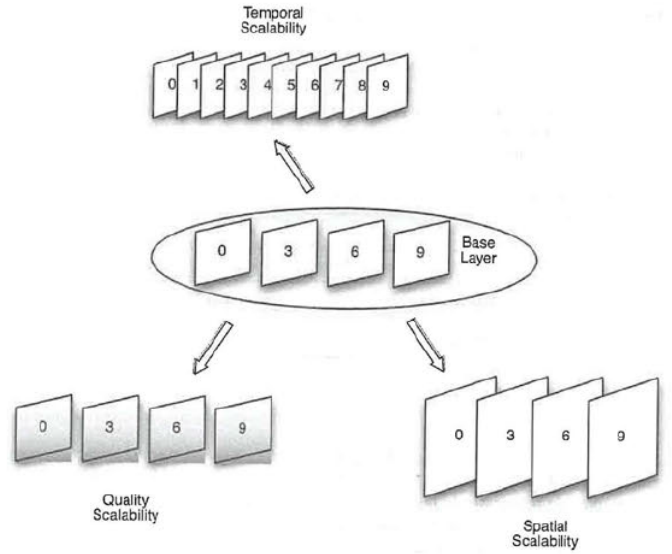


Figure 10.3 Overview of scalability types

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

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00029

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

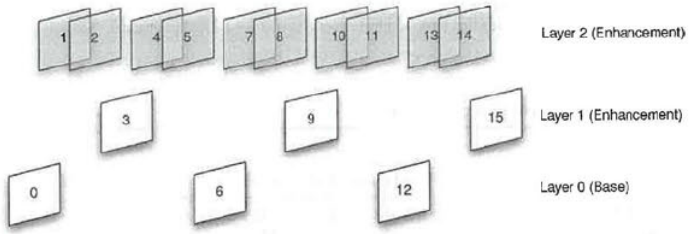


Figure 10.4 Temporally scalable sequence, 3 layers

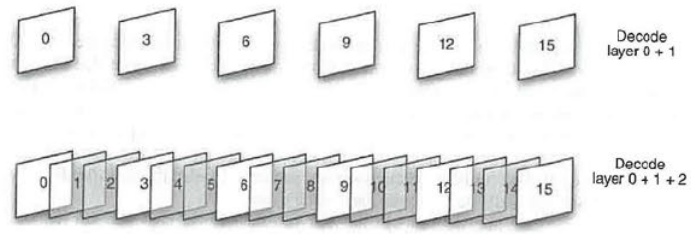


Figure 10.5 Decoding a temporally scalable sequence

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.

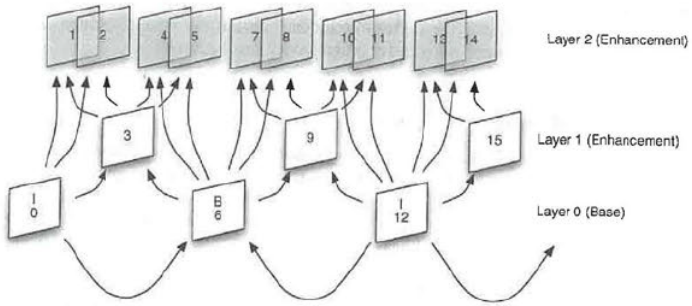


Figure 10.6 Hierarchical prediction structure

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00030

10.2.6 *Quality scalability: overview*

The base layer is coded using a particular quantizer parameter QP to produce the Layer 0 bitstream (Figure 10.7). Consider a single video frame A. At the encoder, this coded frame is decoded from the base layer and reconstructed (frame A'0). Frame A is re-coded at a lower QP and hence a higher quality, with decoded frame A'0 available as a prediction reference, to produce the enhancement layer bitstream Layer 1. Note that A'0 will typically be a very effective prediction reference as it is identical to frame A except for distortion introduced by compression. Note that the 'usual' prediction sources, previously-coded frames in the decoded picture buffer, are also available for prediction of each macroblock.

A base layer decoder simply decodes frame A0. An enhancement layer decoder requires the decoded A, a prediction reference, to reconstruct the higher-quality frame A1.

This process may be repeated to form a 'cascade' of layers 0, 1, 2, etc, each layer (a) using the reconstructed frame from the layer below as a prediction reference and (b) using a progressively lower QP. Note that SVC provides tools that make it possible to reconstruct the enhancement layers without fully decoding the base layer information (constrained inter-layer prediction).

10.2.7 *Spatial scalability: overview*

In the case of Spatial Scalability (Figure 10.8), the base layer has the lowest resolution and successive enhancement layers can be decoded to provide higher resolution decoded frames or fields.

An input video frame A is downsampled at the encoder to produce a low-resolution version A'. Frame A' is coded to produce the base Layer 0 and can be decoded to give low-resolution output frame A0. The encoder reconstructs A0 and upsamples it to produce a reference frame that has the same effective resolution as the original (A). This reference frame is used as a prediction reference, enabling the encoder to produce the enhancement Layer 1. The up-sampled A0 will typically be a good prediction reference for frame A because it is the same frame, with distortions due to downsampling, coding and up-sampling.

An enhancement layer decoder upsamples A0 and uses it to reconstruct the decoded enhancement frame A1. As with Quality scalability, this process may be repeated to give a cascade of layers 0, 1, 2, etc. The highest-resolution layer has the same resolution as the original sequence; lower layers are coded at progressively smaller resolutions.

10.2.8 *Spatial scalability in detail*

As discussed above, the base layer of a spatially scalable bitstream is encoded using the usual H.264/AVC tools described in earlier chapters. Coding a macroblock in an enhancement layer requires a number of changes depending on the type of prediction from the lower layer. H.264 SVC goes beyond the basic approach of up-sampling the lower layer (section 10.2.7) and provides several new prediction modes that improve the coding performance of spatially scalable compression.

In an enhancement layer coder, e.g. Encoder 1 in Figure 10.8, there are a number of options for predicting the current macroblock. First, all of the usual prediction options are available: intra modes using samples from the current frame at the enhancement layer resolution, inter

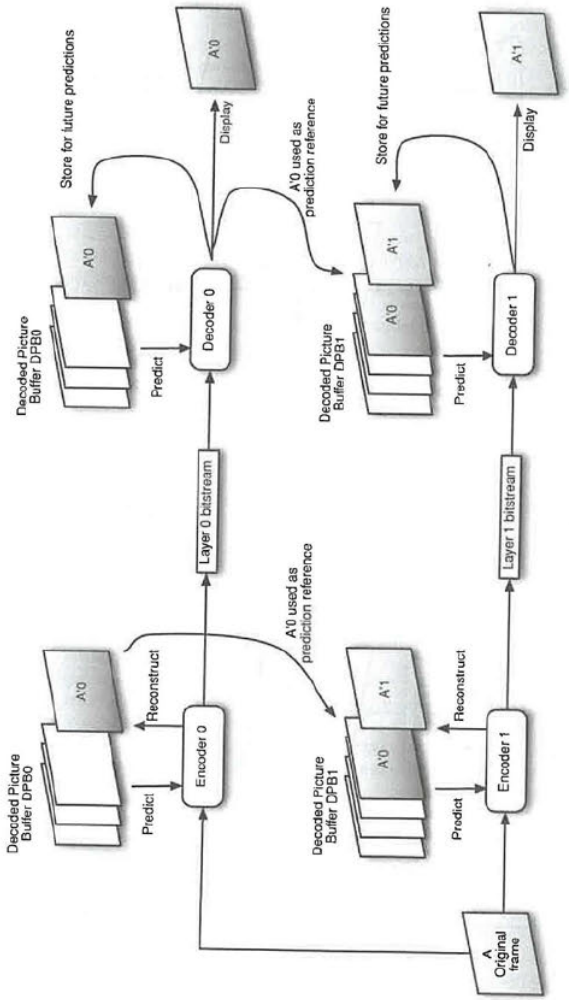


Figure 10.7 Quality Scalability

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00031

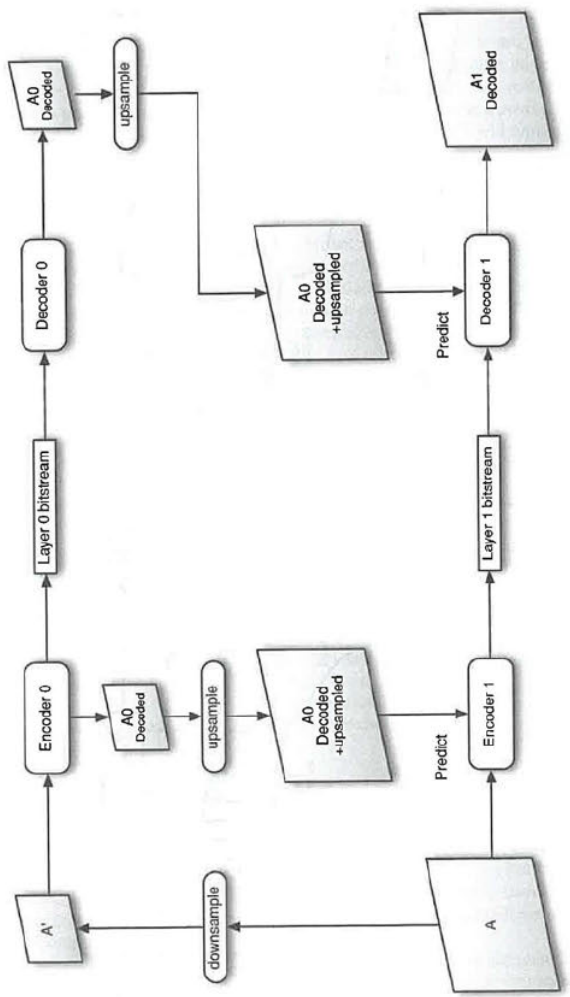


Figure 10.8 Spatial scalability, two layers

modes using samples from previously coded and reconstructed frames at the enhancement layer resolution. Second, the following further options are available, using the upsampled lower layer, the base layer in Figure 10.8, or the next lower-resolution enhancement layer, as a **reference layer**. Note that the current MB position corresponds to a smaller block in the lower-resolution layer. The following discussion assumes an 8 × 8 corresponding block in the lower, reference layer, so-called dyadic scaling, or 2 × horizontal and vertical resolution in the enhancement layer. However, arbitrary inter-layer scaling factors are supported by H.264/SVC.

Prediction options:

1. Upscale the reference layer.
   For Intra blocks, scale the reference layer to the same resolution as the current layer (Figure 10.8) and use the reference layer as an extra prediction reference.
2. Base Mode: Use the prediction choices from the reference layer block.
   When a Base Mode Flag is set to 1, only a residual is sent in the enhancement layer, with no extra prediction choices, i.e. no intra prediction modes or inter partitions, references and motion vectors.
   If the co-located block in the reference layer was coded in Intra mode, the reconstructed intra block from the reference layer is up-sampled using a 4-tap Finite Impulse Response filter to produce a prediction for the current MB. This prediction is subtracted from the MB to produce the enhancement layer residual.
   If the co-located block in the reference layer was coded in Inter mode, the enhancement layer block is predicted using Inter prediction, with (a) the same reference picture indices, (b) partition choices that are up-sampled from the partitions in the reference layer and (c) motion vectors scaled up from the reference layer motion vectors.
3. Motion vector prediction from the reference layer.
   If a Motion Prediction Flag is set to 1, the current enhancement layer macroblock partition is predicted using Inter prediction with (a) the same reference picture indices as the corresponding reference layer block and (b) motion vector differences (MVD) created using the up-scaled motion vectors of the reference layer as predictors (Chapter 6).
4. Residual prediction.
   When a Residual Prediction Flag is set to 1, the enhancement layer residual is predicted from the reference layer residual. First, the reference layer residual is up-sampled using bi-linear interpolation and this up-sampled residual is subtracted from the original enhancement layer block. Then, the enhancement layer residual is formed using any of the methods described above, i.e. conventional intra/inter prediction or base mode prediction. The resulting difference signal is transformed, coded and transmitted as usual (Chapter 7).

Note that inter-layer prediction is constrained as follows. First, the only enhancement layer macroblocks which may be coded with inter-layer intra prediction are those for which the co-located reference samples are intra coded (constrained intra prediction). Second, constrained intra prediction is mandatory for inter-layer prediction of higher layers. This means that intra coded macroblocks in reference layers can be constructed without having to reconstruct any inter coded macroblocks. Hence each layer may be decoded using a single motion compensation loop (single loop decoding), resulting in significantly lower decoder complexity than scalable video coding in earlier standards [i].

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00032

10.2.9 *Quality scalability in detail*

H.264/SVC supports Coarse Grain and Medium Grain quality scalability. Coarse Grain Quality Scalability (CGS) is effectively a special case of Spatial Scalability in which the upsampling / downsampling factor is 1. This means that the enhancement layer resolution is the same as the reference layer resolution (Figure 10.7). The enhancement layer is coded at a lower QP and hence a higher quality than the lower, reference layer. All the 'spatial' scalable coding tools described above may be use to predict the enhancement layer frame from the reference layer reconstructed frame.

A typical application of quality scalability is to provide versions of the sequence coded at different bitrates and quality levels, so that, for example, lower bitrate sub-sequences may be extracted for transmission over channels with different capacities. With CGS, the number of sub-sequence bitrates is limited by the number of layers. Providing a large number of bitrate options using CGS requires a large number of layers, which tends to be complex and inefficient to code.

Medium Grain Quality Scalability (MGS) addresses this limitation and makes it possible to extract substreams at a wide range of bitrates from a scalable bitstream with a small number of quality layers. Using MGS, any NAL unit in an enhancement layer may be discarded to leave a fully decodeable bitstream. This makes it possible to produce a variety of output bitrates. For example, discarding an arbitrary number of enhancement layer NAL units makes it possible to meet an arbitrary bit rate target, within a certain margin of error. Figure 10.9 shows an example. The complete scalable bitstream consists of Base Layer NALUs and enhancement

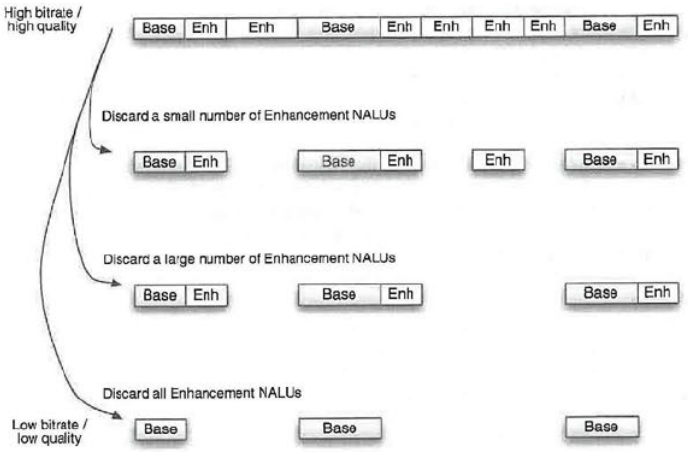


Figure 10.9 Medium Grain Quality Scalability

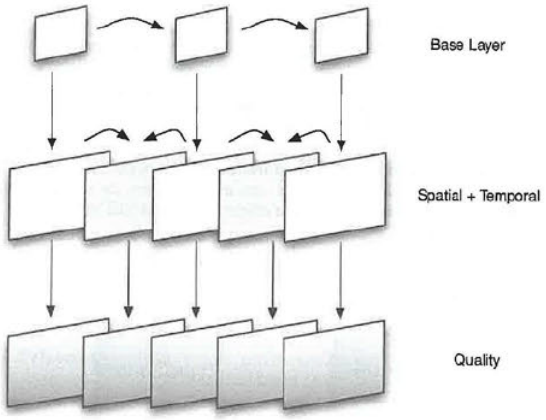


Figure 10.10 Combined Spatial, Temporal and Quality scalability

layer NALUs as shown. Using MGS, selected Enhancement Layer NALUs may be discarded to provide sub-streams at a progressively lower bitrate. The lowest bitrate / quality point is provided by the Base Layer stream, which consists only of Base Layer NALUs. SVC specifies that motion compensated prediction parameters must not change between Base and Enhancement layer representations of certain so-called Key Pictures. This prevents "drift" between the motion compensated reconstruction of these Key Pictures at the encoder and decoder. Drift is therefore restricted to non-Key Pictures.

10.2.10 *Combined scalability*

H.264/SVC provides considerable flexibility in the construction of a scalable bitstream, making it possible to mix Spatial, Temporal and Quality scalability. For example, in Figure 10.10, the base layer is up-sampled spatially and the up-sampled frames are used as references for B-slices to produce a layer with spatial and temporal scalability. This is then used to predict a further layer at the same spatio-temporal resolution but at a higher bitrate and quality.

10.2.11 *SVC performance*

For a given sequence that is required to be delivered at a range of bitrates, a key performance question is as follows – does scalable coding give a **smaller** or **larger** bitrate than simulcast coding?

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00033

Example:

Two coded versions of a sequence are required to be delivered across a network. The choices are (a) code low quality and high quality versions independently using AVC and deliver each sequence (simulcast) or (b) code a low quality Base Layer and an Enhancement Layer that can be decoded with the Base Layer to give the high quality version using scalable decoding. Figure 10.11 illustrates the total bitrates of the two choices. In this case, the base layer of the scalable version is coded at a **higher** bitrate than the corresponding simulcast version. This is generally advisable in order to provide a good reference for predictions of the enhancement layer. However, the total bitrate of the scalable version is **lower** than the combined simulcast bitrate. In this example, scalable coding is more efficient in terms of total bitrate.

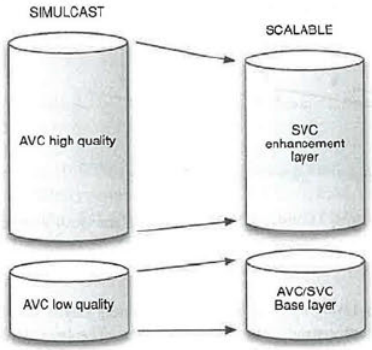


Figure 10.11 Simulcast vs. scalable bitrates

MPEG Technical Report N9577 [iv] compares the performance of SVC and AVC. This report describes a series of tests in which the same video clips were coded using (a) AVC and (b) SVC. The clips were coded at a number of quality levels and resolutions. In each case, a low-rate clip and a high-rate clip were produced. The low-rate clip was coded (a) using AVC and (b) as the base layer of a scalable stream. The high-rate clip was coded (a) as a separate AVC stream and (b) as the enhancement layer of a scalable stream. The operating bitrates were chosen such that the visual quality of the decoded sequences were approximately the same at each rate point, low and high. The perceived quality of each clip was measured by combining the opinion scores of a number of viewers (Mean Opinion Score, see Chapter 2).

Selected results from [iv] are shown in Figure 10.12, Figure 10.13 and Figure 10.14. Quality scalability with two layers is tested in Figure 10.12 for the sequences 'Ofer' and 'Paris'. The lower rate point represents a 30fps CIF sequence coded at a low bitrate. The SVC base layer rate and subjective quality, denoted as 'Quality of the Highest Layer', is slightly higher than the AVC low rate stream. The upper rate point represents the **total** bitrate of (a) both the AVC simulcast streams or (b) both the SVC layers. Note that the x-axis is **cumulative** bitrate. It is clear that in this case, SVC achieves the same quality as AVC with a lower combined bitrate.

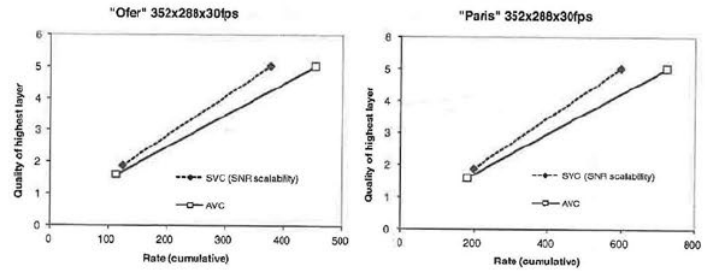


Figure 10.12 Quality scalability at CIF resolution

As illustrated in Figure 10.11, the total rate for two simulcast streams is higher than the total rate for two SVC layers.

Figure 10.13 shows similar results for combined spatial and temporal scalability. The lower rate point represents 15fps, 320 × 240 video and the upper rate point represents 30fps, 640 × 480 video, coded as (a) two simulcast streams or (b) base and enhancement SVC layers. Once again, the combined rate of the SVC layers is lower than the combined rate of the AVC streams, for the same or better visual quality. It is interesting to note that 'Crew' SVC base layer is actually given a very high quality score by the viewers. This may be due to the fact that the low-resolution sequence (320 × 240 × 15fps) has fewer distortion artefacts than the high-resolution sequence (640 × 480 × 30fps) and so is ranked approximately the same.

Finally, Figure 10.14 shows the results for the high-definition sequence 'AlohaWave' with spatial scalability or AVC simulcast delivering 720p / 50fps and 1080p / 50fps streams. Once again, SVC delivers the same or better quality at a lower combined rate than AVC. Note that subjective quality appears to **drop** as the rate and resolution increase. Again, this is probably due to the fact that more obvious coding distortion appears in the higher-resolution sequence.

The results presented in [iv] show a clear, if relatively modest, benefit from using SVC to deliver multiple versions of a sequence at different rate, quality and resolution points.

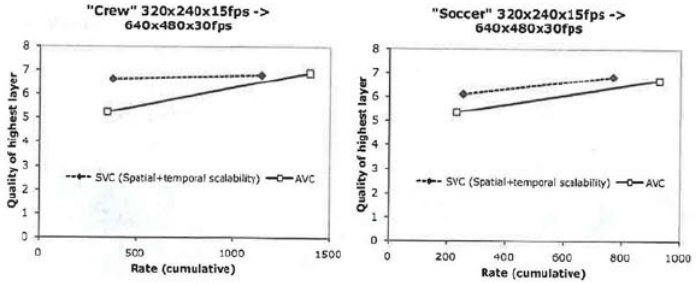


Figure 10.13 Spatial + temporal scalability, CIF → 4CIF resolution

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00034

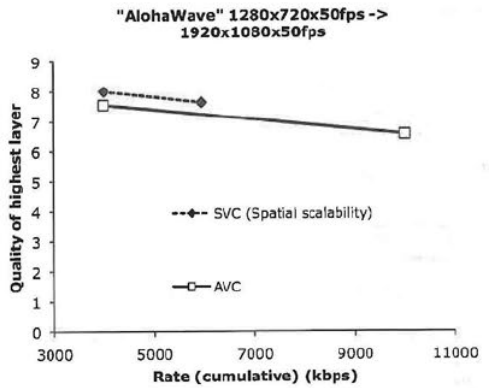


Figure 10.14 Spatial scalability, 720p → 1080p resolution

Despite the fact that scalable coding has been an active research topic since the early 1990s and has been incorporated into earlier standards such as MPEG-2 Video and MPEG-4 Visual, it has not been widely adopted by the video coding industry, perhaps because the cost and complexity outweighs the technical advantages. However, there are some indications that H.264 SVC is a more attractive commercial proposition, with recent product announcements in the videoconferencing market [v,vi].

10.3 Multiview Video Coding

Multiview video is video data that incorporates multiple concurrent versions of a particular scene. An example is shown in Figure 10.15; these three images are three simultaneous views of the same real-world scene, taken from different viewpoints. Potential implementations of multiview video include:

(a) Stereoscopic video. A stereo pair of views of the scene are combined, e.g. using data glasses or autostereoscopic displays, giving the illusion of a three-dimensional view, albeit with a limited viewing angle (Figure 10.16).
(b) 3D video. Multiple actual or rendered views of the scene are presented to the viewer e.g. using 'virtual reality' glasses or an advanced autostereoscopic display, so that the view



Figure 10.15 Three views of the same scene

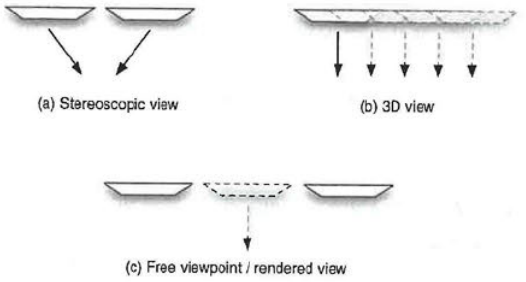


Figure 10.16 Multiview video : view examples

changes with head movements and the viewer has the feeling of 'immersion' in the 3D scene (Figure 10.16).
(c) Free-viewpoint video. A limited number of views of the scene are available, e.g. from multiple cameras at a sports game or multiple surveillance cameras. The viewer may select an arbitrary viewing angle. If this view does not exist, it is rendered or created from the available 'real' views (Figure 10.16).

Applications of multiview video include 3D television, advanced surveillance systems, immersive teleconferencing and gaming.

In a similar way to scalable video, multiview video content has inherent redundancy and the purpose of Multiview Video Coding (MVC) is to capitalise on this redundancy and efficiently code a multiview video scene. Figure 10.17 shows sequences of video frames corresponding

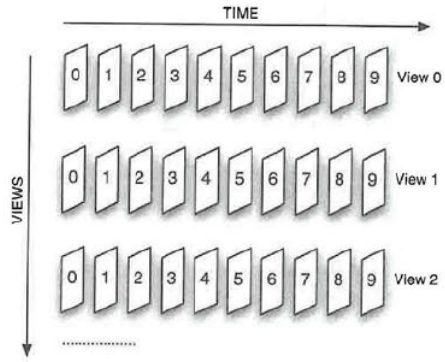


Figure 10.17 Multiview video : views and frames

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00035

to a series of views of a multiview scene. Each view consists of a series of frames or fields that may be coded as a separate H.264/AVC stream, i.e. simulcast coding of each view. However, there is likely to be a correlation between views, particularly if the camera positions are close together. Hence frame 0 of view 0 may be strongly correlated with frame 0 of view 1, if the camera positions are close; frame 0 of view 1 may be correlated with frame 0 of view 2; and so on.

10.3.1 H.264 Multiview Video Coding

The inherent redundancies in a multiview scene can be exploited by introducing predictions between views, i.e. inter-view prediction structures. This has required an extension to H.264/AVC, known as H.264 Multiview Video Coding (H.264 MVC). H.264 Multiview Video Coding is incorporated as Annex H into a draft revision of H.264/AVC [vii, viii]. Reference software is available at [ix].

An example of interview prediction is shown in Figure 10.18. View 0 (top) is predicted using a hierarchical GOP structure (Chapter 6) using conventional H.264/AVC tools. Each GOP consists of an I slice or 'key picture' followed by seven B slices. This means that View 0 can be decoded by an AVC or MVC decoder and can be considered as the Base layer or Base view. Each of the other views uses a similar prediction structure, except that the key pictures are now P slices, predicted from an I or P slice in the previous view. Inter-view correlation

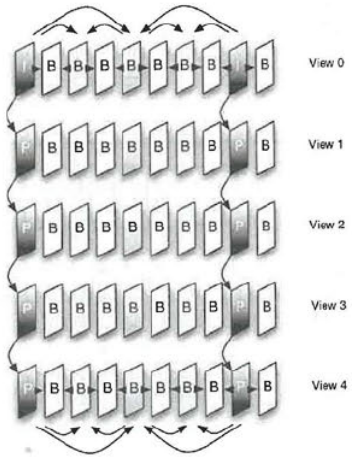


Figure 10.18 Interview prediction of key frames

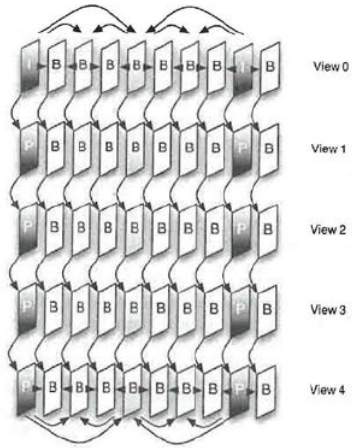


Figure 10.19 Inter-view prediction of all frames

means that the P slices in views 1, 2, 3 . . . are likely to be more efficiently coded than I slices in the same positions.

A more complex prediction structure includes interview prediction for every picture (Figure 10.19). Hence any B- or P-slice in views 1, 2, 3, . . . has a picture in another view available as a reference for prediction.

Annex H to H.264/AVC specifies a number of additions to the basic H.264 syntax to support MVC, including:

- Sequence Parameter Set: specify views and anchor or key picture references.
- Reference Picture List: structured to include support for inter-view prediction.
- NAL Unit order: modified to allow the use of a Prefix NALU, containing extra information about the Base view. This special Prefix NAL Unit may be discarded by an AVC decoder that is not MVC-compatible, so that the base view may still be decoded.
- Picture numbering and reference indices: modified to support multiple views.

MVC has to face the same challenge as SVC, namely, are the benefits of MVC worth the extra complexity compared with conventional simulcast coding of multiple views? Multiview video applications are still at a relatively early stage and there is not yet evidence that H.264 MVC will become popular in the industry as a coding technology. However, the popularity of stereoscopic ("3D") films such as "Avatar" is leading manufacturers to develop stereoscopic TV and Blu-Ray solutions.

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00036

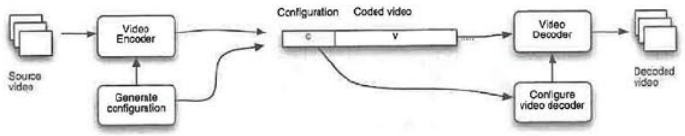


Figure 10.20 Overview of configurable video codec

10.4 Configurable Video Coding

H.264/AVC is a relatively recent addition to a series of popular video coding formats, starting with the early standards ITU-T Recommendation H.261 and ISO/IEC MPEG-1 and including an increasing number of standard and non-standard formats. Coding formats such as MPEG-2 Video, MPEG-4 Visual, VC-1 and H.264/AVC are widely used in current devices and systems. None of these formats or standards is inter-operable, i.e. video coded in one format can only be decoded by a decoder supporting the same format. As the digital video market continues to grow, this presents an increasing problem for device and codec manufacturers as they are faced with the need to support multiple, incompatible coding standards [x] leading to over-designed decoding products. The challenge of efficiently supporting multiple coding formats has led to increased interest in configurable video coding solutions. All of the current formats have certain aspects in common, such as block-based operation, inter-frame prediction and various forms of transform and entropy coding. Finding a way to exploit these commonalities may make it possible to efficiently and flexibly support multiple video coding formats.

The general concept of a configurable video codec is illustrated in Figure 10.20. At the encoder side, video is compressed using a particular configuration of video encoder, e.g. using a standard or non-standard coding format. Configuration information and coded video data are sent to the decoder. The configuration information is used to generate or configure a video decoding algorithm. The configured video decoder then proceeds to decompress the coded video sequence.

A configurable coding system has the potential to greatly increase the flexibility of a video codec, making it possible to re-configure the codec to handle multiple existing video formats, or 'upgrading' the decoder to handle a new coding format. Key questions for the design of such a system include:

- Should the configuration be completely flexible, or should it be limited to a number of pre-defined options?
- When should configuration occur – at the start of a communication session, during a communication session or both?
- Can a re-configurable codec achieve the same computational performance as a 'hard-wired' or fixed software or hardware codec?
- What impact does this approach have on issues such as inter-operability with existing codecs, random access to video streams and intellectual property questions such as license rights to coding algorithms?

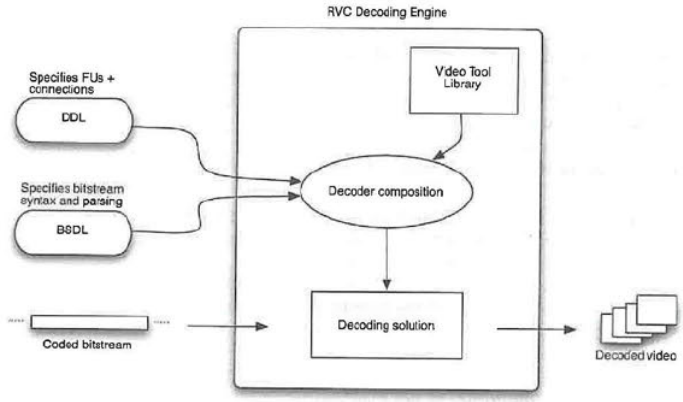


Figure 10.21 Reconfigurable Video Coding scenario

10.4.1 MPEG Reconfigurable Video Coding

MPEG's Reconfigurable Video Coding (RVC) initiative builds on concepts originally proposed over ten years ago [xi] and aims 'to provide a framework allowing a dynamic development, implementation and adoption of standardized video coding solutions with features of higher flexibility and reusability' [xii].

The RVC sub-group of MPEG has developed two standards that enable flexible re-configuration of video codecs to support a number of coding formats in an efficient way. In the RVC model, a decoder is specified as an interconnected set of Functional Units (FUs) which are decoding tools such as inverse transforms and entropy decoders. The available FUs are specified in the Video Tool Library (VTL) standard [xiii]. A particular decoder is defined by a Codec Configuration Representation [xiv], which describes a bit stream format and a set of FU interconnections and parameters. The Codec Configuration Representation is specified prior to starting a decoding session. Hence, an RVC video decoder is constructed from a set of pre-defined decoding tools. An existing or new video format can be supported by reconfiguring the RVC decoder, provided the format uses standard FUs from the Video Tool Library.

Figure 10.21 illustrates a typical RVC decoding scenario. In order to decode a video bitstream, the decoder needs to know (a) how to parse the bitstream and extract the coded data elements and (b) how to decode these elements. The RVC decoding engine receives BSDL and DDL specifications in compressed form. The decoder composition module generates a decoding solution, an actual video decoder, based on the BSDL and DDL specifications. It makes use of selected FUs from the Video Tool Library and connects these according to the DDL. Once the decoding solution has been generated, it can then decode the video bitstream.

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00037

The MPEG RVC approach has a number of potential benefits. A decoder can be modified to decode a different format by sending new BSDL/DDL descriptions, enabling efficient support for multiple coding formats. A non-standard coding format can be supported provided it uses FUs available to the decoder, i.e. FUs in the decoder's VTL. In practical terms this means that a new format should use FUs standardized by MPEG. A new coding tool can be proposed to MPEG for standardization as a new FU – a potentially faster, simpler route than creating a complete new video coding standard.

Two key constraints deliberately selected for the RVC model are:

1. A coding format must use combinations of coding tools (FUs) that are specified in the VTL. Introducing a new tool that is not in the VTL is likely to require a lengthy process of standardizing and disseminating a new version of the VTL.
2. The decoder configuration is fixed for the duration of a communication session, i.e. there is limited potential to change the coding algorithm if and when the characteristics of a video sequence change.

10.4.2 Fully Configurable Video Coding

Fully Configurable Video Coding (FCVC) has been proposed as an alternative to, or an evolution of, the RVC approach. FCVC differs from RVC in two ways:

1. The transmitted configuration information completely describes the video decoding algorithm. This means that (a) a library of pre-defined video coding tools is not necessary at the decoder and (b) any existing or new video coding algorithm can be configured.
2. Re-configuration can occur at any point during a video communication session. This makes it possible to adapt the video coding algorithm dynamically, for example changing aspects of the algorithm based on the statistics of the video sequence.

In the FCVC framework (Figure 10.22) a common decoding engine, the Universal Video Decoder (UVD), can be configured to decode any video sequence or syntax [xv]. In a typical application, the UVD initially has limited or no knowledge of the decoding methods required for a particular video bitstream. The encoder sends a set of configuration commands, the Decoder Description Syntax (DDS) which define the decoder design in terms of a set of primitive operations and interconnections. The UVD generates and connects new functional processing units according to these commands and can then proceed to decode the video bitstream. At a later point, the encoder may signal a change in configuration by sending new Decoder Description Syntax; the UVD implements the change and continues to decode the bitstream using the changed syntax or functionality. FCVC has the following benefits compared with conventional approaches:

(i) Any video decoding processes may be defined and created using a set of low-level configuration commands, or primitives for describing decoding operations. New algorithms can be implemented in decoders in a very short timescale which implies a short time-to-market.

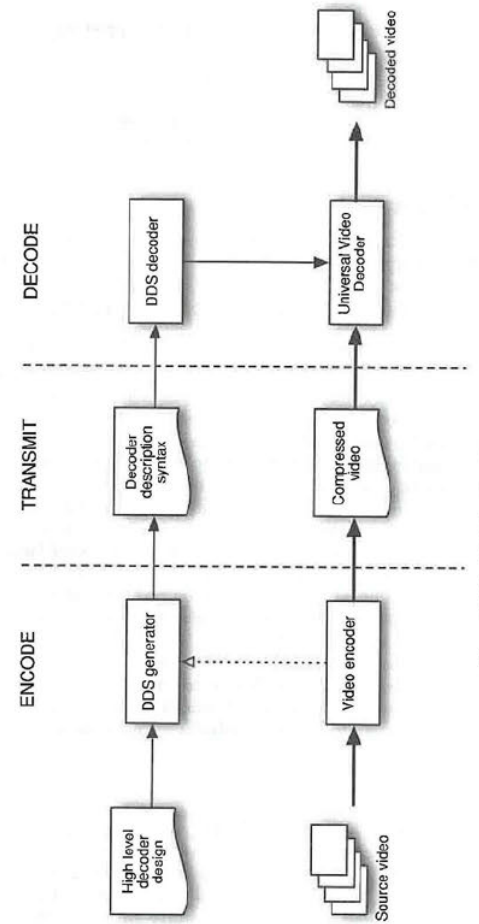


Figure 10.22 Fully Configurable Video Coding framework

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00038

(ii) Re-configuration can be carried out dynamically, enabling on-the-fly adaptation. This allows the codec to change its configuration to suit the video sequence, delivering optimal or near-optimal compression efficiency.
(iii) A single Universal Video Decoder can be re-configured to support any existing or new coded video format.
(iv) The Universal Video Decoder is programmed with only the tools it needs.

Recent efforts have focussed on developing and optimizing a prototype FCVC system [xvi] and on investigating the potential for combining aspects of FCVC and RVC [xvii].

10.5 Beyond H.264/AVC

First standardized in 2003, H.264/AVC is now a relatively mature technology. Other formats such as VC-1 and AVS can arguably offer similar performance but H.264 is certainly one of the leading formats in terms of compression efficiency at the time of writing (early 2010).

The Moving Picture Experts Group (MPEG) and Video Coding Experts Group (VCEG) are examining the need for a new video compression standard. Following a Call for Evidence [xviii] several proposals for improved video compression were presented at an MPEG meeting in July 2009. The consensus was that (a) there is likely to be a need for a new compression format, as consumers demand higher-quality video and as processing capacity improves and (b) there is potential to deliver better performance than the current state-of-the art. A number of different techniques were proposed, including decoder-side motion estimation, larger macroblock sizes (up to 32 × 32), more sophisticated in-loop deblocking filters, adaptive transform sizes and improved intra prediction. In general, all of these proposed algorithms offer the potential for better compression performance at the expense of increased computational complexity.

Results of subjective comparison tests of the new proposals led the committee to conclude that 'for a considerable number of test cases significant gain over AVC High Profile could be achieved' [xix]. For many of the tested video sequences, the proposed new algorithms led to improvements in subjective quality of one point or more on the MOS scale (Chapter 2). This implies that there is scope for a new coding format that significantly out-performs H.264/AVC.

The current plan is to set up a Joint Collaborative Team (JCT) of MPEG and VCEG representatives to work on a new video coding standard. Proposals for the new standard will be reviewed in 2010 and a new standard could be finalized around 2012/2013. It will aim to deliver significantly better compression performance than H.264/AVC, probably at a higher computational cost. A working title for the new standard is still under discussion.

10.6 Summary

H.264/AVC was always conceived as a toolkit of coding algorithms that could be extended to meet future needs. Recent extensions have included the Scalable Video Coding (SVC) and Multiview Video Coding (MVC) Annexes, which build on the basic standard to add support for efficient coding of multiple, correlated video streams. SVC is designed to accommodate multiple versions of the same video scene, at different spatial and temporal resolutions and at different bitrates, whereas MVC is intended to handle multiple closely-related views of a scene. For both SVC and MVC, better compression efficiency is achieved at the expense of increased complexity. Despite some market interest in SVC, it remains to be seen whether either of these extensions will be widely adopted.

Looking beyond H.264/AVC, two trends considered in this chapter are to (a) 'open up' video coding by allowing flexible re-configuration of coding tools and to (b) develop a new, higher-performance standard that improves compression efficiency. These trends are not necessarily mutually exclusive. There is likely to be a continued need for better compression efficiency, as video content becomes increasingly ubiquitous and places unprecedented pressure on restricted network connections. At the same time, the challenge of handling ever more diverse content coded in a wide variety of formats makes reconfigurable coding a potentially useful prospect.

10.7 References

i. H. Schwarz, D. Marpe and T. Wiegand, 'Overview of the scalable video coding extension of the H.264/AVC standard', IEEE Transactions on Circuits and Systems for Video Technology, September 2007.
ii. Recommendation ITU-T H.264 | ISO/IEC 14496-10:2009, 'Advanced Video Coding for generic audio-visual services', March 2009.
iii. Joint Scalable Video Model software, http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm
iv. ISO/IEC JTC 1/SC 29/WG 11 N9577, 'SVC Verification Test Report', Joint Video Team, Turkey, January 2007.
v. Vidyo press release, 'Vidyo unveils VidyoOne "Telework" HD video conferencing system', New Jersey, October 2009.
vi. Radvision press release, 'Radvision to bring scalable video coding technology to SCOPIA conferencing platform', Berlin / Tel Aviv, April 2009.
vii. Joint Video Team Document JVT-AD007, 'Editors' Draft Revision to ITU-T Rec. H.264 | ISO/IEC 14496-10 Advanced Video Coding', Geneva, February 2009.
viii. Y. Chen, K. Wang, K. Ugur, M. Hannuksela, J. Lainema and M. Gabbouj, 'The Emerging MVC Standard for 3D Video Services', *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article 786015, 2009.
ix. Joint Video Team Document JVT-AC207, 'WD 3 Reference Software for MVC', Busan, October 2008.
x. Fujitsu Press Release, 'Fujitsu Launches SD Multi-Decoder LSI Supporting MPEG-2 H.264', (Tokyo), November 2008.
xi. P.A. Chou, A. Eleftheriadis, C. Herpel, C. Reader, and J. Signes, 'The MPEG-4 Systems and Description Languages: A Way Ahead in Audio Visual Information Representation', *Signal Processing: Image Communication*, vol. 9, no. 4, pp. 385–431, May 1997.
xii. E. S. Jang, J. Ohm and M. Mattavelli, 'Whitepaper on Reconfigurable Video Coding (RVC)', ISO/IEC JTC1/SC29/WG11 document N9586, Antalya, January 2008.
xiii. ISO/IEC 23002-4 'Information technology – MPEG video technologies – Part 4: Video tool library', Final Draft International Standard, August 2009.
xiv. ISO/IEC 23001-4 'Information technology – MPEG systems technologies – Part 4: Codec configuration representation', Final Draft International Standard, August 2009.
xv. I. Richardson, M. Bystrom, S. Kannangara and M. de Frutos Lopez, 'Dynamic Configuration: Beyond Video Coding Standards', IEEE International System on Chip Conference, September 2008.
xvi. I. Richardson, C. S. Kannangara, M. Bystrom, J. Philp and M. De. Frutos-Lopez, 'A Framework for Fully Configurable Video Coding,' Proc. International Picture Coding Symposium 2009, Chicago, May 2009.
xvii. I.E. Richardson, C.S. Kannangara, M. Bystrom, J. Philp and Y. Zhao, 'Fully Configurable Video Coding – A Proposed Platform for Reconfigurable Video Coding', Document M16752, ISO/IEC JTC1/SC29/WG11 (MPEG), London, July 2009.
xviii. 'Call for Evidence on High Performance Video Coding', Document N10553, ISO/IEC JTC1/SC29/WG11 (MPEG), Maui, April 2009.
xix. 'Results of Call for Evidence on High Performance Video Coding', Document W10721, ISO/IEC JTC1/SC29/WG11 (MPEG), London, July 2009.

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00039

Index

Note: Page numbers followed by *f*, *n* and *t* indicate figures, notes, and tables, respectively.

1080i video format, 19*t*, 19*f*
1080p video format, 19, 19*t*, 227, 228, 229*t*, 301, 302*f*
3D video, 302
3G, 2, 3
4:2:0 sampling, 15*f*, 16, 17*f*, 31*f*, 122, 125*f*, 126*f*, 156, 182, 184*f*, 203, 225*f*, 226*f*, 227*f*, 257*f*
4:2:2 sampling, 15, 15*f*, 16, 17, 124, 182, 184*f*, 204, 225, 226*f*, 227*f*
4:4:4 sampling, 14*f*, 15*f*, 16, 124, 185, 205, 206, 225, 226*f*, 227*f*
4CIF video format, 16, 17*t*, 301*f*
720p video format, 19, 19*f*, 19*t*, 93, 228, 229*t*, 301, 302*f*

access unit, 99, 100, 101, 102*t*, 114*t*, 116, 230, 230*f*, 231, 231*t*
Arbitrary Slice Order, 237, 238, 238*f*
arithmetic coding, 65, 67*f*, 68, 89, 179, 208, 217, 220, 253

B-skip, 165, 279
Base Layer, 289, 290, 291, 291*f*, 292, 293, 294, 297, 298, 299, 300
Baseline Profile, 93, 99*t*, 115, 115*t*, 141, 141*f*, 169, 210, 224, 261, 263, 263*f*, 267, 273, 278
basis patterns, 44, 45*f*, 87, 88*f*, 90, 90*f*
binarize, 217, 218
bipredíction, 116*t*, 117*t*, 124, 162, 163*f*
Blu-Ray, 4, 98

CABAC (Context Adaptive Binary Arithmetic Coding), 92, 103*t*, 118, 124, 127*f*, 181, 208, 217, 218, 218*f*, 220, 224, 225*f*, 226*f*, 227*f*, 260*t*, 261, 265, 267*t*, 268*t*, 269*f*, 271, 273
CAVLC (Context Adaptive Variable Length Coding), 92, 103*t*, 116*t*, 118*t*, 124, 126*f*, 179, 206, 208, 210, 210*t*, 211, 211*f*, 220, 224, 225*f*, 226*f*, 227*f*, 260, 265, 266*t*, 267*t*, 268 *t*, 273
CCD (Charge Coupled Device), 8, 9
CIF video format, 16, 17*t*, 70, 93, 141, 141*f*, 165, 228, 229*t*, 261, 264, 270, 270*f*, 271, 271*f*, 291, 300, 301*f*
Coarse Grain Quality Scalability, 298
Coded Block Pattern, 96, 100*f*, 101, 103*t*, 120, 119*f*, 120*t*, 121*t*, 124, 127, 210, 212*n*, 264
Coded Picture Buffer, 84, 84*f*, 230, 230*f*, 231, 232*f*, 233, 234*f*, 236, 275*f*
colour plane coding, 206, 226*f*, 227*f*
configuration file, 257, 258*f*, 259, 260*t*, 261
conformance with H.264 standard, 6, 223–253, 235, 236, 236*f*, 237, 237*f*, 261

© 2010 John Wiley & Sons, Ltd

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00040

Constrained Baseline Profile, 93, 169, 224
context models, 219, 219*t*, 220, 220*t*, 260

data partitioned slices, 114, 237, 243–244
DC transform, 181, 182, 203–204
Decoded Picture Buffer, 86, 101, 103*f*, 104*f*, 106, 108*f*, 111, 149, 150, 151, 162, 162*f*, 226, 230, 230*f*, 275*f*, 294
Decoder Description Syntax, 308
Differential Pulse Code Modulation, 41
Direct Mode, 122, 149, 151, 161–162, 260*t*
Discrete Cosine Transform, 30, 43, 87, 180, 181, 185
Discrete Wavelet Transform, 48s
Double Stimulus Continuous Quality Scale, 20
DVD (Digital Versatile Disk), 1, 2, 3, 3*f*, 4, 5, 16, 19, 81, 83, 97, 98, 224, 274*t*

Enhancement Layer, 289, 290, 291, 292, 294, 297, 298, 299, 300
entropy encoder, 27, 56, 57–68
Exp-Golomb codes, 208

field scan, 113, 114, 207*f*, 211
Flexible Macroblock Ordering, 238
frame / field prediction, 164
free-viewpoint video, 303
frequency dependent quantization, 204–205, 204*f*
Full Reference, 23
Fully Configurable Video Coding, 308–310, 309*f*

Group of Pictures, 169–170, 169*f*, 275, 293

H.263 standard, 26, 31, 68
H.264/AVC standard, 81, 82, 91, 92*t*, 99, 99*t*, 180, 207, 224, 252, 290, 293
Hadamard Transform, 181, 182, 203, 220, 284
hierarchical prediction, 170, 273, 293*f*
High Definition, 1, 2, 5, 18–19, 19*f*, 93, 98, 223, 224, 225, 227, 228, 287, 290, 301
High Profile, 119f, 120, 124, 139*t*, 148, 181, 182, 204, 217, 224, 225, 226f, 237, 248, 287, 293, 310
Huffman coding, 58–62, 62*t*, 63, 65, 68
Human Visual System, 13, 20, 204
Hypothetical Reference Decoder, 92*t*, 223, 230, 230*f*, 249*t*, 250*t*, 253

Instantaneous Decoder Refresh, 101, 114
integer transform, 87, 120, 179, 179, 181, 185, 198–199, 199*t*, 200*t*, 211, 224
inter prediction, 6, 27, 40, 85, 85*n*, 87*f*, 91, 101, 120, 122, 132, 137, 149–170, 167*f*, 168*f*, 182, 182, 225, 264, 265, 271, 276, 297
interlaced video, 11, 11*f*, 16, 19*t*, 111, 207, 224, 260*t*
International Standards Organisation, 4
International Telecommunications Union, 4
Intra PCM, 121–122
intra prediction, 26, 27, 32, 38, 39*f*, 85, 86*f*, 91, 95, 103*t*, 116*t*, 117*t*, 120*t*, 137, 138–149, 171*n*, 181, 185, 203, 206, 224, 243, 264, 276, 297, 310
ISO Media File Format, 247
ITU-R 601, 7

Joint Model reference software, 255, 256, 285

Lagrange, 281
licensing of video coding standards, 4, 223, 248–253
loop filter, 171–176, 260*t*, 265, 266*t*, 267*t*, 268*t*
lossless predictive coding, 205–206, 226*f*, 227*f*

macroblock, 31–32, 31*f*, 57–58, 69, 70, 74*f*, 75*f*, 77, 84, 85, 86*f*, 87*f*, 91, 94, 94*f*, 95, 96*f*, 101, 103*t*, 111, 112, 113*f*, 117, 117*t*, 118, 119–135, 137–138, 142*f*, 146*f*, 149, 158, 159, 160–167, 171*n*,

172f, 173*f*, 181, 184*f*, 210*t*, 227, 228, 238, 239*t*, 239*f*, 264, 274*f*, 276, 279, 280, 282, 284
Macroblock Adaptive Frame Field Coding, 112, 113*f*, 164, 260*t*
macroblock partition, 117t, 130, 132, 149, 157, 158f, 158t, 210t, 264, 279, 282, 297
Main Profile, 93, 115*t*, 224, 225*f*, 226, 261, 264, 264*f*, 268, 273
Medium Grain Quality Scalability, 298, 298*f*
mode selection, 166, 255, 256, 260*t*, 261, 279, 281–283, 284, 285
motion compensated prediction, 27, 31–32, 35, 69, 70, 103*t*, 111, 162–164, 162*f*, 165, 224, 241, 243, 247, 249*t*, 252, 264, 281
motion compensation, 26, 30, 31, 32, 33*f*, 35–38, 41, 72*f*, 85*n*, 155, 159, 165, 175, 266*t*, 267*t*, 268*t*
motion estimation, 26, 30–32, 32*f*, 35, 36*f*, 69, 72, 85*n*, 260*t*, 261
motion vector prediction, 60*f*, 158–161, 297
Moving Picture Experts Group (MPEG), 5, 310
MPEG-2 standard, 26, 31, 68, 83, 99, 97*f*, 169, 180, 246, 246*f*, 250, 251, 252, 302, 306
MPEG-LA, 252, 253
multiple reference frames, 130, 131*t*, 170, 267
Multiview Video Coding, 287, 302–305

Network Abstraction Layer, 100, 100*f*, 102*t*, 114, 244, 245*f*
No Reference, 23
NTSC, 17, 250*t*

PAL, 12, 17, 250*t*
Peak Signal to Noise Ratio, 21
Picture Adaptive Frame Field Coding, 142*f*
Picture Order Count, 104, 118, 118*t*
Picture Parameter Set, 100, 102*t*, 114*t*, 115, 116*t*, 116*f*, 205, 244
progressive video, 12, 111

QCIF video format, 16, 17*t*, 37*t*, 53, 115, 115*t*, 141, 142*f*, 227, 228, 229*t*, 239*f*, 259, 265, 266*f*, 267*f*, 268*f*, 269*f*, 270*f*, 271, 273, 273*f*, 278, 279*f*
Quality Scalability, 290, 291, 291*f*, 294, 295*f*, 298–299, 299*f*, 300, 301*f*
quantization, 26, 50–52, 53*f*, 54, 58, 78, 87, 88f, 89*n*, 90, 96, 102*t*, 174, 179–206, 260*t*
quantizer step size, 58, 76, 191, 192*f*, 198, 202, 204, 277

rate control, 247, 255, 260*t*, 265*t*, 266*t*, 267*t*, 268, 268*t*, 269*f*, 274–278
Rate Distortion Optimization, 265, 266, 267, 268*t*
Raw Byte Sequence Payload, 101, 102*t*, 244, 245*f*, 248
Real-Time Protocol, 246
Reconfigurable Video Coding, 287, 307–308, 307*f*
reconstruction, 48, 69, 70, 83, 91, 91*f*
Reduced Reference, 23
redundant slices, 224, 225*f*, 237, 260*t*
rescaling matrix, 189, 200–202
residual, 27, 28, 30, 32, 33*f*, 34*f*, 35–38, 40–43, 49, 53, 54, 54*f*, 69, 70, 72*f*, 74*f*, 75*f*, 78*f*, 84, 84*f*, 85, 86*f*, 87, 87*f*, 90, 96*f*, 97*f*, 101, 103*t*, 121, 124, 125*f*, 126*f*, 127*f*, 141, 143*f*, 165, 180, 193, 210, 243, 244, 250, 281, 297
RGB colour space, 12–13, 14
run-level encoding, 56

Scalable Video Coding, 92*t*, 247, 287, 288–302
scalar quantization, 50
Sequence Parameter Set, 93, 100, 101, 102*t*, 105, 111, 112, 114*t*, 115, 115*t*, 206, 226, 244, 248, 262*t*
SI-slice, 243
simulcast, 288–289, 299, 300, 300*f*, 301
SP-slice, 241, 242*t*, 242*f*, 243, 243*f*, 244*f*
spatial sampling, 9

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00041

Spatial Scalability, 290, 291*f*, 294, 296*f*, 298, 301, 302*f*
SSIM, 23
stereoscopic video, 287, 302
stream analyzer, 263, 263*f*, 264*f*
sub-macroblock partition, 122, 123, 149, 150, 157, 158*f*, 158*t*, 159, 162, 166
sub-pixel motion compensation, 35–38
sub-pixel prediction, 152*f*
Sum of Absolute Errors, 36, 145
Sum of Absolute Transformed Differences, 282, 284
Sum of Squared Distortion, 282
Supplemental Enhancement Information, 92*t*, 102*t*, 114*t*, 248

temporal sampling, 8*f*, 9–11, 26
Temporal Scalability, 170, 290, 291*f*, 292, 292*t*, 293, 301, 301*f*
trailing bits, 101, 102*t*, 245*f*

Universal Video Decoder, 308, 310

VC-1 standard, 26, 68, 180, 306, 310
vector quantization, 52, 53*f*
video coding, 4, 7, 25–79, 81, 82, 82*f*, 83*f*, 89, 92*t*, 223, 247, 248, 252, 253, 262, 268, 287, 288–310
Video Coding Experts Group, 5, 310
Video Coding Layer, 100, 114
video coding patents, 250–251, 252
video compression, 2–6, 16, 21, 24, 25, 43, 51, 81, 83, 85, 223, 237, 248, 250, 251, 251*f*, 252, 287, 288, 310
Video Quality Experts Group, 23
Video Usability Information, 94*t*, 225, 248, 250*t*
videoconferencing, 16, 81, 82, 169, 247, 274*t*, 302
weighted prediction, 116*t*, 137, 149, 163–164, 260*t*

x264 software, 261, 261*n*, 262, 262*f*, 263

YCrCb colour space, 13–16
Youtube, 5

zig zag scan, 54, 112

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00042

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.

ISBN 978-0-470-51692-8

WILEY
wiley.com

IPR2025-00177
Patent Owner's Exhibit No. 2003
Ex. 2003.00043