
**Information technology — Coding of
audio-visual objects —**

**Part 2:
Visual**

*Technologies de l'information — Codage des objets audiovisuels —
Partie 2: Codage visuel*

Reference number
ISO/IEC 14496-2:2001(E)



PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2001

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

1	Scope	1
2	Normative references	1
3	Terms and definitions	2
4	Abbreviations and symbols	12
4.1	Arithmetic operators	12
4.2	Logical operators	13
4.3	Relational operators	13
4.4	Bitwise operators	13
4.5	Conditional operators	13
4.6	Assignment	13
4.7	Mnemonics	14
4.8	Constants	14
5	Conventions	14
5.1	Method of describing bitstream syntax	14
5.2	Definition of functions	15
5.2.1	Definition of next_bits() function	15
5.2.2	Definition of bytealigned() function	15
5.2.3	Definition of nextbits_bytealigned() function	15
5.2.4	Definition of next_start_code() function	16
5.2.5	Definition of next_resync_marker() function	16
5.2.6	Definition of transparent_mb() function	16
5.2.7	Definition of transparent_block() function	16
5.2.8	Definition of byte_align_for_upstream() function	16
5.3	Reserved, forbidden and marker_bit	16
5.4	Arithmetic precision	17
6	Visual bitstream syntax and semantics	17
6.1	Structure of coded visual data	17
6.1.1	Visual object sequence	18
6.1.2	Visual object	18
6.1.3	Video object	18
6.1.4	Mesh object	24
6.1.5	FBA object	25
6.1.6	3D Mesh Object	29
6.2	Visual bitstream syntax	30
6.2.1	Start codes	30
6.2.2	Visual Object Sequence and Visual Object	34
6.2.3	Video Object Layer	35
6.2.4	Group of Video Object Plane	40
6.2.5	Video Object Plane and Video Plane with Short Header	40
6.2.6	Macroblock	53
6.2.7	Block	58
6.2.8	Still Texture Object	59
6.2.9	Mesh Object	73
6.2.10	FBA Object	75
6.2.11	3D Mesh Object	85
6.2.12	Upstream message	103
6.3	Visual bitstream semantics	104
6.3.1	Semantic rules for higher syntactic structures	104
6.3.2	Visual Object Sequence and Visual Object	104
6.3.3	Video Object Layer	109

6.3.4	Group of Video Object Plane	120
6.3.5	Video Object Plane and Video Plane with Short Header	120
6.3.6	Macroblock related	131
6.3.7	Block related	134
6.3.8	Still texture object.....	135
6.3.9	Mesh object	142
6.3.10	FBA object	144
6.3.11	3D Mesh Object.....	151
6.3.12	Upstream message	162
7	The visual decoding process	164
7.1	Video decoding process	165
7.2	Higher syntactic structures	166
7.3	VOP reconstruction	166
7.4	Texture decoding	166
7.4.1	Variable length decoding	167
7.4.2	Inverse scan	168
7.4.3	DC and AC prediction for intra macroblocks.....	169
7.4.4	Inverse quantisation	172
7.4.5	Inverse DCT	175
7.4.6	Upsampling of the Inverse DCT output for Reduced Resolution VOP.....	176
7.5	Shape decoding	177
7.5.1	Higher syntactic structures	177
7.5.2	Macroblock decoding.....	178
7.5.3	Arithmetic decoding	187
7.5.4	Spatial scalable binary shape decoding.....	189
7.5.5	Grayscale Shape Decoding	198
7.5.6	Multiple Auxiliary Component Decoding	201
7.6	Motion compensation decoding.....	201
7.6.1	Padding process	201
7.6.2	Sample interpolation for non-integer motion vectors.....	205
7.6.3	General motion vector decoding process.....	207
7.6.4	Unrestricted motion compensation	209
7.6.5	Vector decoding processing and motion-compensation in progressive P- and S(GMC)-VOP	210
7.6.6	Overlapped motion compensation.....	212
7.6.7	Temporal prediction structure.....	213
7.6.8	Vector decoding process of non-scalable progressive B-VOPs	214
7.6.9	Motion compensation in non-scalable progressive B-VOPs	214
7.6.10	Motion Compensation Decoding of Reduced Resolution VOP	219
7.7	Interlaced video decoding	224
7.7.1	Field DCT and DC and AC Prediction	224
7.7.2	Motion compensation.....	225
7.8	Sprite decoding.....	234
7.8.1	Higher syntactic structures	234
7.8.2	Sprite Reconstruction	235
7.8.3	Low-latency sprite reconstruction	235
7.8.4	Sprite reference point decoding.....	236
7.8.5	Warping.....	237
7.8.6	Sample reconstruction	239
7.8.7	GMC decoding.....	240
7.9	Generalized scalable decoding	241
7.9.1	Temporal scalability	241
7.9.2	Spatial scalability.....	246
7.10	Still texture object decoding.....	251
7.10.1	Decoding of the DC subband	251
7.10.2	ZeroTree Decoding of the Higher Bands.....	252
7.10.3	Inverse Quantisation	257
7.10.4	Still Texture Error Resilience.....	265
7.10.5	Wavelet Tiling.....	268
7.10.6	Scalable binary shape object decoding	270

7.11	Mesh object decoding	276
7.11.1	Mesh geometry decoding	276
7.11.2	Decoding of mesh motion vectors	279
7.12	FBA object decoding	281
7.12.1	Frame based face object decoding	281
7.12.2	DCT based face object decoding	282
7.12.3	Decoding of the viseme parameter fap 1	284
7.12.4	Decoding of the viseme parameter fap 2	284
7.12.5	Fap masking	285
7.12.6	Frame Based Body Decoding	285
7.12.7	DCT based body object decoding	286
7.13	3D Mesh Object Decoding	287
7.13.1	Start codes and bit stuffing	288
7.13.2	The Topological Surgery decoding process	288
7.13.3	The Forest Split decoding process	291
7.13.4	Header decoder	292
7.13.5	partition type	293
7.13.6	Vertex Graph Decoder	294
7.13.7	Triangle Tree Decoder	298
7.13.8	Triangle Data Decoder	299
7.13.9	Forest Split decoder	303
7.13.10	Arithmetic decoder	309
7.14	NEWPRED mode decoding	314
7.14.1	Decoder Definition	314
7.14.2	Upstream message	314
7.15	Output of the decoding process	314
7.15.1	Video data	315
7.15.2	2D Mesh data	315
7.15.3	Face animation parameter data	315
8	Visual-Systems Composition Issues	315
8.1	Temporal Scalability Composition	315
8.2	Sprite Composition	316
8.3	Mesh Object Composition	317
8.4	Spatial Scalability composition	318
9	Profiles and Levels	318
9.1	Visual Object Types	318
9.2	Visual Profiles	321
9.3	Visual Profiles@Levels	322
9.3.1	Natural Visual	322
9.3.2	Synthetic Visual	322
9.3.3	Synthetic/Natural Hybrid Visual	324
Annex A	(normative) Coding transforms	326
A.1	Discrete cosine transform for video texture	326
A.2	Discrete wavelet transform for still texture	327
A.2.1	Adding the mean	327
A.2.2	Wavelet filter	327
A.2.3	Symmetric extension	328
A.2.4	Decomposition level	329
A.2.5	Shape adaptive wavelet filtering and symmetric extension	329
A.3	Shape-Adaptive DCT (SA-DCT)	330
A.3.1	Definition of Forward SA-DCT	330
A.3.2	Definition of Inverse SA-DCT	332
A.4	SA-DCT with DC Separation and Δ DC Correction (Δ DC-SA-DCT)	333
A.4.1	Definition of Forward Δ DC-SA-DCT	334
A.4.2	Definition of Inverse Δ DC-SA-DCT	334
Annex B	(normative) Variable length codes and arithmetic decoding	336
B.1	Variable length codes	336
B.1.1	Macroblock type	336

B.1.2	Macroblock pattern.....	338
B.1.3	Motion vector	340
B.1.4	DCT coefficients.....	342
B.1.5	Shape Coding.....	352
B.1.6	Sprite Coding	357
B.1.7	DCT based facial object decoding	358
B.1.8	Shape decoding for still texture object	367
B.2	Arithmetic Decoding.....	368
B.2.1	Aritmetic decoding for still texture object	368
B.2.2	Arithmetic decoding for shape decoding.....	371
B.2.3	FBA Object Decoding.....	374
Annex C	(normative) Face and body object decoding tables and definitions.....	376
Annex D	(normative) Video buffering verifier.....	409
D.1	Introduction	409
D.2	Video Rate Buffer Model Definition.....	409
D.3	Comparison between ISO/IEC 14496-2 VBV and the ISO/IEC 13818-2 VBV (Informative).....	412
D.4	Video Complexity Model Definition.....	413
D.5	Video Reference Memory Model Definition.....	415
D.6	Interaction between VBV, VCV and VMV (informative).....	416
D.7	Video Presentation Model Definition (informative)	416
Annex E	(informative) Features supported by the algorithm	418
E.1	Error resilience.....	418
E.1.1	Resynchronization.....	418
E.1.2	Data Partitioning	419
E.1.3	Reversible VLC.....	419
E.1.4	Decoder Operation.....	420
E.1.5	Adaptive Intra Refresh (AIR) Method	423
E.1.6	NEWPRED.....	425
E.2	Complexity Estimation	427
E.3	Resynchronization in Case of Unknown Video Header Format.....	427
Annex F	(informative) Preprocessing and postprocessing.....	428
F.1	VOP Generation Tools: Automatic and Semi-automatic Segmentations	428
F.1.1	Automatic Segmentation	428
F.1.2	Semi-automatic Segmentation	438
F.1.3	References.....	446
F.2	Bounding Rectangle of VOP Formation	447
F.3	Postprocessing for Coding Noise Reduction	448
F.3.1	Deblocking filter.....	448
F.3.2	Deringing filter	450
F.3.3	Further issues	452
F.4	Chrominance Decimation and Interpolation Filtering for Interlaced Object Coding	452
Annex G	(normative) Profile and level indication and restrictions	454
Annex H	(informative) Patent statements	457
H.1	Patent statements for ISO/IEC 14496 Version 1	457
H.2	Patent statements for the extensions provided in ISO/IEC 14496 Version 2	458
Annex I	(informative) Encoder Complexity Reduction Based on Intelligent Pre-Quantisation	460
I.1	Introduction	460
I.2	Feature Selection and Pre-quantisation.....	460
I.3	Model Verification and Threshold Setting.....	462
I.3.1	H.263 Quantiser.....	462
I.3.2	MPEG-4 Quantiser	462
Annex J	(normative) View dependent object scalability	464
J.1	Introduction	464
J.2	Decoding Process of a View-Dependent Object.....	464
J.2.1	General Decoding Scheme	464
J.2.2	Computation of the View-Dependent Scalability parameters	466

J.2.3	VD mask computation	468
J.2.4	Differential mask computation	469
J.2.5	DCT coefficients decoding	469
J.2.6	Texture update	469
J.2.7	IDCT	470
Annex K	(normative) Decoder Configuration Information	471
K.1	Introduction	471
K.2	Description of the set up of a visual decoder (informative)	471
K.2.1	Processing of decoder configuration information	472
K.3	Specification of decoder configuration information	473
K.3.1	VideoObject	473
K.3.2	StillTextureObject	473
K.3.3	MeshObject	474
K.3.4	FaceObject	474
K.3.5	3DMeshObject	474
Annex L	(informative) Rate control	475
L.1	Frame Rate Control	475
L.1.1	Introduction	475
L.1.2	Description	475
L.1.3	Summary	479
L.2	Multiple Video Object Rate Control	479
L.2.1	Initialization	479
L.2.2	Quantisation Level Calculation for I-frame and first P-frame	479
L.2.3	Update Rate-Distortion Model	482
L.2.4	Post-Frameskip Control	482
L.3	Macroblock Rate Control	484
L.3.1	Rate-Distortion Model	484
L.3.2	Target Number of Bits for Each Macroblock	485
L.3.3	Macroblock Rate Control	485
Annex M	(informative) Binary shape coding	487
M.1	Introduction	487
M.2	Context-Based Arithmetic Shape Coding	487
M.2.1	Intra Mode	488
M.2.2	Inter Mode	488
M.3	Texture Coding of Boundary Blocks	489
M.4	Encoder Architecture	489
M.5	Encoding Guidelines	490
M.5.1	Lossy Shape Coding	491
M.5.2	Coding Mode Selection	491
M.6	Conclusions	491
M.7	References	492
Annex N	(normative) Visual profiles@levels	493
Annex O	(informative) 3D Mesh Coding	497
O.1	Introduction	497
O.2	Topological Surgery Representation	497
O.2.1	Simple Polygon Representation	498
O.2.2	Vertex Graph representation	499
O.3	Encoding guidelines for 3D Mesh Coding	500
O.3.1	Topological Surgery Encoding	500
O.3.2	Support for non-manifolds and Non-orientable manifolds	501
O.3.3	Support for Error Resilience	503
O.4	Encoder considerations for efficient compression of Vertex Properties	507
O.5	Progressive Forest Split Representation	508
O.5.1	Encoding the Forest	508
O.5.2	Support for meshes with polygonal faces	509
O.5.3	Method for generating of a PFS Representation of a Triangular 3D Mesh	509
O.5.4	Topological Tests	510

O.5.5 Geometric Tests.....512

O.6 Complexity estimation for Computational Graceful Degradation 512

O.7 QoS for SNHC through upstream.....514

O.8 References.....516

Bibliography 517

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 14496 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14496-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC 14496-2:1999), which has been technically revised. It incorporates Cor.1:2000, Cor.2:2001 and Amd.1:2000.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

- *Part 1: Systems*
- *Part 2: Visual*
- *Part 3: Audio*
- *Part 4: Conformance testing*
- *Part 5: Reference software*
- *Part 6: Delivery Multimedia Integration Framework (DMIF)*
- *Part 7: Optimized software for MPEG-4 visual tools*
- *Part 8: Carriage of MPEG-4 contents over IP networks*

Annexes A to D, G, J, K and N form a normative part of this part of ISO/IEC 14496. Annexes E, F, H, I, L, M and O are for information only.

Introduction

Purpose

This part of ISO/IEC 14496 was developed in response to the growing need for a coding method that can facilitate access to visual objects in natural and synthetic moving pictures and associated natural or synthetic sound for various applications such as digital storage media, internet, various forms of wired or wireless communication etc. The use of ISO/IEC 14496 means that motion video can be manipulated as a form of computer data and can be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcast channels.

Application

The applications of ISO/IEC 14496 cover, but are not limited to, such areas as listed below:

IMM	Internet Multimedia
IVG	Interactive Video Games
IPC	Interpersonal Communications (videoconferencing, videophone, etc.)
ISM	Interactive Storage Media (optical disks, etc.)
MMM	Multimedia Mailing
NDB	Networked Database Services (via ATM, etc.)
RES	Remote Emergency Systems
RVS	Remote Video Surveillance
WMM	Wireless Multimedia
	Multimedia

Profiles and levels

ISO/IEC 14496 is intended to be generic in the sense that it serves a wide range of applications, bitrates, resolutions, qualities and services. Furthermore, it allows a number of modes of coding of both natural and synthetic video in a manner facilitating access to individual objects in images or video, referred to as content based access. Applications should cover, among other things, digital storage media, content based image and video databases, internet video, interpersonal video communications, wireless video etc. In the course of creating ISO/IEC 14496, various requirements from typical applications have been considered, necessary algorithmic elements have been developed, and they have been integrated into a single syntax. Hence ISO/IEC 14496 will facilitate the bitstream interchange among different applications.

This part of ISO/IEC 14496 includes one or more complete decoding algorithms as well as a set of decoding tools. Moreover, the various tools of this part of ISO/IEC 14496 as well as that derived from ISO/IEC 13818-2:1996 can be combined to form other decoding algorithms. Considering the practicality of implementing the full syntax of this part of ISO/IEC 14496, however, a limited number of subsets of the syntax are also stipulated by means of “profile” and “level”.

A “profile” is a defined subset of the entire bitstream syntax that is defined by this part of ISO/IEC 14496. Within the bounds imposed by the syntax of a given profile it is still possible to require a very large variation in the performance of encoders and decoders depending upon the values taken by parameters in the bitstream.

In order to deal with this problem “levels” are defined within each profile. A level is a defined set of constraints imposed on parameters in the bitstream. These constraints may be simple limits on numbers. Alternatively they may take the form of constraints on arithmetic combinations of the parameters.

Object based coding syntax

Video object

A *video object* in a scene is an entity that a user is allowed to access (seek, browse) and manipulate (cut and paste). The instances of video objects at a given time are called *video object planes* (VOPs). The encoding process generates a coded representation of a VOP as well as composition information necessary for display. Further, at the decoder, a user may interact with and modify the composition process as needed.

The full syntax allows coding of rectangular as well as arbitrarily shaped video objects in a scene. Furthermore, the syntax supports both non-scalable coding and scalable coding. Thus it becomes possible to handle normal scalabilities as well as object based scalabilities. The scalability syntax enables the reconstruction of useful video from pieces of a total bitstream. This is achieved by structuring the total bitstream in two or more layers, starting from a standalone base layer and adding a number of enhancement layers. The base layer can be coded using a non-scalable syntax, or in the case of picture based coding, even using a syntax of a different video coding standard.

To ensure the ability to access individual objects, it is necessary to achieve a coded representation of its shape. A natural video object consists of a sequence of 2D representations (at different points in time) referred to here as VOPs. For efficient coding of VOPs, both temporal redundancies as well as spatial redundancies are exploited. Thus a coded representation of a VOP includes representation of its shape, its motion and its texture.

FBA object

The FBA object is a collection of nodes in a scene graph which are animated by the FBA (Face and Body Animation) object bitstream. The FBA object is controlled by two separate bitstreams. The first bitstream, called BIFS, contains instances of Body Definition Parameters (BDPs) in addition to Facial Definition Parameters (FDPs), and the second bitstream, FBA bitstream, contains Body Animation Parameters (BAPs) together with Facial Animation Parameters (FAPs).

A 3D (or 2D) *face object* is a representation of the human face that is structured for portraying the visual manifestations of speech and facial expressions adequate to achieve visual speech intelligibility and the recognition of the mood of the speaker. A face object is animated by a stream of *face animation parameters* (FAP) encoded for low-bandwidth transmission in broadcast (one-to-many) or dedicated interactive (point-to-point) communications. The FAPs manipulate key feature control points in a mesh model of the face to produce animated visemes for the mouth (lips, tongue, teeth), as well as animation of the head and facial features like the eyes. FAPs are quantized with careful consideration for the limited movements of facial features, and then prediction errors are calculated and coded arithmetically. The remote manipulation of a face model in a terminal with FAPs can accomplish lifelike visual scenes of the speaker in real-time without sending pictorial or video details of face imagery every frame.

A simple streaming connection can be made to a decoding terminal that animates a default face model. A more complex session can initialize a custom face in a more capable terminal by downloading *face definition parameters* (FDP) from the encoder. Thus specific background images, facial textures, and head geometry can be portrayed. The composition of specific backgrounds, face 2D/3D meshes, texture attribution of the mesh, etc. is described in ISO/IEC 14496-1:1999. The FAP stream for a given user can be generated at the user's terminal from video/audio, or from text-to-speech. FAPs can be encoded at bitrates up to 2-3kbit/s at necessary speech rates. Optional temporal DCT coding provides further compression efficiency in exchange for delay. Using the facilities of ISO/IEC 14496-1:1999, a composition of the animated face model and synchronized, coded speech audio (low-bitrate speech coder or text-to-speech) can provide an integrated low-bandwidth audio/visual speaker for broadcast applications or interactive conversation.

Limited scalability is supported. Face animation achieves its efficiency by employing very concise motion animation controls in the channel, while relying on a suitably equipped terminal for rendering of moving 2D/3D faces with non-normative models held in local memory. Models stored and updated for rendering in the terminal can be simple or complex. To support speech intelligibility, the normative specification of FAPs intends for their selective or complete use as signaled by the encoder. A masking scheme provides for selective transmission of FAPs according to what

parts of the face are naturally active from moment to moment. A further control in the FAP stream allows face animation to be suspended while leaving face features in the terminal in a defined quiescent state for higher overall efficiency during multi-point connections.

A body model is a representation of a virtual human or human-like character that allows portraying body movements adequate to achieve nonverbal communication and general actions. A body model is animated by a stream of *body animation parameters* (BAP) encoded for low-bitrate transmission in broadcast and dedicated interactive communications. The BAPs manipulate independent degrees of freedom in the skeleton model of the body to produce animation of the body parts. The BAPs are quantized considering the joint limitations, and prediction errors are calculated and coded arithmetically. Similar to the face, the remote manipulation of a body model in a terminal with BAPs can accomplish lifelike visual scenes of the body in real-time without sending pictorial and video details of the body every frame.

The BAPs, if correctly interpreted, will produce reasonably similar high level results in terms of body posture and animation on different body models, also without the need to initialize or calibrate the model. The BDP set defines the set of parameters to transform the default body to a customized body optionally with its body surface, body dimensions, and texture.

The *body definition parameters* (BDP) allow the encoder to replace the local model of a more capable terminal. BDP parameters include body geometry, calibration of body parts, degrees of freedom, and optionally deformation information.

The FBA Animation specification is defined in ISO/IEC 14496-1:1999/Amd.1:2000 and this part of ISO/IEC 14496. This clause is intended to facilitate finding various parts of specification. As a rule of thumb, FAP and BAP specification is found in the part 2, and FDP and BDP specification in the part 1. However, this is not a strict rule. For an overview of FAPs/BAPs and their interpretation, read subclauses “6.1.5.2 Facial animation parameter set”, “6.1.5.3 Facial animation parameter units”, “6.1.5.4 Description of a neutral face” as well as the Table C-1. The viseme parameter is documented in subclause “7.12.3 Decoding of the viseme parameter fap 1” and the Table C-5 in annex C. The expression parameter is documented in subclause “7.12.4 Decoding of the expression parameter fap 2” and the Table C-3. FBA bitstream syntax is found in subclauses “6.2.10 FBA Object”, semantics in “6.3.10 FBA Object”, and subclause “7.12 FBA object decoding” explains in more detail the FAP/BAP decoding process. FAP/BAP masking and interpolation is explained in subclauses “6.3.11.1 FBA Object Plane”, “7.12.1.1 Decoding of FBA”, “7.12.5 FBA masking”. The FIT interpolation scheme is documented in subclause “7.2.5.3.2.4 FIT” of ISO/IEC 14496-1:1999. The FDPs and BDPs and their interpretation are documented in subclause “7.2.5.3.2.6 FDP” of ISO/IEC 14496-1:1999. In particular, the FDP feature points are documented in Figure C-1. Details on body models are documented in annex C.

Mesh object

A 2D *mesh object* is a representation of a 2D deformable geometric shape, with which synthetic video objects may be created during a composition process at the decoder, by spatially piece-wise warping of existing video object planes or still texture objects. The instances of mesh objects at a given time are called *mesh object planes* (mops). The geometry of mesh object planes is coded losslessly. Temporally and spatially predictive techniques and variable length coding are used to compress 2D mesh geometry. The coded representation of a 2D mesh object includes representation of its geometry and motion.

3D Mesh Object

The 3D Mesh Object is a 3D polygonal model that can be represented as an IndexedFaceSet or Hierarchical 3D Mesh node in BIFS. It is defined by the position of its vertices (geometry), by the association between each face and its sustaining vertices (connectivity), and optionally by colours, normals, and texture coordinates (properties). Properties do not affect the 3D geometry, but influence the way the model is shaded. 3D mesh coding (3DMC) addresses the efficient coding of 3D mesh object. It comprises a basic method and several options. The basic 3DMC method operates on manifold model and features incremental representation of single resolution 3D model. The model may be triangular or polygonal – the latter are triangulated for coding purposes and are fully recovered in the decoder. Options include: (a) support for computational graceful degradation control; (b) support for non-manifold model; (c) support for error resilience; and (d) quality scalability via hierarchical transmission of levels of detail with implicit support for smooth transition between consecutive levels. The compression of application-specific geometry streams (Face Animation Parameters) and generalized animation parameters (BIFS Anim) are currently addressed elsewhere in this part of ISO/IEC 14496.

In 3DMC, the compression of the connectivity of the 3D mesh (e.g. how edges, faces, and vertices relate) is lossless, whereas the compression of the other attributes (such as vertex coordinates, normals, colours, and texture coordinates) may be lossy.

Single Resolution Mode

The incremental representation of a single resolution 3D model is based on the Topological Surgery scheme. For manifold triangular 3D meshes, the Topological Surgery representation decomposes the connectivity of each *connected component* into a *simple polygon* and a *vertex graph*. All the triangular faces of the 3D mesh are connected in the simple polygon forming a *triangle tree*, which is a spanning tree in the dual graph of the 3D mesh. Figure V2 - 1 shows an example of a triangular 3D mesh, its dual graph, and a triangle tree. The vertex graph identifies which pairs of boundary edges of the simple polygon are associated with each other to reconstruct the connectivity of the 3D mesh. The triangle tree does not fully describe the triangulation of the simple polygon. The missing information is recorded as a *marching edge*.

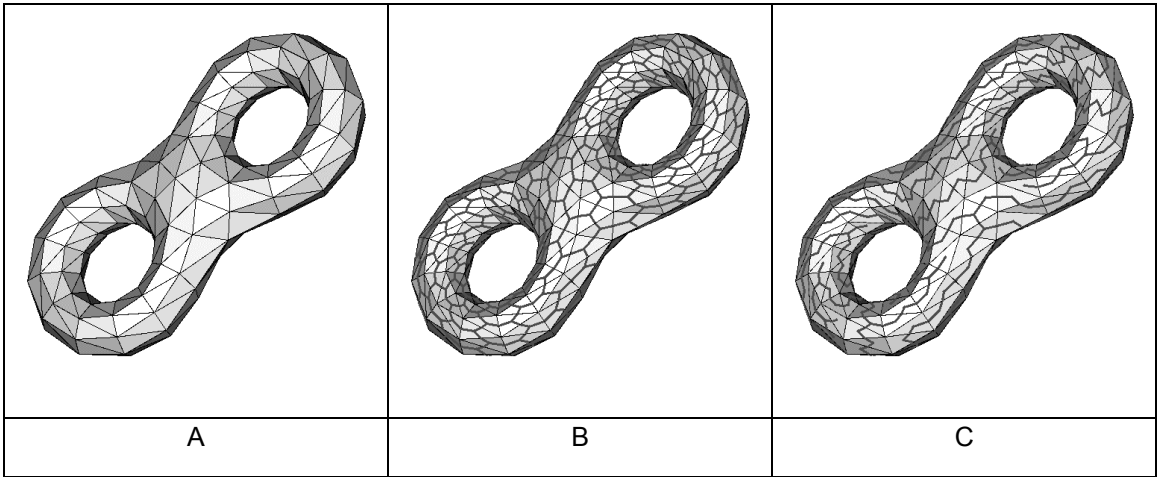


Figure V2 - 1 -- A triangular 3D mesh (A), its dual graph (B), and a triangle tree (C)

For manifold 3D meshes, the connectivity is represented in a similar fashion. The polygonal faces of the 3D mesh are connected in a simple polygon forming a *face tree*. The faces are triangulated, and which edges of the resulting triangular 3D mesh are edges of the original 3D mesh is recorded as a sequence of *polygon_edge* bits. The face tree is also a spanning tree in the dual graph of the 3D mesh, and the vertex graph is always composed of edges of the original 3D mesh.

The vertex coordinates and optional properties of the 3D mesh (normals, colours, and texture coordinates) are quantized, predicted as a function of decoded ancestors with respect to the order of traversal, and the errors are entropy encoded.

Incremental Representation

When a 3D mesh is downloaded over networks with limited bandwidth (e.g. PSTN), it may be desired to begin decoding and rendering the 3D mesh before it has all been received. Moreover, content providers may wish to control such incremental representation to present the most important data first. The basic 3DMC method supports this by interleaving the data such that each triangle may be reconstructed as it is received. Incremental representation is also facilitated by the options of hierarchical transmission for quality scalability and partitioning for error resilience.

Hierarchical Mode

An example of a 3D mesh represented in hierarchical mode is illustrated in Figure V2 - 2. The hierarchical mode allows the decoder to show progressively better approximations of the model as data are received. The hierarchical 3D mesh decomposition can also be organized in the decoder as layered detail, and view-dependent expansion of this detail can be subsequently accomplished during a viewer's interaction with the 3D model. Downloadable

scalability of 3D model allows decoders with widely varied rendering performance to use the content, without necessarily making repeated requests to the encoder for specific versions of the content and without the latencies of updating view-dependent model from the encoder. This quality scalability of 3D meshes is complementary to scaleable still texture and supports bitstream scalability and downloading of quality hierarchies of hybrid imagery to the decoder.

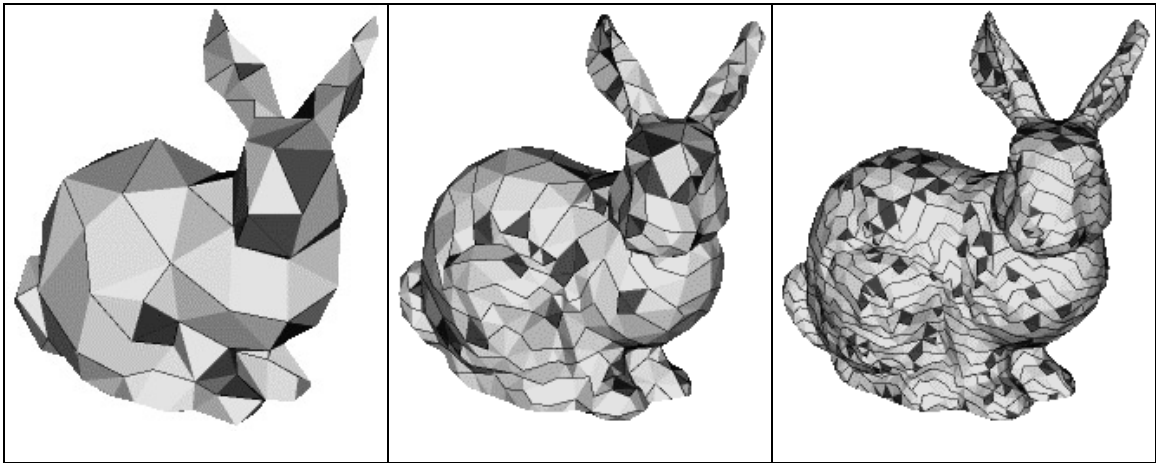


Figure V2 - 2 -- A hierarchy of levels of detail

The hierarchical representation of a 3D model is based on the Progressive Forest Split scheme. In the Progressive Forest Split representation, a manifold 3D mesh is represented as a *base 3D mesh* followed by a sequence of *forest split* operations. The base 3D mesh is encoded as a single resolution 3D mesh using the Topological Surgery scheme. Each forest split operation is composed of a *forest* in the graph of the current 3D mesh, and a *sequence of simple polygons*. Figure V2 - 3 illustrates the method. To prevent visual artifacts which may occur while switching from a level of detail to the next one, the data structure supports smooth transition between consecutive levels of detail in the form of linear interpolation of vertex positions.

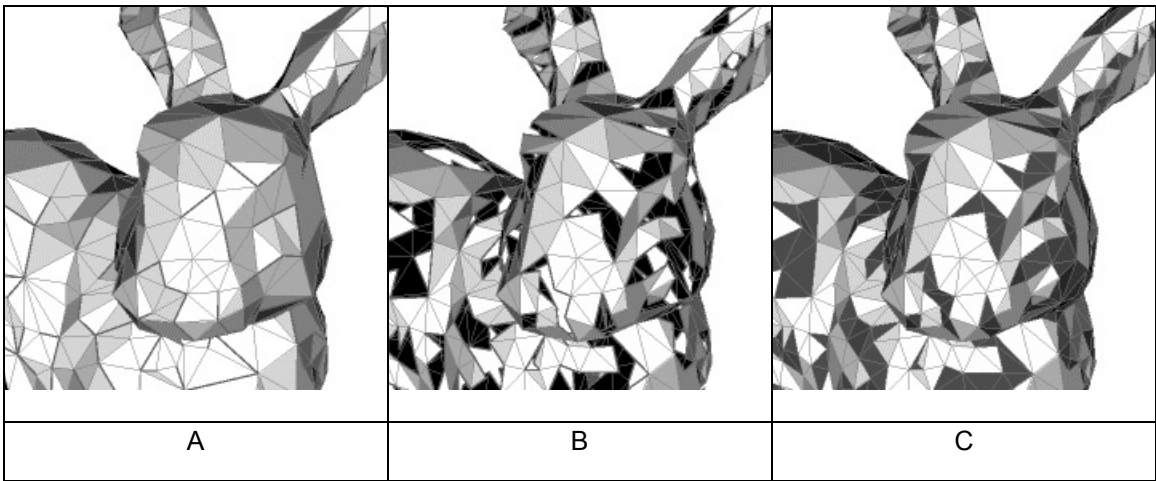


Figure V2 - 3 -- The Forest Split operation. A 3D mesh with a forest of edges marked (A). The tree loops resulting from cutting the 3D mesh through the forest edges (B). Each tree loop is filled by a simple polygon composed of polygonal faces (C).

Error Resilience for 3D mesh object

If the 3D mesh is partitioned into independent parts, it may be possible to perform more efficient data transmission in an error-prone environment, e.g., an IP network or datacasting service in a broadcast TV network. It must be possible to resynchronize after a channel error, and continue data transmission and rendering from that point instead of starting over from scratch. Even with the presence of channel errors, the decoder can start decoding and rendering from the next partition that is received intact from the channel.

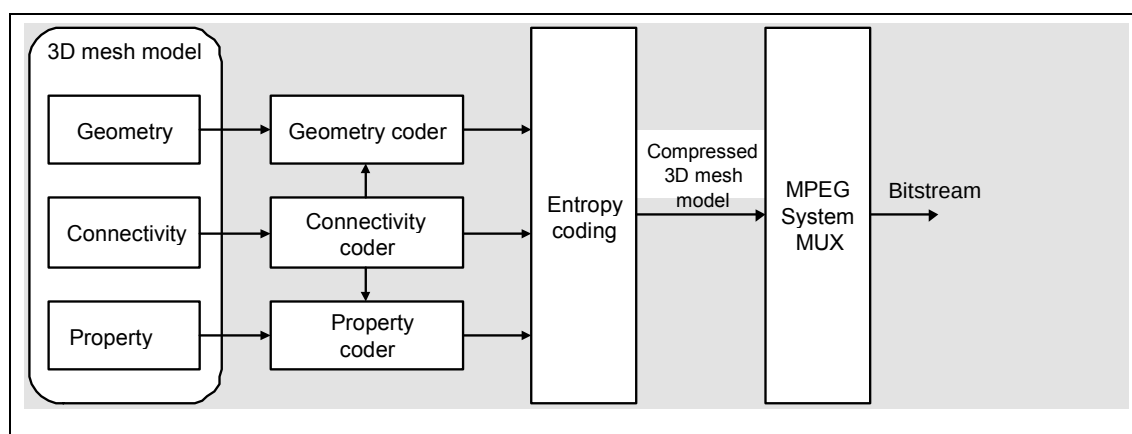
Flexible partitioning methods can be used to organize the data, such that it fits the underlying network packet structure more closely, and overhead is reduced to the minimum. To allow flexible partitioning, several connected components may be merged into one partition, where as a large connected component may be divided into several independent partitions. Merging and dividing of connected components using different partition types can be done at any point in the 3D mesh object.

Stitching for Non-Manifold and Non-Orientable Meshes

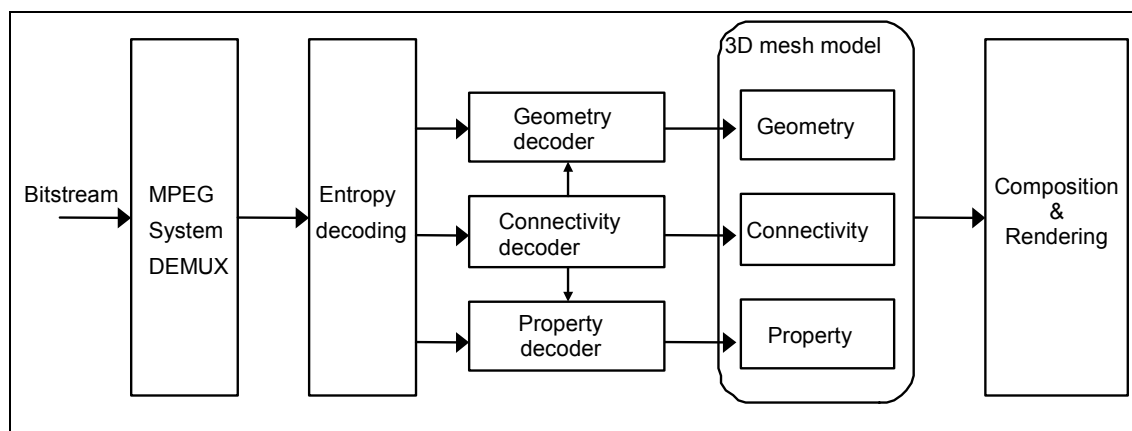
The connectivity of a non-manifold and non-orientable 3D mesh is represented as a manifold 3D mesh and a sequence of *stitches*. Each stitch describes how to identify one or more pairs of vertices along two linear paths of edges, and each one of these paths is contained in one of the vertex graphs that span the connected components of the manifold 3D mesh.

Encoder and Decoder Block Diagrams

High level block diagrams of a general 3D polygonal model encoder and decoder are shown in Figure V2 - 4. They consist of a 3D mesh connectivity (de)coder, geometry (de)coder, property (de)coder, and entropy (de)coding blocks. Connectivity, vertex position, and property information are extracted from 3D mesh model described in VRML or MPEG-4 BIFS format. The connectivity (de)coder is used for an efficient representation of the association between each face and its sustaining vertices. The geometry (de)coder is used for a lossy or lossless compression of vertex coordinates. The property (de)coder is used for a lossy or lossless compression of colour, normal, and texture coordinate data.



(a) 3D mesh encoder



(b) 3D mesh decoder

Figure V2 - 4 -- General block diagram of the 3D mesh compress

Overview of the object based nonscalable syntax

The coded representation defined in the non-scalable syntax achieves a high compression ratio while preserving good image quality. Further, when access to individual objects is desired, the shape of objects also needs to be coded, and depending on the bandwidth available, the shape information can be coded in a lossy or lossless fashion.

The compression algorithm employed for texture data is not lossless as the exact sample values are not preserved during coding. Obtaining good image quality at the bitrates of interest demands very high compression, which is not achievable with intra coding alone. The need for random access, however, is best satisfied with pure intra coding. The choice of the techniques is based on the need to balance a high image quality and compression ratio with the requirement to make random access to the coded bitstream.

A number of techniques are used to achieve high compression. The algorithm first uses block-based motion compensation to reduce the temporal redundancy. Motion compensation is used both for causal prediction of the current VOP from a previous VOP, and for non-causal, interpolative prediction from past and future VOPs. Motion vectors are defined for each 16-sample by 16-line region of a VOP or 8-sample by 8-line region of a VOP as required. The prediction error, is further compressed using the discrete cosine transform (DCT) to remove spatial correlation before it is quantised in an irreversible process that discards the less important information. Finally, the shape information, motion vectors and the quantised DCT information, are encoded using variable length codes.

Temporal processing

Because of the conflicting requirements of random access to and highly efficient compression, three main VOP types are defined. Intra coded VOPs (I-VOPs) are coded without reference to other pictures. They provide access points to the coded sequence where decoding can begin, but are coded with only moderate compression. Predictive coded VOPs (P-VOPs) are coded more efficiently using motion compensated prediction from a past intra or predictive coded VOPs and are generally used as a reference for further prediction. Bidirectionally-predictive coded VOPs (B-VOPs) provide the highest degree of compression but require both past and future reference VOPs for motion compensation. Bidirectionally-predictive coded VOPs are never used as references for prediction (except in the case that the resulting VOP is used as a reference for scalable enhancement layer). The organisation of the three VOP types in a sequence is very flexible. The choice is left to the encoder and will depend on the requirements of the application.

Coding of Shapes

In natural video scenes, VOPs are generated by segmentation of the scene according to some semantic meaning. For such scenes, the shape information is thus binary (binary shape). Shape information is also referred to as alpha plane. The binary alpha plane is coded on a macroblock basis by a coder which uses the context information, motion compensation and arithmetic coding.

For coding of shape of a VOP, a bounding rectangle is first created and is extended to multiples of 16×16 blocks with extended alpha samples set to zero. Shape coding is then initiated on a 16×16 block basis; these blocks are also referred to as binary alpha blocks.

Motion representation - macroblocks

The choice of 16×16 blocks (referred to as macroblocks) for the motion-compensation unit is a result of the trade-off between the coding gain provided by using motion information and the overhead needed to represent it. Each macroblock can further be subdivided to 8×8 blocks for motion estimation and compensation depending on the overhead that can be afforded. In order to encode the highly active scene with higher VOP rate, a Reduced Resolution VOP tool is provided. When this tool is used, the size of the macroblock used for motion compensation decoding is 32×32 pixels and the size of block is 16×16 pixels.

Depending on the type of the macroblock, motion vector information and other side information is encoded with the compressed prediction error in each macroblock. The motion vectors are differenced with respect to a prediction value and coded using variable length codes. The maximum length of the motion vectors allowed is decided at the encoder. It is the responsibility of the encoder to calculate appropriate motion vectors. The specification does not specify how this should be done.

Spatial redundancy reduction

Both source VOPs and prediction errors VOPs have significant spatial redundancy. This part of ISO/IEC 14496 uses a block-based DCT method with optional visually weighted quantisation, and run-length coding. After motion compensated prediction or interpolation, the resulting prediction error is split into 8×8 blocks. These are transformed into the DCT domain where they can be weighted before being quantised. After quantisation many of the DCT coefficients are zero in value and so two-dimensional run-length and variable length coding is used to encode the remaining DCT coefficients efficiently.

Chrominance formats

This part of ISO/IEC 14496 currently supports the 4:2:0 chrominance format.

Pixel depth

This part of ISO/IEC 14496 supports pixel depths between 4 and 12 bits in luminance and chrominance planes.

Generalized scalability

The scalability tools in this part of ISO/IEC 14496 are designed to support applications beyond that supported by single layer video. The major applications of scalability include internet video, wireless video, multi-quality video services, video database browsing etc. In some of these applications, either normal scalabilities on picture basis such as those in ISO/IEC 13818-2:1996 may be employed or object based scalabilities may be necessary; both categories of scalability are enabled by this part of ISO/IEC 14496.

Although a simple solution to scalable video is the simulcast technique that is based on transmission/storage of multiple independently coded reproductions of video, a more efficient alternative is scalable video coding, in which the bandwidth allocated to a given reproduction of video can be partially re-utilised in coding of the next reproduction of video. In scalable video coding, it is assumed that given a coded bitstream, decoders of various complexities can decode and display appropriate reproductions of coded video. A scalable video encoder is likely to have increased complexity when compared to a single layer encoder. However, this part of ISO/IEC 14496 provides several different forms of scalabilities that address non-overlapping applications with corresponding complexities.

The basic scalability tools offered are temporal scalability and spatial scalability. Moreover, combinations of these basic scalability tools are also supported and are referred to as hybrid scalability. In the case of basic scalability, two layers of video referred to as the lower layer and the enhancement layer are allowed, whereas in hybrid scalability up to four layers are supported.

Object based Temporal scalability

Temporal scalability is a tool intended for use in a range of diverse video applications from video databases, internet video, wireless video and multiview/stereoscopic coding of video. Furthermore, it may also provide a migration path from current lower temporal resolution video systems to higher temporal resolution systems of the future.

Temporal scalability involves partitioning of VOPs into layers, where the lower layer is coded by itself to provide the basic temporal rate and the enhancement layer is coded with temporal prediction with respect to the lower layer. These layers when decoded and temporally multiplexed yield full temporal resolution. The lower temporal resolution systems may only decode the lower layer to provide basic temporal resolution whereas enhanced systems of the future may support both layers. Furthermore, temporal scalability has use in bandwidth constrained networked applications where adaptation to frequent changes in allowed throughput are necessary. An additional advantage of temporal scalability is its ability to provide resilience to transmission errors as the more important data of the lower layer can be sent over a channel with better error performance, whereas the less critical enhancement layer can be sent over a channel with poor error performance. Object based temporal scalability can also be employed to allow graceful control of picture quality by controlling the temporal rate of each video object under the constraint of a given bit-budget.

Object Spatial scalability

Spatial scalability is a tool intended for use in video applications involving multi quality video services, video database browsing, internet video and wireless video, i.e., video systems with the primary common feature that a minimum of two layers of spatial resolution are necessary. Spatial scalability involves generating two spatial resolution video layers from a single video source such that the lower layer is coded by itself to provide the basic spatial resolution and the enhancement layer employs the spatially interpolated lower layer and carries the full spatial resolution of the input video source.

Object based spatial scalability is composed of texture spatial scalability and shape spatial scalability. They can be used independently or together by its application. Binary shape spatial scalability is used for the applications that have 'binary shape only' mode as well as the applications of general object based spatial scalability.

An additional advantage of spatial scalability is its ability to provide resilience to transmission errors as the more important data of the lower layer can be sent over a channel with better error performance, whereas the less critical enhancement layer data can be sent over a channel with poor error performance. Further, it can also allow interoperability between various standards.

Hybrid scalability

There are a number of applications where neither the temporal scalability nor the spatial scalability may offer the necessary flexibility and control. This may necessitate use of temporal and spatial scalability simultaneously and is referred to as the hybrid scalability. Among the applications of hybrid scalability are wireless video, internet video, multiviewpoint/stereoscopic coding etc.

Error Resilience

This part of ISO/IEC 14496 provides error robustness and resilience to allow accessing of image or video information over a wide range of storage and transmission media. The error resilience tools developed for this part of ISO/IEC 14496 can be divided into three major categories. These categories include synchronization, data recovery, and error concealment. It should be noted that these categories are not unique to this part of ISO/IEC 14496, and have been used elsewhere in general research in this area. It is, however, the tools contained in these categories that are of interest, and where this part of ISO/IEC 14496 makes its contribution to the problem of error resilience.

Information technology — Coding of audio-visual objects —

Part 2: Visual

1 Scope

This part of ISO/IEC 14496 specifies the coded representation of picture information in the form of natural or synthetic visual objects like video sequences of rectangular or arbitrarily shaped pictures, moving 2D meshes, animated 3D face and body models and texture for synthetic objects. The coded representation allows for content based access for digital storage media, digital video communication and other applications. ISO/IEC 14496 specifies also the decoding process of the aforementioned coded representation. The representation supports constant bitrate transmission, variable bitrate transmission, robust transmission, content based random access (including normal random access), object based scalable decoding (including normal scalable decoding), object based bitstream editing, as well as special functions such as fast forward playback, fast reverse playback, slow motion, pause and still pictures. Synthetic objects and coding of special 2D/3D meshes, texture, and animation parameters are provided for use with downloadable models to exploit mixed media and the bandwidth improvement associated with remote manipulation of such models. ISO/IEC 14496 is intended to allow some level of interoperability with ISO/IEC 11172-2:1993, ISO/IEC 13818-2:1996 and ITU-T Recommendation H.263.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 14496. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 14496 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ITU-T Recommendation T.81 (1992)|ISO/IEC 10918-1:1994, *Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines*.

ISO/IEC 11172-1:1993, *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 1: Systems*.

ISO/IEC 11172-2:1993, *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 2: Video*.

ISO/IEC 11172-3:1993, *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 3: Audio*.

ITU-T Recommendation H.222.0(2000)|ISO/IEC 13818-1:2000, *Information technology — Generic coding of moving pictures and associated audio information: Systems*.

ITU-T Recommendation H.262(2000)|ISO/IEC 13818-2:2000, *Information technology — Generic coding of moving pictures and associated audio information: Video*.

ISO/IEC 13818-3:1998, *Information technology — Generic coding of moving pictures and associated audio information — Part 3: Audio*.

ITU-R Recommendation BT.601-5, *Encoding parameters of digital television for studios*.

IEEE Standard Specifications for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, IEEE Std 1180-1990, December 6, 1990.

IEC 60461:2001, *Time and control code for video tape recorder*.

ITU-T Recommendation H.261:1993, *Video codec for audiovisual services at p x 64 kbit/s*.

ITU-T Recommendation H.263:2000, *Enhanced reference picture selection mode*.

3 Terms and definitions

For the purposes of this part of ISO/IEC 14496, the following terms and definitions apply.

- 3.1 3D mesh:** A polygonal mesh. It is defined by the position of its vertices (geometry), by the association between each face and its sustaining vertices (connectivity), and optionally by colors, normals, and texture coordinates (properties). Properties do not affect the 3D geometry, but influence the way the model is shaded.
- 3.2 3D mesh connectivity:** The association between each face of a 3D mesh and its sustaining vertices.
- 3.3 3D mesh geometry:** The three dimensional coordinates of the vertices of a 3D mesh.
- 3.4 3D mesh properties:** The normals, colors, and texture coordinates of a 3D mesh.
- 3.5 AC coefficient:** Any DCT coefficient for which the frequency in one or both dimensions is non-zero.
- 3.6 B-VOP; bidirectionally predictive-coded video object plane (VOP):** A VOP that is coded using motion compensated prediction from past and/or future reference VOPs.
- 3.7 backward motion vector:** A motion vector that is used for motion compensation from a reference VOP at a later time in display order.
- 3.8 backward prediction:** Prediction from the future reference VOP.
- 3.9 base layer:** An independently decodable layer of a scalable hierarchy.
- 3.10 binary alpha block:** A block of size 16×16 pels, colocated with macroblock, representing shape information of the binary alpha map; it is also referred to as a bab.
- 3.11 binary alpha map:** A 2D binary mask used to represent the shape of a video object such that the pixels that are opaque are considered as part of the object where as pixels that are transparent are not considered to be part of the object.
- 3.12 bitstream; stream:** An ordered series of bits that forms the coded representation of the data.
- 3.13 bitrate:** The rate at which the coded bitstream is delivered from the storage medium or network to the input of a decoder.
- 3.14 block:** An 8-row by 8-column (or 16-row by 16-column in motion compensation decoding of Reduced Resolution VOP) matrix of samples, or 64 DCT coefficients (source, quantised or dequantised).
- 3.15 body animation parameters, BAP:** Coded streaming animation parameters that manipulate the body joints, and that govern deformations of the body surface with the use of BAT.

- 3.16 body definition parameters, BDP:** Downloadable data to customize a baseline body model in the decoder to a particular body, or to download a body model along with the information about how to animate it. The BDPs are normally transmitted once per session, followed by a stream of compressed BAPs. BDPs may include body geometry, body skeleton, and body deformation tables.
- 3.17 body deformation table, BodyDefTable:** A downloadable function mapping from incoming BAPs to body surface geometry that provides a combination of BAPs for controlling body surface geometry deformation.
- 3.18 boundary (of a 3D mesh):** The set of boundary edges of a 3D mesh.
- 3.19 boundary edge (of a 3D mesh):** An edge of a 3D mesh which has exactly one incident face.
- 3.20 bounding loop:** The boundary of a simple polygon.
- 3.21 bounding loop index:** An index of the bounding loop table which contains the list of vertices lying on the bounding loop
- 3.22 branching triangle (of a triangle tree):** A triangle of a triangle tree with three incident triangle tree edges.
- 3.23 branching vertex (of a vertex tree or of a vertex graph):** A vertex of a vertex tree with three or more incident vertex tree edges.
- 3.24 byte aligned:** A bit in a coded bitstream is byte-aligned if its position is a multiple of 8-bits from the first bit in the stream.
- 3.25 byte:** Sequence of 8-bits.
- 3.26 context based arithmetic encoding:** The method used for coding of binary shape; it is also referred to as cae.
- 3.27 channel:** A digital medium or a network that stores or transports a bitstream constructed according to ISO/IEC 14496.
- 3.28 chrominance format:** Defines the number of chrominance blocks in a macroblock.
- 3.29 chrominance component:** A matrix, block or single sample representing one of the two colour difference signals related to the primary colours in the manner defined in the bitstream. The symbols used for the chrominance signals are Cr and Cb.
- 3.30 coded B-VOP:** A B-VOP that is coded.
- 3.31 coded VOP:** A coded VOP is a coded I-VOP, a coded P-VOP, a coded B-VOP, or a coded S(GMC)-VOP.
- 3.32 coded I-VOP:** An I-VOP that is coded.
- 3.33 coded P-VOP:** A P-VOP that is coded.
- 3.34 coded S(GMC)-VOP:** A S(GMC)-VOP that is coded.
- 3.35 coded video bitstream:** A coded representation of a series of one or more VOPs as defined in this part of ISO/IEC 14496.
- 3.36 coded representation:** A data element as represented in its encoded form.

- 3.37 coding parameters:** The set of user-definable parameters that characterise a coded video bitstream. Bitstreams are characterised by coding parameters. Decoders are characterised by the bitstreams that they are capable of decoding.
- 3.38 component:** A matrix, block or single sample from one of the three matrices (luminance and two chrominance) that make up a picture.
- 3.39 composition process:** The (non-normative) process by which reconstructed VOPs are composed into a scene and displayed.
- 3.40 compression:** Reduction in the number of bits used to represent an item of data.
- 3.41 connected component (of a 3D mesh):** A 3D mesh can be decomposed into one or more disjoint sets called connected components. Every face of the 3D mesh is assigned to an unique connected component, such that any pair of two faces that share an edge belongs to the same connected component.
- 3.42 constant bitrate coded video:** A coded video bitstream with a constant bitrate.
- 3.43 constant bitrate:** Operation where the bitrate is constant from start to finish of the coded bitstream.
- 3.44 conversion ratio:** The size conversion ratio for the purpose of rate control of shape.
- 3.45 corner (of a 3D mesh):** A (face,vertex) pair, where the vertex is one of the sustaining vertices of the face.
- 3.46 data element:** An item of data as represented before encoding and after decoding.
- 3.47 DC coefficient:** The DCT coefficient for which the frequency is zero in both dimensions.
- 3.48 DCT coefficient:** The amplitude of a specific cosine basis function.
- 3.49 decoder input buffer:** The first-in first-out (FIFO) buffer specified in the video buffering verifier.
- 3.50 decoder:** An embodiment of a decoding process.
- 3.51 decoding order:** The order in which the VOPs are transmitted and decoded. This order is not necessarily the same as the display order.
- 3.52 decoding (process):** The process defined in this part of ISO/IEC 14496 that reads an input coded bitstream and produces decoded VOPs or audio samples, or 3D meshes.
- 3.53 dequantisation:** The process of rescaling the quantised DCT coefficients after their representation in the bitstream has been decoded and before they are presented to the inverse DCT.
- 3.54 digital storage media; DSM:** A digital storage or transmission device or system.
- 3.55 discrete cosine transform; DCT:** Either the forward discrete cosine transform or the inverse discrete cosine transform. The DCT is an invertible, discrete orthogonal transformation. The inverse DCT is defined in annex A.
- 3.56 display order:** The order in which the decoded pictures are displayed. Normally this is the same order in which they were presented at the input of the encoder.
- 3.57 dual graph (of a 3D mesh):** A graph composed of a set of *dual graph nodes* and a set of *dual graph edges*. Each dual graph node corresponds to a face of the 3D mesh. Each dual graph edge corresponds to an internal edge of the 3D mesh and links two dual graph nodes.

- 3.58 edge (of a 3D mesh):** An unordered pair of vertex indices of a 3D mesh, consecutive in one or more faces of the 3D mesh (modulo cyclical permutations of the sequences).
- 3.59 editing:** The process by which one or more coded bitstreams are manipulated to produce a new coded bitstream. Conforming edited bitstreams must meet the requirements defined in this part of ISO/IEC 14496.
- 3.60 encoder:** An embodiment of an encoding process.
- 3.61 encoding (process):** A process, not specified in this part of ISO/IEC 14496, that reads a stream of input pictures or audio samples and produces a valid coded bitstream as defined in this part of ISO/IEC 14496.
- 3.62 enhancement layer:** A relative reference to a layer (above the base layer) in a scalable hierarchy. For all forms of scalability, its decoding process can be described by reference to the lower layer decoding process and the appropriate additional decoding process for the enhancement layer itself.
- 3.63 face (of a 3D mesh):** An alternating sequence of vertices and edges, such that each edge in the sequence connects the vertices immediately preceding and following it in the sequence (with the exception of the last edge that connects the vertex preceding it with the first vertex of the sequence). No vertex may appear more than once in the sequence. In practice, a face is often represented by the list of vertices in the sequence.
- 3.64 global motion compensation, GMC:** The use of global spatial transformation to improve the efficiency of the prediction of sample values. The prediction uses global spatial transformation to provide offsets into the past reference VOPs containing previously decoded sample values that are used to form the prediction error.
- 3.65 face animation parameter units, FAPU:** Special normalized units (e.g. translational, angular, logical) defined to allow interpretation of FAPs with any facial model in a consistent way to produce reasonable results in expressions and speech pronunciation.
- 3.66 face animation parameters, FAP:** Coded streaming animation parameters that manipulate the displacements and angles of face features, and that govern the blending of visemes and face expressions during speech.
- 3.67 face animation table, FAT:** A downloadable function mapping from incoming FAPs to feature control points in the face mesh that provides piecewise linear weightings of the FAPs for controlling face movements.
- 3.68 face calibration mesh:** Definition of a 3D mesh for calibration of the shape and structure of a baseline face model.
- 3.69 face definition parameters, FDP:** Downloadable data to customize a baseline face model in the decoder to a particular face, or to download a face model along with the information about how to animate it. The FDPs are normally transmitted once per session, followed by a stream of compressed FAPs. FDPs may include feature points for calibrating a baseline face, face texture and coordinates to map it onto the face, animation tables, etc.
- 3.70 face feature control point:** A normative vertex point in a set of such points that define the critical locations within face features for control by FAPs and that allow for calibration of the shape of the baseline face.
- 3.71 face forest (of a 3D mesh):** The set of all the face trees corresponding to the connected components of a 3D mesh.

- 3.72** **face interpolation transform, FIT:** A downloadable node type defined in ISO/IEC 14496-1:1999 for optional mapping of incoming FAPs to FAPs before their application to feature points, through weighted rational polynomial functions, for complex cross-coupling of standard FAPs to link their effects into custom or proprietary face models.
- 3.73** **face model mesh:** A 2D or 3D contiguous geometric mesh defined by vertices and planar polygons utilizing the vertex coordinates, suitable for rendering with photometric attributes (e.g. texture, color, normals).
- 3.74** **face tree (of a 3D mesh):** A spanning tree constructed in the dual graph of a connected component.
- 3.75** **feathering:** A tool that tapers the values around edges of binary alpha mask for composition with the background.
- 3.76** **flag:** A one bit integer variable which may take one of only two values (zero and one).
- 3.77** **forbidden:** The term “forbidden” when used in the clauses defining the coded bitstream indicates that the value shall never be used. This is usually to avoid emulation of start codes.
- 3.78** **forced updating:** The process by which macroblocks are intra-coded from time-to-time to ensure that mismatch errors between the inverse DCT processes in encoders and decoders cannot build up excessively.
- 3.79** **forward compatibility:** A newer coding standard is forward compatible with an older coding standard if decoders designed to operate with the newer coding standard are able to decode bitstreams of the older coding standard.
- 3.80** **forward motion vector:** A motion vector that is used for motion compensation from a reference frame VOP at an earlier time in display order.
- 3.81** **forward prediction:** Prediction from the past reference VOP.
- 3.82** **frame:** A frame contains lines of spatial information of a video signal. For progressive video, these lines contain samples starting from one time instant and continuing through successive lines to the bottom of the frame.
- 3.83** **frame period:** The reciprocal of the frame rate.
- 3.84** **frame rate:** The rate at which frames are be output from the composition process.
- 3.85** **future reference VOP:** A future reference VOP is a reference VOP that occurs at a later time than the current VOP in display order.
- 3.86** **hierarchical 3D mesh:** a BIFS node representing a 3D mesh.
- 3.87** **hierarchical representation (of a 3D mesh):** A 3D mesh representation with two or more levels of detail in increasing order of resolution, composed of a first coarse level of detail (base 3D mesh) followed by one or more refinement operations. Each refinement operation determines how to construct a new level of detail as a refinement of the current level of detail.
- 3.88** **hybrid scalability:** Hybrid scalability is the combination of two (or more) types of scalability.
- 3.89** **imaginary edge:** An imaginary edge is inserted to triangulate a polygon.
- 3.90** **incremental representation (of a 3D mesh):** A representation that allows the terminal to start rendering a subset of the 3D mesh before the whole bitstream has been completely received and/or decoded.

- 3.91 IndexedFaceSet:** a BIFS node representing a 3D mesh.
- 3.92 interlace:** The property of conventional television frames where alternating lines of the frame represent different instances in time. In an interlaced frame, one of the field is meant to be displayed first. This field is called the first field. The first field can be the top field or the bottom field of the frame.
- 3.93 internal edge (of a 3D mesh):** An edge of a 3D mesh which has exactly two incident faces.
- 3.94 I-VOP; intra-coded VOP:** A VOP coded using information only from itself.
- 3.95 intra coding:** Coding of a macroblock or VOP that uses information only from that macroblock or VOP.
- 3.96 intra shape coding:** Shape coding that does not use any temporal prediction.
- 3.97 inter shape coding:** Shape coding that uses temporal prediction.
- 3.98 jump edge (of a 3D mesh):** An edge of the vertex graph which does not belong to any vertex tree.
- 3.99 level:** A defined set of constraints on the values which may be taken by the parameters of this part of ISO/IEC 14496 within a particular profile. A profile may contain one or more levels. In a different context, level is the absolute value of a non-zero coefficient (see “run”).
- 3.100 layer:** In a scalable hierarchy denotes one out of the ordered set of bitstreams and (the result of) its associated decoding process.
- 3.101 layered bitstream:** A single bitstream associated to a specific layer (always used in conjunction with layer qualifiers, e. g. “enhancement layer bitstream”).
- 3.102 leaf triangle (of a 3D mesh):** A triangle of a triangle tree with exactly one incident triangle tree edge.
- 3.103 leaf vertex (of a 3D mesh):** A vertex of a vertex tree with exactly one incident vertex tree edge.
- 3.104 lower layer:** A relative reference to the layer immediately below a given enhancement layer (implicitly including decoding of *all* layers below this enhancement layer).
- 3.105 luminance component:** A matrix, block or single sample representing a monochrome representation of the signal and related to the primary colours in the manner defined in the bitstream. The symbol used for luminance is Y.
- 3.106 Mbit:** 1 000 000 bits.
- 3.107 macroblock:** The four 8×8 (or 16×16 in motion compensation decoding of Reduced Resolution VOP) blocks of luminance data and the two (for 4:2:0 chrominance format) corresponding 8×8 (or 16×16 in motion compensation decoding of Reduced Resolution VOP) blocks of chrominance data coming from a 16×16 (or 32×32 in motion compensation decoding of Reduced Resolution VOP) section of the luminance component of the picture. Macroblock is sometimes used to refer to the sample data and sometimes to the coded representation of the sample values and other data elements defined in the macroblock header of the syntax defined in this part of ISO/IEC 14496. The usage is clear from the context.
- 3.108 manifold (3D mesh):** A 3D mesh which has neither singular vertices nor singular edges.
- 3.109 marching edge (of a 3D mesh):** An edge shared by two triangles along a triangle run.
- 3.110 mesh (2D):** A 2D triangular mesh refers to a planar graph which tessellates a video object plane into triangular patches. The vertices of the triangular mesh elements are referred to as node points. The straight-line segments between node points are referred to as edges. Two triangles are adjacent if they share a common edge.

- 3.111 **mesh geometry**: The spatial locations of the node points and the triangular structure of a mesh.
- 3.112 **mesh motion**: The temporal displacements of the node points of a mesh from one time instance to the next.
- 3.113 **mesh object plane, MOP**: The instance of a mesh object at a given time.
- 3.114 **motion compensation**: The use of motion vectors to improve the efficiency of the prediction of sample values. The prediction uses motion vectors to provide offsets into the past and/or future reference VOPs containing previously decoded sample values that are used to form the prediction error.
- 3.115 **motion estimation**: The process of estimating motion vectors during the encoding process.
- 3.116 **motion vector**: A two-dimensional vector used for motion compensation that provides an offset from the coordinate position in the current picture or field to the coordinates in a reference VOP.
- 3.117 **motion vector for shape**: A motion vector used for motion compensation of shape.
- 3.118 **multiple auxiliary components**: Auxiliary components are defined for the VOP on a pixel-by-pixel basis, and contain data related to the video object, such as disparity, depth, and additional texture. Up to three auxiliary components (including the grayscale shape) are possible.
- 3.119 **non-intra coding**: Coding of a macroblock or a VOP that uses information both from itself and from macroblocks and VOPs occurring at other times.
- 3.120 **non-manifold (3D mesh)**: A 3D mesh that contains singular vertices or singular edges.
- 3.121 **non-orientable (3D mesh)**: A manifold 3D mesh which is not orientable.
- 3.122 **opaque macroblock**: A macroblock with shape mask of all 255's.
- 3.123 **orientable (3D mesh)**: A manifold 3D mesh for which an orientation can be chosen for each of its faces such that for each internal edge, the two faces incident to the edge induce opposite orientations on the common edge. All non-manifold 3D meshes are non-orientable.
- 3.124 **orientation (of a face of a 3D mesh)**: One of the two possible cyclical orderings of its sustaining vertices.
- 3.125 **orientation (of an edge of a 3D mesh)**: One of the two possible orderings of the two vertices of the edge.
- 3.126 **oriented (manifold 3D mesh)**: A manifold 3D mesh for which an orientation has been chosen for each of its faces such that for each internal edge, the two faces incident to the edge induce opposite orientations on the common edge. Note that a manifold 3D mesh may be orientable but not oriented. Such a mesh can be oriented by choosing a consistent orientation for its faces (which may require inverting the orientation of one or more faces).
- 3.127 **P-VOP; predictive-coded VOP**: A picture that is coded using motion compensated prediction from the past VOP.
- 3.128 **parameter**: A variable within the syntax of this part of ISO/IEC 14496 which may take one of a range of values. A variable which can take one of only two values is called a flag.
- 3.129 **partition**: a bitstream segment of a 3D mesh that can be decoded independently.
- 3.130 **past reference picture**: A past reference VOP is a reference VOP that occurs at an earlier time than the current VOP in composition order.

- 3.131 picture:** Source, coded or reconstructed image data. A source or reconstructed picture consists of three rectangular matrices of 8-bit numbers representing the luminance and two chrominance signals. A “coded VOP” was defined earlier. For progressive video, a picture is identical to a frame.
- 3.132 prediction:** The use of a predictor to provide an estimate of the sample value or data element currently being decoded.
- 3.133 prediction error:** The difference between the actual value of a sample or data element and its predictor.
- 3.134 predictor:** A linear combination of previously decoded sample values or data elements.
- 3.135 profile:** A subset of the syntax of this part of ISO/IEC 14496, defined in terms of Visual Object Types.
- 3.136 progressive:** The property of film frames where all the samples of the frame represent the same instances in time.
- 3.137 quantisation matrix:** A set of sixty-four 8-bit values used by the dequantiser.
- 3.138 quantised DCT coefficients:** DCT coefficients before dequantisation. A variable length coded representation of quantised DCT coefficients is transmitted as part of the coded video bitstream.
- 3.139 quantiser scale:** A scale factor coded in the bitstream and used by the decoding process to scale the dequantisation.
- 3.140 random access:** The process of beginning to read and decode the coded bitstream at an arbitrary point.
- 3.141 reconstructed VOP:** A reconstructed VOP consists of three matrices of 8-bit numbers representing the luminance and two chrominance signals. It is obtained by decoding a coded VOP.
- 3.142 reference layer:** A layer to be referenced for prediction in a scalable hierarchy. The `video_object_id` of the reference layer should be the same value as the `video_object_id` of the enhancement layer. The `ref_layer_id` of the enhancement layer is set to the same value as the `video_object_layer_id` of the reference layer.
- 3.143 reference VOP:** A reference VOP is a reconstructed VOP that was coded in the form of a coded I-VOP, a coded P-VOP, or a coded S(GMC)-VOP. Reference VOPs are used for forward and backward prediction when P-VOPs, B-VOPs, and S(GMC)-VOPs are decoded.
- 3.144 regular triangle (of a triangle tree):** A triangle of a triangle tree with exactly two incident triangle tree edges.
- 3.145 regular vertex (of a 3D mesh):** A vertex of a 3D mesh is *regular* if the set of incident edges and incident faces can be ordered in an alternating sequence of edges and faces, such that each face in the sequence is bounded by the edges immediately preceeding and following it in the sequence.
- 3.146 regular vertex (of a vertex tree):** A vertex of a vertex tree with exactly two incident vertex tree edges.
- 3.147 reordering delay:** A delay in the decoding process that is caused by VOP reordering.
- 3.148 reserved:** The term “reserved” when used in the clauses defining the coded bitstream indicates that the value may be used in the future for ISO/IEC defined extensions.
- 3.149 scalable hierarchy:** coded video data consisting of an ordered set of more than one video bitstream.

- 3.150 scalability:** Scalability is the ability of a decoder to decode an ordered set of bitstreams to produce a reconstructed sequence. Moreover, useful video is output when subsets are decoded. The minimum subset that can thus be decoded is the first bitstream in the set which is called the base layer. Each of the other bitstreams in the set is called an enhancement layer. When addressing a specific enhancement layer, “lower layer” refers to the bitstream that precedes the enhancement layer.
- 3.151 side information:** Information in the bitstream necessary for controlling the decoder.
- 3.152 run:** The number of zero coefficients preceding a non-zero coefficient, in the scan order. The absolute value of the non-zero coefficient is called “level”.
- 3.153 S-VOP:** A picture that is coded using information obtained by warping whole or part of a static sprite.
- 3.154 S(GMC)-VOP:** A picture that is coded using prediction based on global motion compensation from the past VOP.
- 3.155 saturation:** Limiting a value that exceeds a defined range by setting its value to the maximum or minimum of the range as appropriate.
- 3.156 simple polygon (of a 3D mesh):** A connected component of which the dual graph is a tree. A simple polygon has exactly one boundary, and all its vertices lie on the boundary.
- 3.157 singular edge (of a 3D mesh):** An edge of a 3D mesh which has three or more incident faces.
- 3.158 singular vertex (of a 3D mesh):** A vertex of a 3D mesh which is not a regular vertex.
- 3.159 source; input:** Term used to describe the video material or some of its attributes before encoding.
- 3.160 spanning forest (of a graph):** A subgraph of a graph composed of one or more trees, each tree spanning a connected component of the graph.
- 3.161 spatial prediction:** prediction derived from a decoded frame of the reference layer decoder used in spatial scalability.
- 3.162 spatial scalability:** A type of scalability where an enhancement layer also uses predictions from sample data derived from a lower layer without using motion vectors. The layers can have different VOP sizes or VOP rates.
- 3.163 static sprite:** The luminance, chrominance and binary alpha plane for an object which does not vary in time.
- 3.164 start codes:** 32-bit codes embedded in that coded bitstream that are unique. They are used for several purposes including identifying some of the structures in the coding syntax.
- 3.165 stuffing (bits); stuffing (bytes):** Code-words that may be inserted into the coded bitstream that are discarded in the decoding process. Their purpose is to increase the bitrate of the stream which would otherwise be lower than the desired bitrate.
- 3.166 temporal prediction:** prediction derived from reference VOPs other than those defined as spatial prediction.
- 3.167 temporal scalability:** A type of scalability where an enhancement layer also uses predictions from sample data derived from a lower layer using motion vectors. The layers have identical frame size, and but can have different VOP rates.
- 3.168 top layer:** the topmost layer (with the highest layer_id) of a scalable hierarchy.
- 3.169 transparent macroblock:** A macroblock with shape mask of all zeros.

- 3.170 traversal order:** The traversal order determines which branch to be traversed first at each branching triangle.
- 3.171 tree loop (of a 3D mesh):** An alternating sequence of vertices and edges bounding the hole created by cutting a 3D mesh through a tree of edges within a forest split operation.
- 3.172 tree loop face:** a face incident to a tree loop.
- 3.173 tree loop corner:** a corner of a tree loop face incident to a tree loop edge.
- 3.174 triangle tree (of a 3D mesh):** A spanning tree in the dual graph of a connected component of a triangular 3D mesh.
- 3.175 triangle run:** A path in a triangle tree connecting a first leaf or branching triangle to a last leaf or branching triangle through zero or more intermediate triangles.
- 3.176 triangular 3D mesh:** A 3D mesh with triangular faces. Also, a 3D mesh after all the polygonal faces have been triangulated.
- 3.177 variable bitrate:** Operation where the bitrate varies with time during the decoding of a coded bitstream.
- 3.178 variable length coding; VLC:** A reversible procedure for coding that assigns shorter code-words to frequent events and longer code-words to less frequent events.
- 3.179 vertex graph (of a 3D mesh):** The graph formed by the vertices and the edges of the 3D mesh which do not correspond to dual graph edges included in the face forest.
- 3.180 vertex run:** A path in a vertex tree connecting a first leaf or branching vertex to a last leaf or branching vertex through zero or more intermediate regular vertices.
- 3.181 vertex tree:** A rooted tree spanning one connected component of the vertex graph.
- 3.182 video buffering verifier; VBV:** Part of a hypothetical decoder that is conceptually connected to the output of the encoder. Its purpose is to provide a constraint on the variability of the data rate that an encoder or editing process may produce.
- 3.183 video complexity verifier; VCV:** Part of a hypothetical decoder that is conceptually connected to the output of the encoder. Its purpose is to provide a constraint on the maximum processing requirements of the bitstream that an encoder or editing process may produce.
- 3.184 video memory verifier; VMV:** Part of a hypothetical decoder that is conceptually connected to the output of the encoder. Its purpose is to provide a constraint on the maximum reference memory requirements of the bitstream that an encoder or editing process may produce.
- 3.185 video object plane, VOP:** The instance of a video object at a given time.
- 3.186 video presentation verifier; VPV:** Part of a hypothetical decoder that is conceptually connected to the output of the encoder. Its purpose is to provide a constraint on the maximum presentation memory requirements of the bitstream that an encoder or editing process may produce.
- 3.187 video session:** The highest syntactic structure of coded video bitstreams. It contains a series of one or more coded video objects.
- 3.188 viseme:** the physical (visual) configuration of the mouth, tongue and jaw that is visually correlated with the speech sound corresponding to a phoneme.

- 3.189 virtual triangle:** A virtual triangle is a leaf triangle added, if necessary, at the end of a partition to signal the end of the partition to the decoder.
- 3.190 VOP reordering:** The process of reordering the reconstructed VOPs when the decoding order is different from the composition order for display. VOP reordering occurs when B-VOPs are present in a bitstream. There is no VOP reordering when decoding low delay bitstreams.
- 3.191 warping:** Processing applied to extract a sprite VOP from a static sprite or a reference VOP. It consists of a global spatial transformation driven by a few motion parameters (0,2,4,6,8), to recover luminance, chrominance and shape information.
- 3.192 Y-vertex:** The vertex of a branching triangle opposite to the incident marching edge.
- 3.193 zigzag scanning order:** A specific sequential ordering of the DCT coefficients from (approximately) the lowest spatial frequency to the highest.

4 Abbreviations and symbols

The mathematical operators used to describe this part of ISO/IEC 14496 are similar to those used in the C programming language. However, integer divisions with truncation and rounding are specifically defined. Numbering and counting loops generally begin from zero.

4.1 Arithmetic operators

+	Addition.
-	Subtraction (as a binary operator) or negation (as a unary operator).
++	Increment. i.e. $x++$ is equivalent to $x = x + 1$
--	Decrement. i.e. $x--$ is equivalent to $x = x - 1$
$\times$ * }	Multiplication.
^	Power.
/	Integer division with truncation of the result toward zero. For example, $7/4$ and $-7/-4$ are truncated to 1 and $-7/4$ and $7/-4$ are truncated to -1.
//	Integer division with rounding to the nearest integer. Half-integer values are rounded away from zero unless otherwise specified. For example $3//2$ is rounded to 2, and $-3//2$ is rounded to -2.
///	Integer division with sign dependent rounding to the nearest integer. Half-integer values when positive are rounded away from zero, and when negative are rounded towards zero. For example $3///2$ is rounded to 2, and $-3///2$ is rounded to -1.
////	Integer division with truncation towards the negative infinity.
÷	Used to denote division in mathematical equations where no truncation or rounding is intended.
%	Modulus operator. Defined only for positive numbers.
Sign()	$\text{Sign}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$

$$\text{Abs}(x) = \begin{cases} x & x \geq 0 \\ -x & x < 0 \end{cases}$$

$\sum_{i=a}^{i=b} f(i)$ The summation of the $f(i)$ with i taking integral values from a up to, but not including b .

4.2 Logical operators

`||` Logical OR.

`&&` Logical AND.

`!` Logical NOT.

4.3 Relational operators

`>` Greater than.

`>=` Greater than or equal to.

`≥` Greater than or equal to.

`<` Less than.

`<=` Less than or equal to.

`≤` Less than or equal to.

`==` Equal to.

`!=` Not equal to.

`max [ , ... , ]` the maximum value in the argument list.

`min [ , ... , ]` the minimum value in the argument list.

4.4 Bitwise operators

`&` AND

`|` OR

`>>` Shift right with sign extension.

`<<` Shift left with zero fill.

4.5 Conditional operators

`?:` $(condition? a : b) = \begin{cases} a & \text{if } condition \text{ is true,} \\ b & \text{otherwise.} \end{cases}$

4.6 Assignment

`=` Assignment operator.

4.7 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bitstream.

- bslbf** Bit string, left bit first, where “left” is the order in which bit strings are written in this part of ISO/IEC 14496. Bit strings are generally written as a string of 1s and 0s within single quote marks, e.g. ‘1000 0001’. Blanks within a bit string are for ease of reading and have no significance. For convenience large strings are occasionally written in hexadecimal, in this case conversion to a binary in the conventional manner will yield the value of the bit string. Thus the left most hexadecimal digit is first and in each hexadecimal digit the most significant of the four bits is first.
- uimsbf** Unsigned integer, most significant bit first.
- simsbf** Signed integer, in twos complement format, most significant (sign) bit first.
- vlcblf** Variable length code, left bit first, where “left” refers to the order in which the VLC codes are written. The byte order of multibyte words is most significant byte first.

4.8 Constants

- II* 3,141 592 653 58...
- e* 2,718 281 828 45...

5 Conventions

5.1 Method of describing bitstream syntax

The bitstream retrieved by the decoder is described in subclause 6.2. Each data item in the bitstream is in bold type. It is described by its name, its length in bits, and a mnemonic for its type and order of transmission.

The action caused by a decoded data element in a bitstream depends on the value of that data element and on data elements previously decoded. The decoding of the data elements and definition of the state variables used in their decoding are described in subclause 6.3. The following constructs are used to express the conditions when data elements are present, and are in normal type:

while (condition) { data_element ... }	If the condition is true, then the group of data elements occurs next in the data stream. This repeats until the condition is not true.
do { data_element ... } while (condition)	The data element always occurs at least once. The data element is repeated until the condition is not true.
do { ... continue ... } while (condition)	Continues execution of the next repetition of the nearest while-do loop.
if (condition) { data_element ... } else {	If the condition is true, then the first group of data elements occurs next in the data stream. If the condition is not true, then the second group of data

data_element ... }	elements occurs next in the data stream.
for (i = m; i < n; i++) { data_element ... }	The group of data elements occurs (n-m) times. Conditional constructs within the group of data elements may depend on the value of the loop control variable i, which is set to m for the first occurrence, incremented by one for the second occurrence, and so forth.
/* comment ... */	Explanatory comment that may be deleted entirely without in any way altering the syntax.

This syntax uses the 'C-code' convention that a variable or expression evaluating to a non-zero value is equivalent to a condition that is true and a variable or expression evaluating to a zero value is equivalent to a condition that is false. In many cases a literal string is used in a condition. For example;

```
if ( video_object_layer_shape == "rectangular" ) ...
```

In such cases the literal string is that used to describe the value of the bitstream element in subclause 6.3. In this example, we see that "rectangular" is defined in a Table 6-14 to be represented by the two bit binary number '00'.

As noted, the group of data elements may contain nested conditional constructs. For compactness, the brackets { } are omitted when only one data element follows.

data_element [n] data_element [n] is the n+1th element of an array of data.

data_element [m][n] data_element [m][n] is the m+1, n+1th element of a two-dimensional array of data.

data_element [l][m][n] data_element [l][m][n] is the l+1, m+1, n+1th element of a three-dimensional array of data.

While the syntax is expressed in procedural terms, it should not be assumed that subclause 6.2 implements a satisfactory decoding procedure. In particular, it defines a correct and error-free input bitstream. Actual decoders must include means to look for start codes in order to begin decoding correctly, and to identify errors, erasures or insertions while decoding. The methods to identify these situations, and the actions to be taken, are not standardised.

5.2 Definition of functions

Several utility functions for picture coding algorithm are defined as follows:

5.2.1 Definition of next_bits() function

The function next_bits() permits comparison of a bit string with the next bits to be decoded in the bitstream.

5.2.2 Definition of bytealigned() function

The function bytealigned () returns 1 if the current position is on a byte boundary, that is the next bit in the bitstream is the first bit in a byte. Otherwise it returns 0.

5.2.3 Definition of nextbits_bytealigned() function

The function nextbits_bytealigned() returns a bit string starting from the next byte aligned position. This permits comparison of a bit string with the next byte aligned bits to be decoded in the bitstream. If the current location in the bitstream is already byte aligned and the 8 bits following the current location are '01111111', the bits subsequent to these 8 bits are returned. The current location in the bitstream is not changed by this function.

5.2.4 Definition of next_start_code() function

The next_start_code() function removes any zero bit and a string of 0 to 7 '1' bits used for stuffing and locates the next start code.

next_start_code() {	No. of bits	Mnemonic
zero_bit	1	'0'
while (!bytealigned())		
one_bit	1	'1'
}		

5.2.5 Definition of next_resync_marker() function

The next_resync_marker() function removes any zero bit and a string of 0 to 7 '1' bits used for stuffing and locates the next resync marker; it thus performs similar operation as next_start_code() but for resync_marker.

next_resync_marker() {	No. of bits	Mnemonic
zero_bit	1	'0'
while (!bytealigned())		
one_bit	1	'1'
}		

5.2.6 Definition of transparent_mb() function

The function transparent_mb() returns 1 if the current macroblock consists only of transparent pixels. Otherwise it returns 0.

5.2.7 Definition of transparent_block() function

The function transparent_block(j) returns 1 if the 8×8 with index j consists only of transparent pixels. Otherwise it returns 0. The index value for each block is defined in Figure 6-5.

5.2.8 Definition of byte_align_for_upstream() function

The function byte_align_for_upstream() removes a string of '1' used for stuffing from the upstream message. When the message is already byte aligned before the byte_align_for_upstream() function, additional byte stuffing is no longer allowed.

byte_align_for_upstream() {	No. of bits	Mnemonic
while (!byte_aligned())		
one_bit	1	'1'
}		

5.3 Reserved, forbidden and marker_bit

The terms "reserved" and "forbidden" are used in the description of some values of several fields in the coded bitstream.

The term "reserved" indicates that the value may be used in the future for ISO/IEC defined extensions.

The term “forbidden” indicates a value that shall never be used (usually in order to avoid emulation of start codes).

The term “marker_bit” indicates a one bit integer in which the value zero is forbidden (and it therefore shall have the value ‘1’). These marker bits are introduced at several points in the syntax to avoid start code emulation.

The term “zero_bit” indicates a one bit integer with the value zero.

5.4 Arithmetic precision

In order to reduce discrepancies between implementations of this part of ISO/IEC 14496, the following rules for arithmetic operations are specified.

- (a) Where arithmetic precision is not specified, such as in the calculation of the IDCT, the precision shall be sufficient so that significant errors do not occur in the final integer values.
- (b) Where ranges of values are given, the end points are included if a square bracket is present, and excluded if a round bracket is used. For example, [a , b) means from a to b, including a but excluding b.

6 Visual bitstream syntax and semantics

6.1 Structure of coded visual data

Coded visual data can be of several different types, such as video data, still texture data, 2D mesh data or facial animation parameter data.

Synthetic objects and their attribution are structured in a hierarchical manner to support both bitstream scalability and object scalability. ISO/IEC 14496-1:1999 of the specification provides the approach to spatial-temporal scene composition including normative 2D/3D scene graph nodes and their composition supported by Binary Interchange Format Specification. At this level, synthetic and natural object composition relies on ISO/IEC 14496-1:1999 with subsequent (non-normative) rendering performed by the application to generate specific pixel-oriented views of the models.

Coded video data consists of an ordered set of video bitstreams, called layers. If there is only one layer, the coded video data is called non-scalable video bitstream. If there are two layers or more, the coded video data is called a scalable hierarchy.

One of the layers is called base layer, and it can always be decoded independently. Other layers are called enhancement layers, and can only be decoded together with the lower layers (previous layers in the ordered set), starting with the base layer. The multiplexing of these layers is discussed in ISO/IEC 14496-1:1999. The base layer of a scalable set of streams can be coded by other standards. The Enhancement layers shall conform to this part of ISO/IEC 14496. In general the visual bitstream can be thought of as a syntactic hierarchy in which syntactic structures contain one or more subordinate structures.

Visual texture, referred to herein as still texture coding, is designed for maintaining high visual quality in the transmission and rendering of texture under widely varied viewing conditions typical of interaction with 2D/3D synthetic scenes. Still texture coding provides for a multi-layer representation of luminance, color and shape. This supports progressive transmission of the texture for image build-up as it is received by a terminal. Also supported is the downloading of the texture resolution hierarchy for construction of image pyramids used by 3D graphics APIs. Quality and SNR scalability are supported by the structure of still texture coding.

Coded mesh data consists of a single non-scalable bitstream. This bitstream defines the structure and motion of a 2D mesh object. Texture that is to be mapped onto the mesh geometry is coded separately.

Coded FBA parameter data consists of one non-scaleable bitstream. It defines the animation of the face and body model of the decoder. Face and body animation data is structured as standard formats for downloadable models and their animation controls, and a single layer of compressed face and body animation parameters used for remote manipulation of the face and body model. The face is a node in a scene graph that includes face geometry

ready for rendering. The body is also a node in a scene graph that includes body geometry ready for rendering. The face node belongs to the scene graph that is defined by the body node. The shape, texture and expressions of the face are generally controlled by the bitstream containing instances of Facial Definition Parameter (FDP) sets and/or Facial Animation Parameter (FAP) sets. The body is also controlled by instances of Body Definition Parameter (BDP) sets and/or Body Animation Parameter (BAP) sets. Upon initial or baseline construction, the FBA object contains a generic face with a neutral expression and a generic body with default position. This object can receive FAPs and BAPs from the bitstream and be subsequently rendered to produce animation of the face and the body. If FDPs and BDPs are transmitted, the generic face and body are transformed into a particular face and body of specific shape and appearance. A downloaded face model via FDPs is a scene graph for insertion in the face node, and the downloaded body model via BDPs is a scene graph for insertion of the body node.

6.1.1 Visual object sequence

Visual object sequence is the highest syntactic structure of the coded visual bitstream.

A visual object sequence commences with a `visual_object_sequence_start_code` which is followed by one or more visual objects coded concurrently. The visual object sequence is terminated by a `visual_object_sequence_end_code`.

6.1.2 Visual object

A visual object commences with a `visual_object_start_code`, is followed by profile and level identification, and a visual object id, and is followed by a video object, a still texture object, a mesh object, or an FBA object.

6.1.3 Video object

A video object commences with a `video_object_start_code`, and is followed by one or more video object layers.

6.1.3.1 Progressive and interlaced sequences

This part of ISO/IEC 14496 deals with coding of both progressive and interlaced sequences.

The sequence, at the output of the decoding process, consists of a series of reconstructed VOPs separated in time and are readied for display via the compositor.

6.1.3.2 Frame

A frame consists of three rectangular matrices of integers; a luminance matrix (Y), and two chrominance matrices (Cb and Cr).

6.1.3.3 VOP

A reconstructed VOP is obtained by decoding a coded VOP. A coded VOP may have been derived from either a progressive or interlaced frame.

6.1.3.4 VOP types

There are four types of VOPs that use different coding methods:

1. An Intra-coded (I) VOP is coded using information only from itself.
2. A Predictive-coded (P) VOP is a VOP which is coded using motion compensated prediction from a past reference VOP.
3. A Bidirectionally predictive-coded (B) VOP is a VOP which is coded using motion compensated prediction from a past and/or future reference VOP(s).
4. A sprite (S) VOP is a VOP for a sprite object or a VOP which is coded using prediction based on global motion compensation from a past reference VOP.

6.1.3.5 I-VOPs and group of VOPs

I-VOPs are intended to assist random access into the sequence. Applications requiring random access, fast-forward playback, or fast reverse playback may use I-VOPs relatively frequently.

I-VOPs may also be used at scene cuts or other cases where motion compensation is ineffective.

Group of VOP (GOV) header is an optional header that can be used immediately before a coded I-VOP to indicate to the decoder:

- 1) the modulo part (i.e. the full second units) of the time base for the next VOP to be displayed after having decoded a GOV header
- 2) if the first consecutive B-VOPs immediately following the coded I-VOP can be reconstructed properly in the case of a random access.

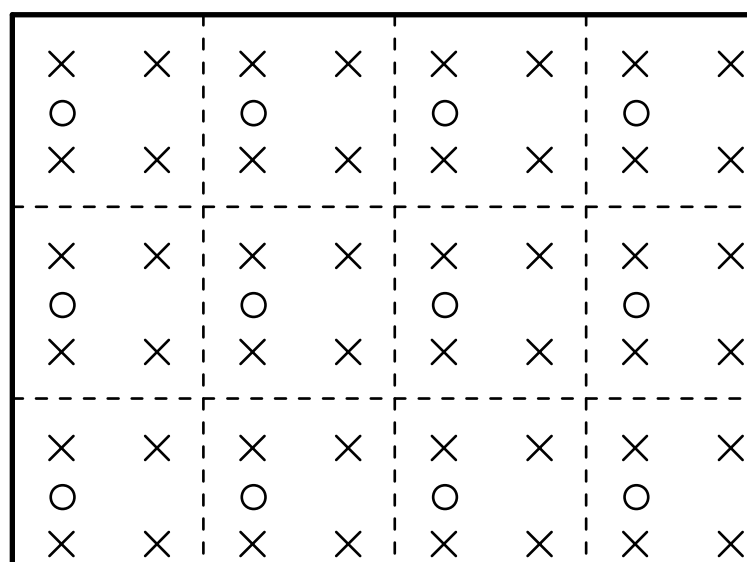
In a non scalable bitstream or the base layer of a scalable bitstream, the first coded VOP following a GOV header shall be a coded I-VOP.

6.1.3.6 Format

In this format the Cb and Cr matrices shall be one half the size of the Y-matrix in both horizontal and vertical dimensions. The Y-matrix shall have an even number of lines and samples.

The luminance and chrominance samples are positioned as shown in Figure 6-1. The two variations in the vertical and temporal positioning of the samples for interlaced VOPs are shown in Figure 6-2 and Figure 6-3.

Figure 6-4 shows the vertical and temporal positioning of the samples in a progressive frame.



X Represent luminance samples

O Represent chrominance samples

Figure 6-1 -- The position of luminance and chrominance samples in 4:2:0 data

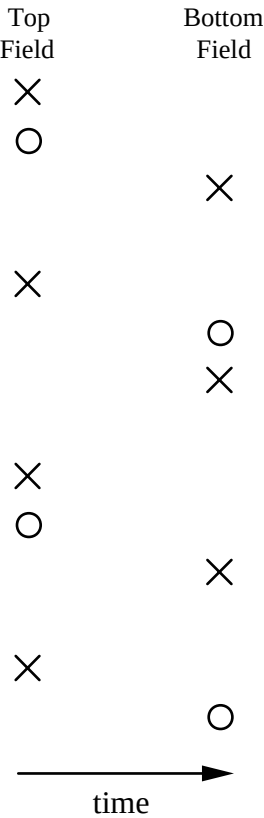


Figure 6-2 -- Vertical and temporal positions of samples in an interlaced frame with top_field_first=1

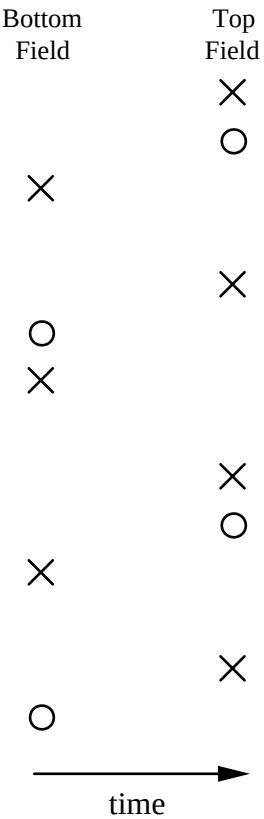


Figure 6-3 -- Vertical and temporal position of samples in an interlaced frame with top_field_first=0

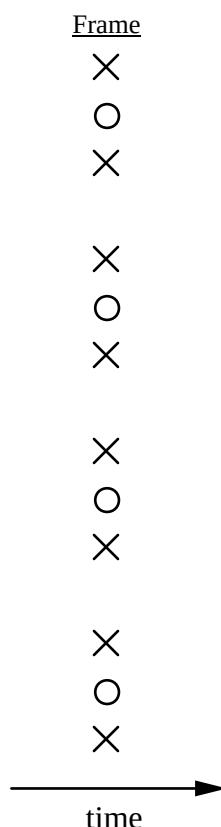


Figure 6-4 -- Vertical and temporal positions of samples in a progressive frame

The binary alpha plane for each VOP is represented by means of a bounding rectangle as described in clause F.2, and it has always the same number of lines and pixels per line as the luminance plane of the VOP bounding rectangle. The positions between the luminance and chrominance pixels of the bounding rectangle are defined in this clause according to the 4:2:0 format. For the progressive case, each 2x2 block of luminance pixels in the bounding rectangle associates to one chrominance pixel. For the interlaced case, each 2x2 block of luminance pixels of the same field in the bounding rectangle associates to one chrominance pixel of that field.

In order to perform the padding process on the two chrominance planes, it is necessary to generate a binary alpha plane which has the same number of lines and pixels per line as the chrominance planes. Therefore, when non-scalable shape coding is used, this binary alpha plane associated with the chrominance planes is created from the binary alpha plane associated with the luminance plane by the subsampling process defined below:

For each 2x2 block of the binary alpha plane associated with the luminance plane of the bounding rectangle (of the same frame for the progressive and of the same field for the interlaced case), the associated pixel value of the binary alpha plane associated with the chrominance planes is set to 255 if any pixel of said 2x2 block of the binary alpha plane associated with the luminance plane equals 255.

6.1.3.7 VOP reordering

When a video object layer contains coded B-VOPs, the number of consecutive coded B-VOP is variable and unbounded. The first coded VOP shall not be a B-VOP.

A video object layer may contain no coded P-VOPs or no coded S(GMC)-VOPs. A video object layer may also contain no coded I-VOPs in which case some care is required at the start of the video object layer and within the video object layer to effect both random access and error recovery.

The order of the coded VOPs in the bitstream, also called decoding order, is the order in which a decoder reconstructs them. The order of the reconstructed VOPs at the output of the decoding process, also called the

display order, is not always the same as the decoding order and this subclause defines the rules of VOP reordering that shall happen within the decoding process.

When the video object layer contains no coded B-VOPs, the decoding order is the same as the display order.

When B-VOPs are present in the video object layer re-ordering is performed according to the following rules:

If the current VOP in decoding order is a B-VOP the output VOP is the VOP reconstructed from that B-VOP.

If the current VOP in decoding order is a I-VOP, P-VOP, or S(GMC)-VOPs the output VOP is the VOP reconstructed from the previous I-VOP or P-VOP, or S(GMC)-VOP if one exists. If none exists, at the start of the video object layer, no VOP is output.

The following is an example of VOPs taken from the beginning of a video object layer. In this example there are two coded B-VOPs between successive coded P-VOPs and also two coded B-VOPs between successive coded I- and P-VOPs. VOP '1I' is used to form a prediction for VOP '4P'. VOPs '4P' and '1I' are both used to form predictions for VOPs '2B' and '3B'. Therefore the order of coded VOPs in the coded sequence shall be '1I', '4P', '2B', '3B'. However, the decoder shall display them in the order '1I', '2B', '3B', '4P'.

At the encoder input,

1	2	3	4	5	6	7	8	9	10	11	12	13
I	B	B	P	B	B	P	B	B	I	B	B	P

At the encoder output, in the coded bitstream, and at the decoder input,

1	4	2	3	7	5	6	10	8	9	13	11	12
I	P	B	B	P	B	B	I	B	B	P	B	B

At the decoder output,

1	2	3	4	5	6	7	8	9	10	11	12	13
I	B	B	P	B	B	P	B	B	I	B	B	P

In terms of VOP reordering, an S(GMC)-VOP can be regarded as a P-VOP.

6.1.3.8 Macroblock

A macroblock contains a section of the luminance component and the spatially corresponding chrominance components. The term macroblock can either refer to source and decoded data or to the corresponding coded data elements. A skipped macroblock is one for which no information is transmitted. Presently there is only one chrominance format for a macroblock, namely, 4:2:0 format. The orders of blocks in a macroblock is illustrated below:

A 4:2:0 Macroblock consists of 6 blocks. This structure holds 4 Y, 1 Cb and 1 Cr Blocks and the block order is depicted in Figure 6-5.

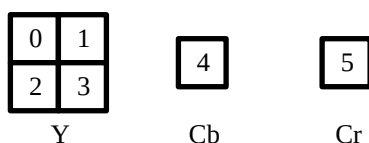


Figure 6-5 -- 4:2:0 Macroblock structure

The organisation of VOPs into macroblocks is as follows.

For the case of a progressive VOP, the interlaced flag (in the VOP header) is set to “0” and the organisation of lines of luminance VOP into macroblocks is called frame organization and is illustrated in Figure 6-6. In this case, frame DCT coding is employed.

For the case of interlaced VOP, the interlaced flag is set to “1” and the organisation of lines of luminance VOP into macroblocks can be either frame organization or field organization and thus both frame and field DCT coding may be used in the VOP.

- In the case of frame DCT coding, each luminance block shall be composed of lines from two fields alternately. This is illustrated in Figure 6-6.
- In the case of field DCT coding, each luminance block shall be composed of lines from only one of the two fields. This is illustrated in Figure 6-7.

Only frame DCT coding is applied to the chrominance blocks. It should be noted that field based predictions may be applied for these chrominance blocks which will require predictions of 8x4 regions (after half-sample filtering).

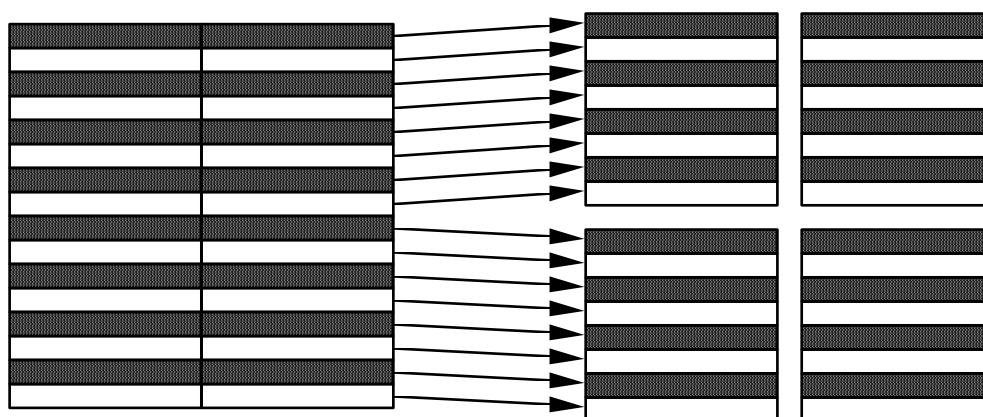


Figure 6-6 -- Luminance macroblock structure in frame DCT coding

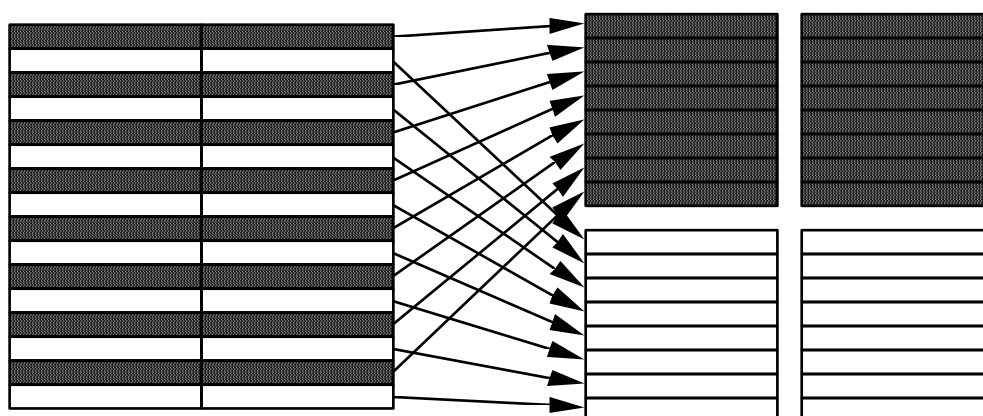


Figure 6-7 -- Luminance macroblock structure in field DCT coding

6.1.3.9 Block

The term **block** can refer either to source and reconstructed data or to the DCT coefficients or to the corresponding coded data elements.

When the block refers to source and reconstructed data it refers to an orthogonal section of a luminance or chrominance component with the same number of lines and samples. There are 8 lines and 8 samples/line in the block.

6.1.4 Mesh object

A 2D triangular *mesh* refers to a tessellation of a 2D visual object plane into triangular patches. The vertices of the triangular patches are called *node points*. The straight-line segments between node points are called *edges*. Two triangles are *adjacent* if they share a common edge.

A *dynamic* 2D mesh consists of a temporal sequence of 2D triangular meshes, where each mesh has the same topology, but node point locations may differ from one mesh to the next. Thus, a dynamic 2D mesh can be specified by the geometry of the initial 2D mesh and motion vectors at the node points for subsequent meshes, where each motion vector points from a node point of the previous mesh in the sequence to the corresponding node point of the current mesh. The dynamic 2D mesh can be used to create 2D animations by mapping texture from e.g. a video object plane onto successive 2D meshes.

A 2D dynamic mesh with *implicit structure* refers to a 2D dynamic mesh of which the initial mesh has either *uniform* or *Delaunay* topology. In both cases, the topology of the initial mesh does not have to be coded (since it is implicitly defined), only the node point locations of the initial mesh have to be coded. Note that in both the uniform and Delaunay case, the mesh is restricted to be *simple*, i.e. it consists of a single connected component without any holes, topologically equivalent to a disk.

A *mesh object* represents the geometry and motion of a 2D triangular mesh. A mesh object consists of one or more *mesh object planes*, each corresponding to a 2D triangular mesh at a certain time instance. An example of a mesh object is shown in the figure below.

A sequence of mesh object planes represents the piece-wise deformations to be applied to a video object plane or still texture object to create a synthetic animated video object. Triangular patches of a video object plane are to be warped according to the motion of corresponding triangular mesh elements. The motion of mesh elements is specified by the temporal displacements of the mesh node points.

The syntax and semantics of the mesh object pertains to the mesh geometry and mesh motion only; the video object to be used in an animation is coded separately. The warping or texture mapping applied to render visual object planes is handled in the context of scene composition. Furthermore, the syntax does not allow explicit encoding of other mesh properties such as colors or texture coordinates.

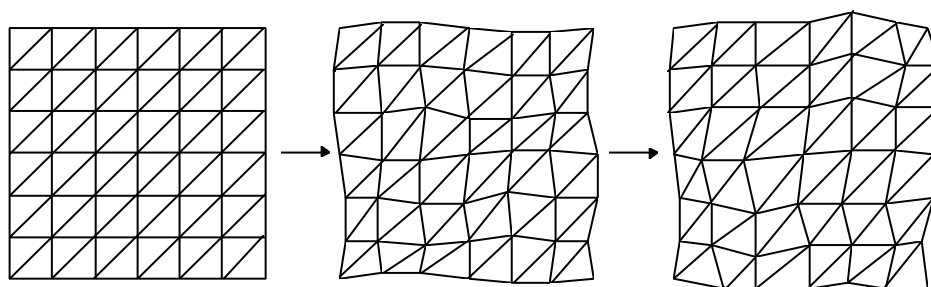


Figure 6-8 -- Mesh object with uniform triangular geometry

6.1.4.1 Mesh object plane

There are two types of mesh object planes that use different coding methods.

An *intra-coded* mesh object plane codes the geometry of a single 2D mesh. An intra-coded mesh is either of uniform or Delaunay type. In the case of a mesh of uniform type, the mesh geometry is coded by a small set of parameters. In the case of a mesh of Delaunay type, the mesh geometry is coded by the locations of the node points and boundary edge segments. The triangular mesh structure is specified implicitly by the coded information.

A *predictive-coded* mesh object plane codes a 2D mesh using temporal prediction from a past reference mesh object plane. The triangular structure of a predictive-coded mesh is identical to the structure of the reference mesh used for prediction; however, the locations of node points may change. The displacements of node points represent the motion of the mesh and are coded by specifying the motion vectors of node points from the reference mesh towards the predictive-coded mesh.

The locations of mesh node points correspond to locations in a video object or still texture object. Mesh node point locations and motion vectors are represented and coded with half pixel accuracy.

6.1.5 FBA object

Conceptually the FBA object consists of a collection of nodes in a scene graph which are animated by the FBA object bitstream. The shape, texture and expressions of the face are generally controlled by the bitstream containing instances of Facial Definition Parameter (FDP) sets and/or Facial Animation Parameter (FAP) sets. Upon construction, the FBA object contains a generic face with a neutral expression. This face can already be rendered. It is also immediately capable of receiving the FAPs from the bitstream, which will produce animation of the face: expressions, speech etc. If FDPs are received, they are used to transform the generic face into a particular face determined by its shape and (optionally) texture. Optionally, a complete face model can be downloaded via the FDP set as a scene graph for insertion in the face node.

The FDP and FAP sets are designed to allow the definition of a facial shape and texture, as well as animation of faces reproducing expressions, emotions and speech pronunciation. The FAPs, if correctly interpreted, will produce reasonably similar high level results in terms of expression and speech pronunciation on different facial models, without the need to initialize or calibrate the model. The FDPs allow the definition of a precise facial shape and texture in the setup phase. If the FDPs are used in the setup phase, it is also possible to produce more precisely the movements of particular facial features. Using a phoneme/bookmark to FAP conversion it is possible to control facial models accepting FAPs via TTS systems. The translation from phonemes to FAPs is not standardized. It is assumed that every decoder has a default face model with default parameters. Therefore, the setup stage is not necessary to create face animation. The setup stage is used to customize the face at the decoder.

Upon construction, the Body model contains a generic virtual human or human-like body with the default posture. This body can already be rendered. It is also immediately capable of receiving the BAPs from the bitstream, which will produce animation of the body. If BDPs are received, they are used to transform the decoder's generic body into a particular body determined by the parameter contents. Any component can be null. A null component is replaced by the corresponding default component when the body is rendered. Similar to the face, the BAPs can be transmitted also without first downloading BDPs, in which case the decoder animates its local model.

No assumption is made and no limitation is imposed on the range of defined mobilities for humanoid animation. In other words the human body model should be capable of supporting various applications, from realistic simulation of human motions to network games using simple human-like models.

6.1.5.1 Structure of the face and body object bitstream

A face and body object is formed by a temporal sequence of face and body object planes. This is depicted as follows in Figure 6-9.

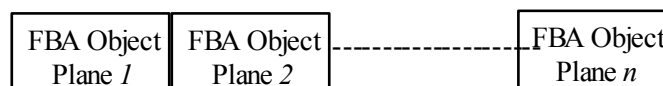
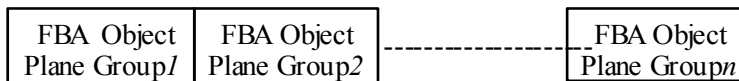
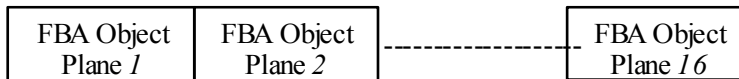


Figure 6-9 -- Structure of the FBA object bitstream

An FBA object represents a node in an ISO/IEC 14496 scene graph. An ISO/IEC 14496 scene is understood as a composition of Audio-Visual objects according to some spatial and temporal relationships. The scene graph is the hierarchical representation of the ISO/IEC 14496 scene structure (see ISO/IEC 14496-1:1999).

Alternatively, an FBA object can be formed by a temporal sequence of FBA object plane groups (called segments for simplicity), where each FBA object plane group itself is composed of a temporal sequence of 16 FBA object planes, as depicted in the following:

FBA object:**FBA object plane group:**

When the alternative FBA object bitstream structure is employed, the bitstream is decoded by DCT-based FBA object decoding as described in subclause 7.12.2. Otherwise, the bitstream is decoded by the frame-based FBA object decoding. Refer to Table C-1 for a specification of default minimum and maximum values for each FAP and BAP.

6.1.5.2 Facial animation parameter set

The FAPs are based on the study of minimal facial actions and are closely related to muscle actions. They represent a complete set of basic facial actions, and therefore allow the representation of most natural facial expressions. Exaggerated values permit the definition of actions that are normally not possible for humans, but could be desirable for cartoon-like characters.

The FAP set contains two high level parameters visemes and expressions. A viseme is a visual correlate to a phoneme. The viseme parameter allows viseme rendering (without having to express them in terms of other parameters) and enhances the result of other parameters, insuring the correct rendering of visemes. Only static visemes which are clearly distinguished are included in the standard set. Additional visemes may be added in future extensions of the standard. Similarly, the expression parameter allows definition of high level facial expressions. The facial expression parameter values are defined by textual descriptions. To facilitate facial animation, FAPs that can be used together to represent natural expression are grouped together in FAP groups, and can be indirectly addressed by using an expression parameter. The expression parameter allows for a very efficient means of animating faces. In annex C, a list of the FAPs is given, together with the FAP grouping, and the definitions of the facial expressions.

6.1.5.3 Facial animation parameter units

All the parameters involving translational movement are expressed in terms of the *Facial Animation Parameter Units (FAPU)*. These units are defined in order to allow interpretation of the FAPs on any facial model in a consistent way, producing reasonable results in terms of expression and speech pronunciation. They correspond to fractions of distances between some key facial features and are defined in terms of distances between feature points. The fractional units used are chosen to allow enough precision. annex C contains the list of the FAPs and the list of the FDP feature points. For each FAP the list contains the name, a short description, definition of the measurement units, whether the parameter is unidirectional (can have only positive values) or bi-directional, definition of the direction of movement for positive values, group number (for coding of selected groups), FDP subgroup number (annex C) and quantisation step size. FAPs act on FDP feature points in the indicated subgroups. The measurement units are shown in Table 6-1, where the notation 3.1.y represents the y coordinate of the feature point 3.1; also refer to Figure 6-10.

Table 6-1 -- Facial Animation Parameter Units

Description		FAPU Value
$IRISD0 = 3.1.y - 3.3.y = 3.2.y - 3.4.y$	Iris diameter (by definition it is equal to the distance between upper and lower eyelid) in neutral face	$IRISD = IRISD0 / 1024$
$ES0 = 3.5.x - 3.6.x$	Eye separation	$ES = ES0 / 1024$
$ENS0 = 3.5.y - 9.15.y$	Eye - nose separation	$ENS = ENS0 / 1024$
$MNS0 = 9.15.y - 2.2.y$	Mouth - nose separation	$MNS = MNS0 / 1024$
$MW0 = 8.3.x - 8.4.x$	Mouth width	$MW = MW0 / 1024$
AU	Angle Unit	10^{-5} rad

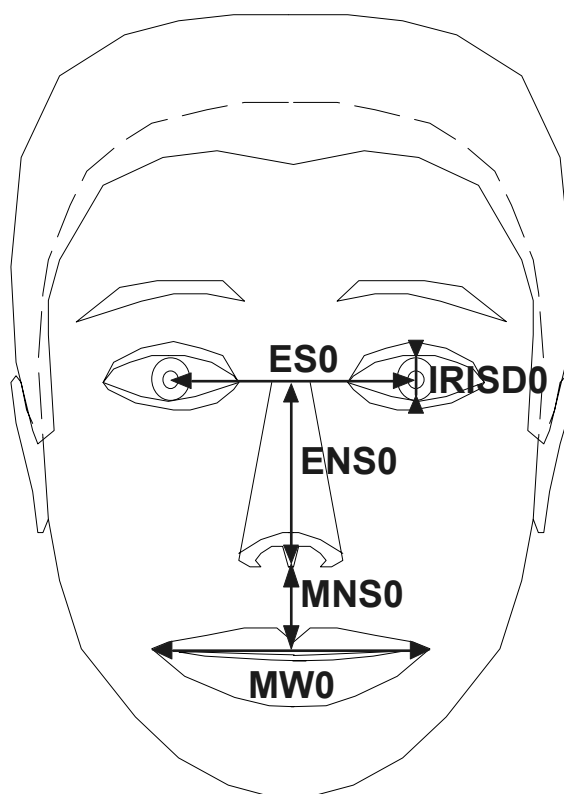


Figure 6-10 -- The Facial Animation Parameter Units

6.1.5.4 Description of a neutral face

At the beginning of a sequence, the face is supposed to be in a neutral position. Zero values of the FAPs correspond to a neutral face. All FAPs are expressed as displacements from the positions defined in the neutral face. The neutral face is defined as follows:

- the coordinate system is right-handed; head axes are parallel to the world axes
- gaze is in direction of Z axis
- all face muscles are relaxed
- eyelids are tangent to the iris
- the pupil is one third of IRISD0
- lips are in contact; the line of the lips is horizontal and at the same height of lip corners
- the mouth is closed and the upper teeth touch the lower ones
- the tongue is flat, horizontal with the tip of tongue touching the boundary between upper and lower teeth (feature point 6.1 touching 9.11 in annex C)

6.1.5.5 Facial definition parameter set

The FDPs are used to customize the proprietary face model of the decoder to a particular face or to download a face model along with the information about how to animate it. The definition and description of FDP fields is given in annex C. The FDPs are normally transmitted once per session, followed by a stream of compressed FAPs. However, if the decoder does not receive the FDPs, the use of FAPUs ensures that it can still interpret the FAP stream. This insures minimal operation in broadcast or teleconferencing applications. The FDP set is specified in BIFS syntax (see ISO/IEC 14496-1:1999). The FDP node defines the face model to be used at the receiver. Two options are supported:

- calibration information is downloaded so that the proprietary face of the receiver can be configured using facial feature points and optionally a 3D mesh or texture.
- a face model is downloaded with the animation definition of the Facial Animation Parameters. This face model replace the proprietary face model in the receiver.

6.1.5.6 Body animation parameter set

BAP parameters comprise joint angles connecting different body parts. These include: toe, ankle, knee, hip, spine (C1-C7, T1-T12, L1-L5), shoulder, clavicle, elbow, wrist, and the hand fingers. The detailed joint list, with the rotation normals, are given in the following subclause. The rotation angles are assumed to be positive in the counterclockwise rotation direction with respect to the rotation normal. The rotation angles are defined as zero in the default posture, as defined below.

Note that the normals of rotation move with the body, and they are fixed with respect to the parent body part. That is to say, the axes of rotation are not aligned with the body or world coordinate system, but move with the body parts.

The hands are capable of performing complicated motions and are included in the body hierarchy. There are totally 29 degrees of freedom on each hand, assuming that the hand has a standard structure with five fingers.

The unit of rotations (BAPU) is defined as $\pi/10E-5$ radians. The unit of translation BAPs (BAPs HumanoidRoot_tr_vertical, HumanoidRoot_tr_lateral, HumanoidRoot_tr_frontal) is defined in millimeters.

6.1.5.7 Description of the default posture of the body

The default posture is defined by standing posture. This posture is defined as follows: the feet should point to the front direction, the two arms should be placed on the side of the body with the palm of the hands facing inward. This posture also implies that all BAPs have default values as 0.

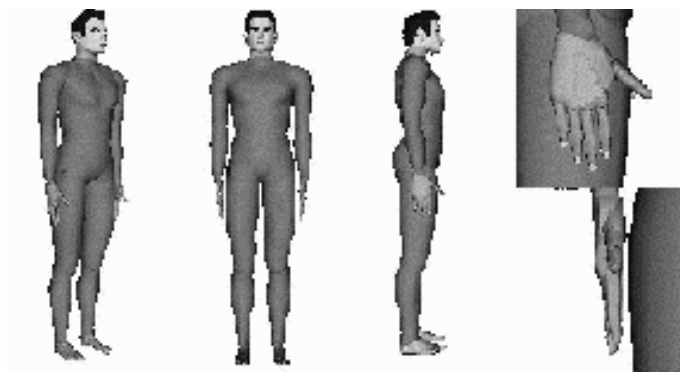


Figure V2 - 5

6.1.5.8 Body Definition Parameter Set

The BDPs are used to customize the proprietary body model of the decoder to a particular body or to download a body model along with the information about how to animate it. The definition and description of BDP fields is given in annex C. The BDPs are normally transmitted once per session, followed by a stream of compressed BAPs. However, if the decoder does not receive the BDPs, model-independence of BAPs ensures that it can still interpret the BAP stream. This insures minimal operation in broadcast or teleconferencing applications. The BDP set is specified in BIFS syntax (see ISO/IEC 14496-1:1999/Amd.1:2000). The BDP node defines the body model to be used at the receiver.

6.1.6 3D Mesh Object

The compressed bitstream for a 3D mesh is composed of a header data block with global information, followed by a sequence of connected component data blocks, each one associated with one connected component of the 3D mesh.

3D Mesh Header	CC Data #1	...	CC Data #nCC
----------------	------------	-----	--------------

If a 3D mesh is coded in error resilience mode, connected component data blocks are grouped or divided into partitions.

Partition #1	Partition #2	...	Partition #nPT
--------------	--------------	-----	----------------

Additionally, if the 3D mesh is represented in hierarchical mode, the last connected component data block is followed by one or more refinement step data blocks, each one of them representing a Forest Split operation. Forest split can be applied to all the components of a bitstream.

Forest Split #1	Forest Split #2	...	Forest Split #nFS
-----------------	-----------------	-----	-------------------

Each connected component data block is composed of three records, the Vertex Graph record, the Triangle Tree record, and the Triangle Data record.

Vertex Graph	Triangle Tree	Triangle Data
--------------	---------------	---------------

The triangle tree record contains the structure of a triangle spanning tree which links all the triangles of the corresponding connected component forming a simple polygon. The 3D mesh is represented in a triangulated form in the bitstream, which also contains the information necessary to reconstruct the original faces. The vertex graph record contains the information necessary to stitch pairs of boundary edges of the simple polygon to reconstruct the original connectivity, not only within the current connected component, but also to previously decoded connected components. The connectivity information is categorized as *global* information (per connected component) and *local* information (per triangle). The global information is stored in the Vertex Graph and Triangle Tree records. The local information is stored in the Triangle Data record. The triangle data is arranged on a per triangle basis, where the ordering of the triangles is determined by the traversal of the triangle tree.

Data for triangle #1	Data for triangle #2	...	Data for triangle #nT
----------------------	----------------------	-----	-----------------------

The data for a given triangle is organized as follows:

marching edge	td_orientation	polygon_edge	coord	normal	color	texCoord
---------------	----------------	--------------	-------	--------	-------	----------

The *marching edge*, *td_orientation* and *polygon_edge* constitute the per triangle connectivity information. The other fields contain information to reconstruct the vertex coordinates (*coord*) and optionally, normal, color, and texture coordinate (*texCoord*) information.

If the 3D mesh is encoded in hierarchical mode, each Forest Split data block is composed of an optional pre-smoothing data block, an optional post-smoothing data block, a pre-update data block, an optional smoothing constraints data block, a post-update data block, and an optional other-update data block.

Pre-smoothing	Post-smoothing	Pre-update	Constraints	Post-update	Other-update
---------------	----------------	------------	-------------	-------------	--------------

The pre-smoothing data block contains the parameters used to apply a smoothing step as a global predictor for vertex coordinates. The post-smoothing data block contains the parameters used to apply a smoothing step to the vertex coordinates to remove quantisation artifacts. The pre update data block contains the information necessary to update the connectivity, and the property data updates for properties bound per-face and per-corner to the new faces created by the connectivity update. The smoothing constraints data block contains information used to perform the smoothing steps with sharp edge discontinuities and fixed vertices. After the connectivity update is applied, the pre-smoothing operation specified by the parameters stored in the pre-smoothing data block is applied as a global predictor for the vertex coordinates. The post update data block contains tree loop vertex coordinate updates, with respect to the vertex coordinates predicted by the pre-smoothing step, if applied; normal, color, and texture coordinate updates for properties bound per-vertex to tree loop vertices; normal, and color updates for properties bound per-face to tree loop faces; and normal, color, and texture coordinate updates for properties bound per-corner to tree loop corners. The other-update data block contains vertex coordinate updates and property updates for all the vertices, faces, and corners not updated by data included in the post-update data block.

6.2 Visual bitstream syntax

6.2.1 Start codes

Start codes are specific bit patterns that do not otherwise occur in the video stream.

Each start code consists of a start code prefix followed by a start code value. The start code prefix is a string of twenty three bits with the value zero followed by a single bit with the value one. The start code prefix is thus the bit string '0000 0000 0000 0000 0000 0001'.

The start code value is an eight bit integer which identifies the type of start code. Many types of start code have just one start code value. However *video_object_start_code* and *video_object_layer_start_code* are represented by many start code values.

All start codes shall be byte aligned. This shall be achieved by first inserting a bit with the value zero and then, if necessary, inserting bits with the value one before the start code prefix such that the first bit of the start code prefix is the first (most significant) bit of a byte. For stuffing of 1 to 8 bits, the codewords are as follows in Table 6-2.

Table 6-2-- Stuffing codewords

Bits to be stuffed	Stuffing Codeword
1	0
2	01
3	011
4	0111
5	01111
6	011111
7	0111111
8	01111111

Table 6-3 defines the start code values for all start codes used in the visual bitstream.

Table 6-3 — Start code values

name	start code value (hexadecimal)
video_object_start_code	00 through 1F
video_object_layer_start_code	20 through 2F
reserved	30 through AF
visual_object_sequence_start_code	B0
visual_object_sequence_end_code	B1
user_data_start_code	B2
group_of_vop_start_code	B3
video_session_error_code	B4
visual_object_start_code	B5
vop_start_code	B6
reserved	B7-B9
fba_object_start_code	BA
fba_object_plane_start_code	BB
mesh_object_start_code	BC
mesh_object_plane_start_code	BD
still_texture_object_start_code	BE
texture_spatial_layer_start_code	BF
texture_snr_layer_start_code	C0
texture_tile_start_code	C1
texture_shape_layer_start_code	C2
stuffing_start_code	C3
reserved	C4-C5
System start codes (see note)	C6 through FF
NOTE System start codes are defined in ISO/IEC 14496-1:1999	

The use of the start codes is defined in the following syntax description with the exception of the video_session_error_code. The video_session_error_code has been allocated for use by a media interface to indicate where uncorrectable errors have been detected.

This syntax for visual bitstreams defines two types of information:

1. Configuration information

- a. Global configuration information, referring to the whole group of visual objects that will be simultaneously decoded and composited by a decoder (VisualObjectSequence()).
- b. Object configuration information, referring to a single visual object (VO). This is associated with VisualObject().
- c. Object layer configuration information, referring to a single layer of a single visual object (VOL) VideoObjectLayer().

2. Elementary stream data, containing the data for a single layer of a visual object.

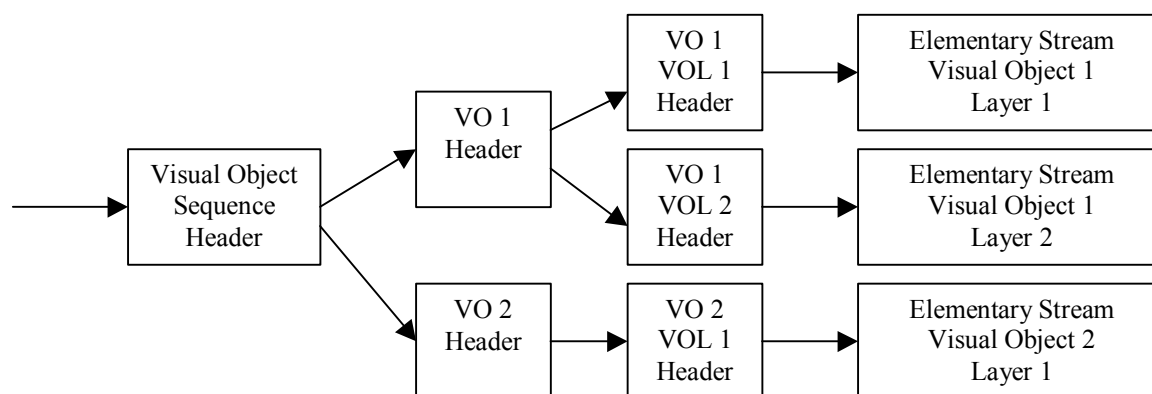


Figure 6-11 -- Example Visual Information – Logical Structure

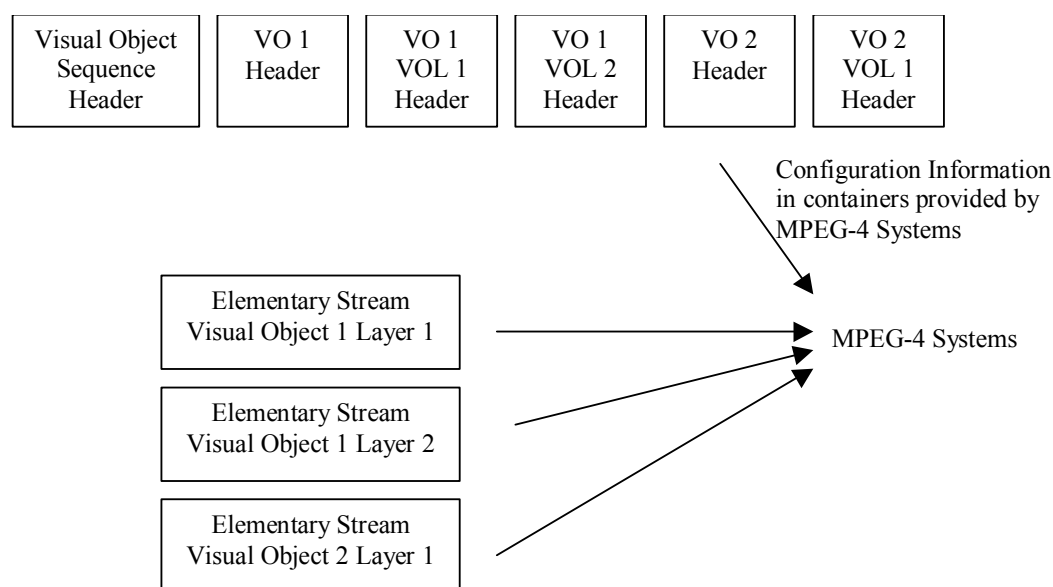


Figure 6-12 -- Example Visual Bitstream – Separate Configuration Information / Elementary Stream

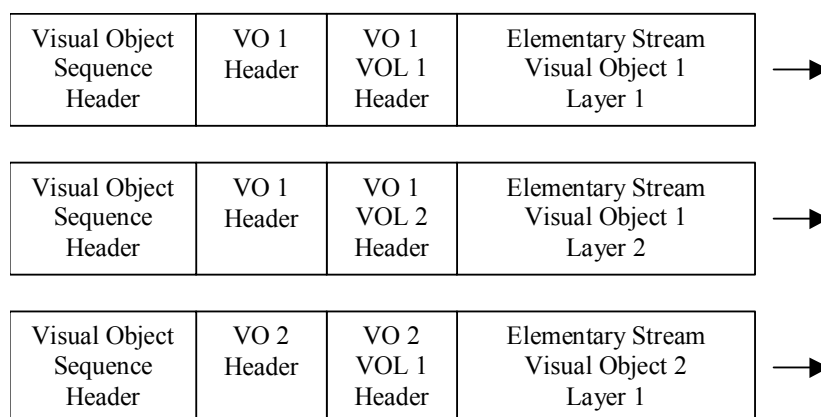


Figure 6-13 -- Example Visual Bitstream – Combined Configuration Information / Elementary Stream