Chapter 6

MPEG-4 - Standard for Multimedia Applications

6.1 Introduction

After having completed the tasks of setting two highly acclaimed international standards for audiovisual applications, i.e.

- MPEG-1 (ISO/IEC 11172), a standard for storage and retrieval of moving pictures and audio on storage media
- MPEG-2 (ISO/IEC 13818), a standard for digital television.

MPEG is now working to produce MPEG-4, a standard for multimedia applications, scheduled for the status of International Standard in December 1998 with the ISO number 14496 [1]. This is prompted by a need for a new standard with the convergence of the three separate technologies in the fields of entertainment and communications (e.g., digital television), interactive graphics applications (e.g., animation) and interactive multimedia applications (e.g., World Wide Web). There are clearly commonalities in the technologies with regard to the production, distribution and access of content and MPEG-4 aims to provide the standardized elements enabling the integration of the three processes within a common framework.

6.2 MPEG-4 Development Process

MPEG-4 work started in July 1993. The first Call for Proposal was made two and a half years later after the definition of the scope has been achieved.

All interested parties, whether within or outside MPEG, are invited to participate so as to attract state-of-the-arts technology to be considered for incorporation into the new standard. After that first call, other calls were issued for other technology areas.

The proposals of technology received were assessed and, if found promising, incorporated in the so-called Verification Models (VMs). A VM describes, in text and some sort of programming language, the operation of encoder and decoder. VMs are used to carry out simulations with the aim to optimize the performance of the coding schemes.

It is envisaged that with the rapid advance of computer processing power, software platforms will gain increasing importance in the implementation of the standard. Therefore MPEG decided to maintain software implementation of the different parts of the standard which can be used for the purpose of testing in the development process and for commercial implementations.

When sufficient confidence has been achieved in the stability of the standard under development, a Working Draft (WD) is produced which may undergo several revisions. The WDs already had the structure and form of a standard but they were kept internal to MPEG for revision. The WD then becomes a Committee Draft (CD) which undergoes a formal ballot by National Bodies (NBs). Ballots by NBs are usually accompanied by technical comments. If the number of positive votes is more than 2/3 of the total, the CDs will become Final CDs or FCDs.

The FCDs will be sent again to the National Bodies for a second ballot, the outcome of which will be considered with a similar process as for the CD stage. After that, the FCD becomes a Final Draft International Standard (FDIS). It will then be sent to National Bodies for a final ballot where NBs are only allowed to cast a yes/no ballot without comments. If the number of positive votes is above 75%, the DIS will become International Standard (IS) and is sent to the ISO Central Secretariat for publication.

6.3 Features of the MPEG-4 Standard [2]

Unlike the previous video coding standards where the coding unit is frame or set of frames, MPEG-4 adopts a different content-based approach [2]. The shift in paradigm is prompted by the realisation that the traditional coding techniques do not take into account the semantic properties of the image content coded and therefore result in objectionable artefacts such as "blocking" artefacts. Moreover, the overriding need of a new standard to provide new functionalities renders the current standards inappropriate.

MPEG-4 addresses the following content-based functionalities:

- content-based interactivity
  - content-based multimedia data access tools
  - content-based manipulation and bitstream editing
  - hybrid natural and synthetic data coding
  - improved temporal random access
- compression
  - improved coding efficiency
  - coding of multiple concurrent data streams
- universal access
  - robustness in error-prone environments
  - content-based scalability

MPEG-4 achieves these goals by providing standardized ways to:

1. represent units of aural, visual or audiovisual content, called audiovisual objects (AVOs) which can be of natural or synthetic origin;
2. describe the composition of these objects to create compound AVOs that form audiovisual scenes;
3. multiplex and synchronize the data associated with AVOs, so that they can be transported over network channels providing a QoS appropriate for the specific AVOs; and
4. interact with the audiovisual scene generated at the receiver's end.

6.3.1 Coded Representation of Primitive AVOs

With reference to Fig. 6.1, the audiovisual scene is made up of AVOs which can be classified into primitive and compound AVOs. Examples the primitive AVOs are the talking person, the voice associated with the person, the background, the desk, etc. The talking person together the voice constitute a compound AVO. MPEG-4 defines the coded representation of the AVOs, be it natural or synthetic, 2- or 3-dimensional. To satisfy the above stated functionalities, the coding algorithm must be more efficient than the current standards and the coded representation allows the AVOs to be handled and accessed independently and interactively.

6.3.2 Composition of AVOs

MPEG-4 provides a standardized way to describe a scene, allowing the user to:

- place AVOs anywhere in a given coordinate system;
- apply transforms to change the geometrical or acoustical appearance of a AVO;
- group primitive AVOs in order to form compound media objects;
- apply streamed data to AVOs, in order to modify their attributes;
- change interactively the user's viewing and listening points anywhere in the scene.

Again, with reference to Fig. 6.1, for example, one can replace the person with a different person, changes her dress or hairstyle; group the desk and the globe to form a compound AVO since they are static; or change the background using a different sprite. The above features are made possible by a scene description tool which builds on several concepts from VRML in terms of both its structure and the functionality of object composition nodes.

6.3.3 Description, Synchronization and Delivery of Streaming Data for AVOs

The MPEG-4 System Layer Model is depicted in Fig. 6.2. The AVOs are delivered as streaming data in one or more *elementary streams.* Each AVO is identified by an *object descriptor* in the elementary streams, and each stream is characterised by a set of descriptors conveying the configuration and quality of service (QoS) information. These enable the decoder to determine the required resources and the precise timing for decoding, and the QoS the encoder requests for transmission.

In the *synchronization layer*, the elementary streams are packetized into access units and each access unit is timestamped for synchronization. Independent of the media type, this layer allows identification of access units in elementary streams, recovery of the AVOs or scene description's time base and enables synchronization among them.

The *delivery layer* contains a two-layer multiplexer. The first multiplexer layer known as the "FlexMux" layer, is defined according to the DMIF specification (part 6 of MPG-4 standard). This allows grouping of

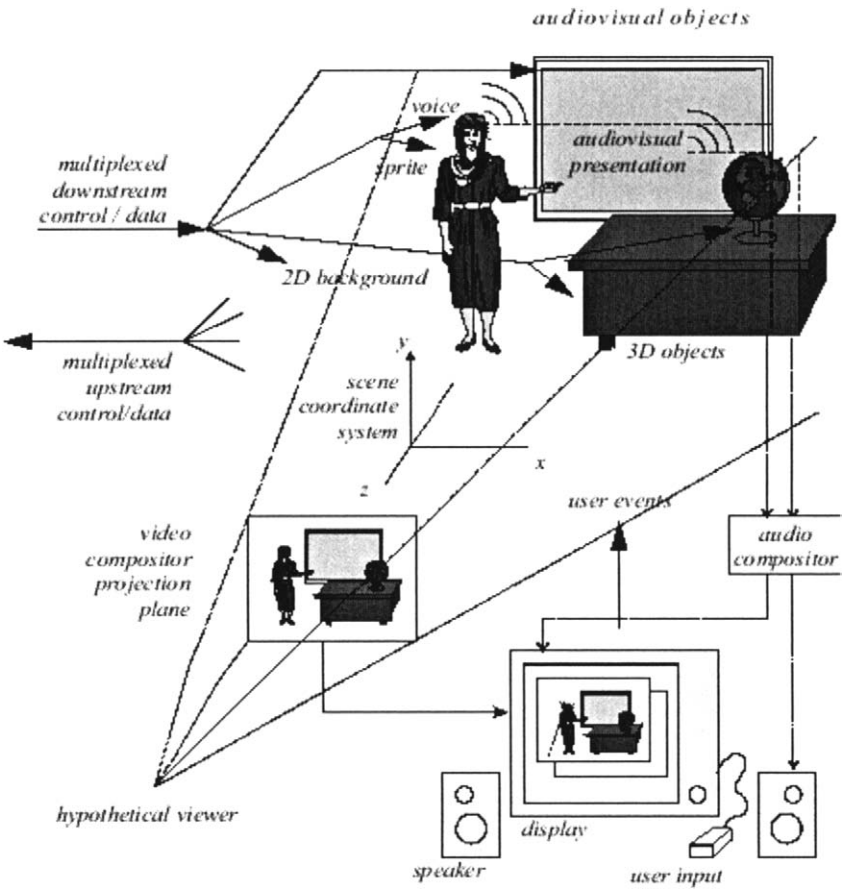


Figure 6.1: An example of an MPEG-4 audiovisual scene. ©ISO/IEC 1998

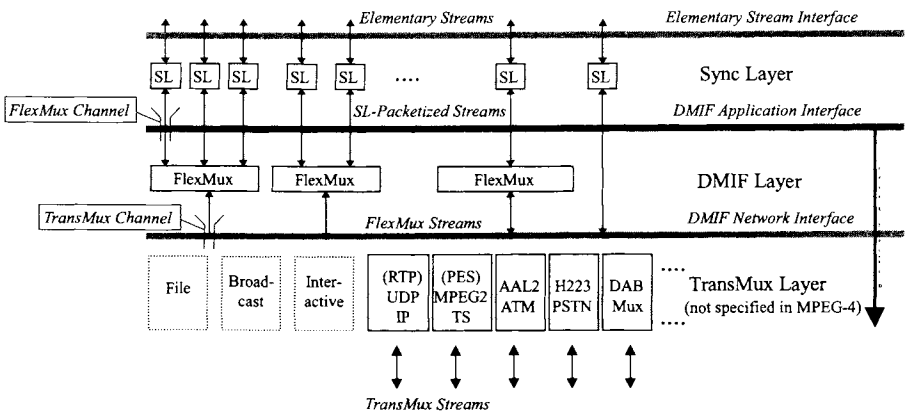


Figure 6.2: The MPEG-4 System Layer Model. ©ISO/IEC 1998

the elementary streams with a low multiplexing overhead. The "TransMux" layer offers transport services matching the requested QoS. Note that only the interface to this layer (i.e., DMIF Network Interface) is specified by MPEG-4 while the mapping of the data packets and control signalling are to be provided by the respective transport protocol offering the service.

Use of the FlexMux multiplexing tool is optional but the synchronization layer must always be present. Hence with the MPEG-4 System Layer model as shown in Fig. 6.2, the following functionalities can be provided:

1. identify access units, transport timestamps and clock reference information and identify data loss;

2. optionally interleave data from different elementary streams into FlexMux streams;

3. convey control information to

   - indicate the required QoS for each elementary stream and FlexMux stream;
   - translate such QoS requirements into actual network resources;
   - associate elementary streams to media objects;
   - convey the mapping of elementary streams to FlexMux and TransMux channels

6.3.4 Interaction with AVOs

MPEG-4 provides functionalilties for the user to interact with the audiovisual scene. Depending on the extent allowed by the author of the scene, the user has the ability to:

- change the viewing/listening point of the scene;
- drag objects in the scene to a different position;
- trigger a cascade of events by clicking on a specific object;
- select the desired language when multiple language tracks are available;
- trigger more complex kinds of behavior, e.g., answering a virtual phone.

6.3.5 Identification of Intellectual Property

It is important to have the possibility to identify intellectual property being coded into MPEG-4 media objects. Therefore, MPEG works with representatives of different creative industries in the definition of syntax and tools to support this. A full elaboration of the requirements for the identification of intellectual property can be found in MPEG97/N1918 [3], which is publicly available.

6.4 Technical Description of the MPEG-4 Standard

The MPEG-4 standard, as in the cases for MPEG-1 and MPEG-2 standards, defines an idealised decoding device together with the bitstream syntax and semantics in the form of a System Decoder Model. Fig. 6.3 shows the major components of an MPEG-4 decoder. As shown in Fig. 6.3, the MPEG-4 bitstreams are received as TransMux streams and demultiplexed into the FlexMux streams and then into the elementary streams. The elementary streams are parsed and passed to the appropriate decoders to recover the data in the AVOs and the scene description information needed for scene composition. The AVOs are composed into an audiovisual scene according to the scene description information as described by the author in the composition layer. The composed audiovisual scene is then rendered on the display or listening devices. The end user has the ability of interacting with the individual AVOs to the extend allowable by the author.

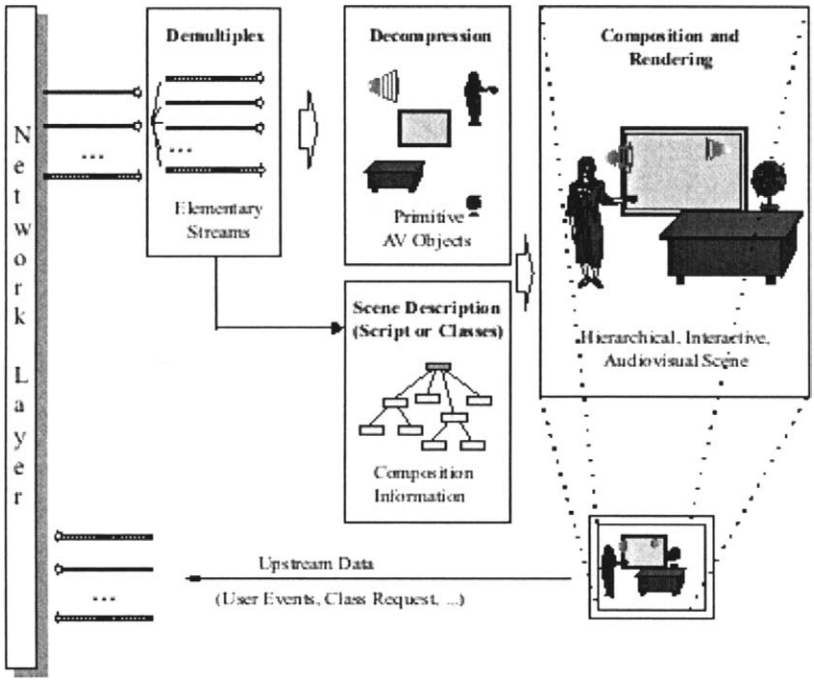


Figure 6.3: Major components of an MPEG-4 decoder. ©ISO/IEC 1998

For intellectual property rights (IPR) protection, the coded AVOs are supplemented with an optional Intellectual Property Identification (IPI) data set, carrying information about the right holders. The provision of the data sets allows the implementation of mechanisms for audit trailing, monitoring, billing and copy protection.

6.4.1 DMIF

The Delivery Multimedia Integration Framework (DMIF) is a session protocol for the management of multimedia streaming over generic networks. The DMIF architecture is such that applications which rely on DMIF for communication do not have to be concerned with the underlying communication method. The implementation of DMIF takes care of the delivery technology details presenting a simple interface to the application. Fig. 6.4 shows a conceptual model of the DMIF architecture. As shown in Fig. 6.4, the DMIF is located between the MPEG-4 application and the transport network.

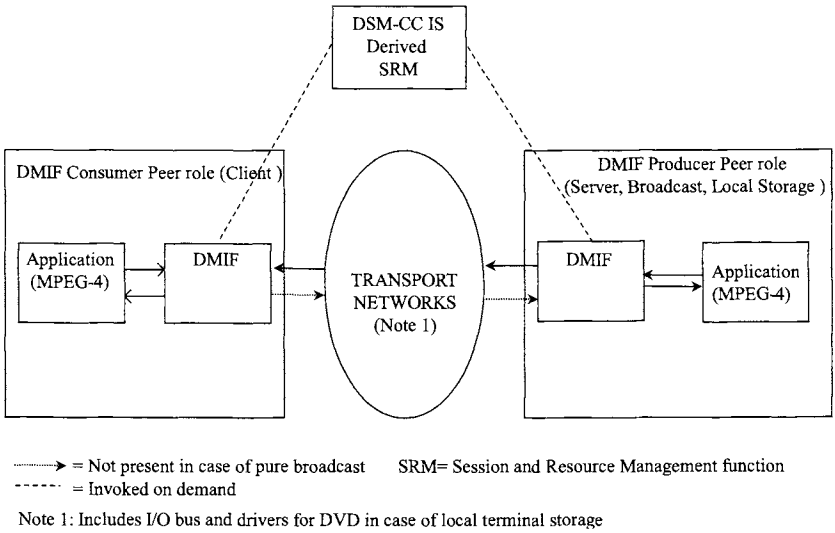


Figure 6.4: A conceptual model of the DMIF architecture. ©ISO/IEC 1998

DMIF presents a consistent interface to the application irregardless of whether MPEG-4 streams are received from a remote interactive peer over the transport networks and/or by interacting with broadcasting or storage media. An MPEG-4 application initiates a request for service through the DMIF to a another peer application or multiple peer applications. It may contain requests for certain QoS and specific channel bandwidth for that service. An interactive peer over a network may select a service, obtain a scene description and request specific streams for AVOs from the scene to be transmitted with appropriate QoS. DMIF ensures the timely establishment of channels with the specified QoSs and bandwidths over a variety of intervening networks between interactive peers. Control of DMIF spans both the FlexMux and TransMux layers as shown in Fig. 6.2 above. MPEG-4 offers a transparent interface with signalling primitive semantics at the interface to DMIF which are interpreted and translated into the appropriate protocols of each network. The exact maping for these translation are beyond the scope of MPEG-4 but left to the network provided.

The DMIF SRM functionality in Fig. 6.4 encompasses the MPEG-2 DSM-CC SRM functionality which is optional. However, DMIF provides a globally unique network session identifier which can be used to tag the resources and log their usage for subsequent billing.

6.4.2 Demultiplexing, Sychronization and Buffer Management

MPEG-4 defines a System Decoder Model along with the bitstream syntax and semantics that enables a compliant MPEG-4 decoder to decode the bitstream successfully. It provides the precise definition of the terminal's operation without making unnecessary assumptions about implementation details.

6.4.2.1 Demultiplexing

The incoming data from some network connection or a storage device have to be demultiplexed to retrieve the individual elementary streams. Demultiplexing occurs on the delivery layer that is modeled as consisting of a TransMux layer and a DMIF layer. The data retrieval consists of two tasks. Firstly, the channels must be located and opened which requires a transport control entity that manages, among others, the tables that associate transport channels to specific elementary streams. The linking of each stream to the actual channel as well as the management of the sessions and channels is handled in the DMIF layer.

Secondly, the incoming streams must be properly demultiplexed to recover SL-packetized streams from downstream channels to be passed on to the synchronization layer. In interactive applications, a corresponding multiplexing stage will multiplex upstream data in upstream channels.

The Multiplex layer is to abstract any underlying multiplex functionality that is suitable to transport MPEG-4 data streams. Note that this layer is not defined in MPEG-4. The TransMux Layer is assumed to provide protection and multiplexing functionality, indicating that this layer is responsible for offering a specific QoS.

On the other hand, the FlexMux tool is specified by MPEG-4 to optionally provide a flexible, low delay, low overhead method for interleaving data when the packet size or overhead of the underlying protocol stack is large. The FlexMux requires reliable error detection and sufficient framing of FlexMux packets (for random access and error recovery) from the underlying layer.

6.4.2.2 Sychronization

The sync layer has a minimum set of tools for consistency checking, padding, to convey time base information and to carry time stamped access units of an elementary stream. Time stamps are used to convey the nominal decoding

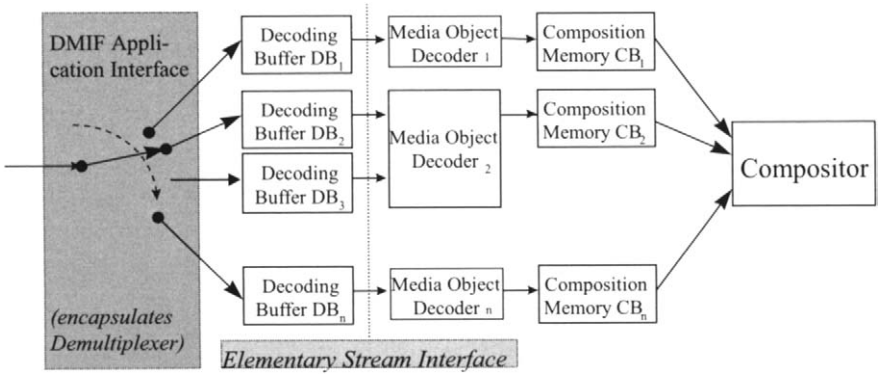


Figure 6.5: Buffer architecture of the System Decoder Model. ©ISO/IEC 1998

and composition time for an access unit. The sync layer requires reliable error detection and framing of each individual packet from the underlying FlexMux layer.

To be able to relate elementary streams to media objects within a scene, object descriptors are used which themselves are conveyed in one or more elementary streams. Object descriptors convey information about the number and properties of elementary streams that are associated to particular AVOs.

6.4.2.3 Buffer Management

To predict how the decoder will behave when it decodes the various elementary data streams that form an MPEG-4 session, the Systems Decoder Model enables the encoder to specify and monitor the minimum buffer resources that are needed to decode a session. The required buffer resources are conveyed to the decoder within object descriptors during the setup of the MPEG-4 session, so that the decoder can decide whether it is capable of handling this session. Fig. 6.5 shows the buffer architecture of the System Decoder Model.

6.4.2.4 Time Identification

For real time operation, a timing model is assumed in which the end-to-end delay from the signal output from an encoder to the signal input to a decoder is constant. Furthermore, the transmitted data streams must

contain implicit or explicit timing information. There are two types of timing information. The first is used to convey the speed of the encoder clock, or time base, to the decoder. The second, consisting of time stamps attached to portions of the encoded AV data, contains the desired decoding time for access units or composition and expiration time for composition units. This information is conveyed in SL-packet headers generated in the sync layer. With this timing information, the inter-picture interval and audio sample rate can be adjusted at the decoder to match the encoder's inter-picture interval and audio sample rate for synchronized operation.

6.4.3 Syntax Description

MPEG-4 defines a syntactic description language to describe the exact binary syntax of a bit stream conveying data for a media object representation as well as that of the scene description information. This language is an extension of C++, and is used to describe the syntactic representation of objects and the overall media object class definitions and scene description information in an integrated way.

6.5 Coding of Audio Objects

MPEG-4 provides tools for coding of natural sounds (e.g. speech and music) and synthesized sounds from structured descriptions. The synthesized sounds can be derived from text data and instrument descriptions. The representations can be compressed and provide effects such as reverberation and spatialization. The audio coding tools covering bit rates from 6 kbits/s to 24 kbits/s are aimed at applications in AM digital audio broadcasting to provide improvements over the existing AM modulation services. Several codecs (e.g. CELP, Twin VQ and ACC) have been tested and compared to a reference AM system and found to give superior performance.

6.5.1 Natural Sound

For natural sound coding, MPEG-4 supports bit rates from 2 kbits/s up to 64 kbits/s. Coding at less than 2 kbits/s is also supported for variable rate coding. The inclusion of MPEG-2 AAC standard in MPEG-4 provides coding of higher quality audio at higher bit rates.

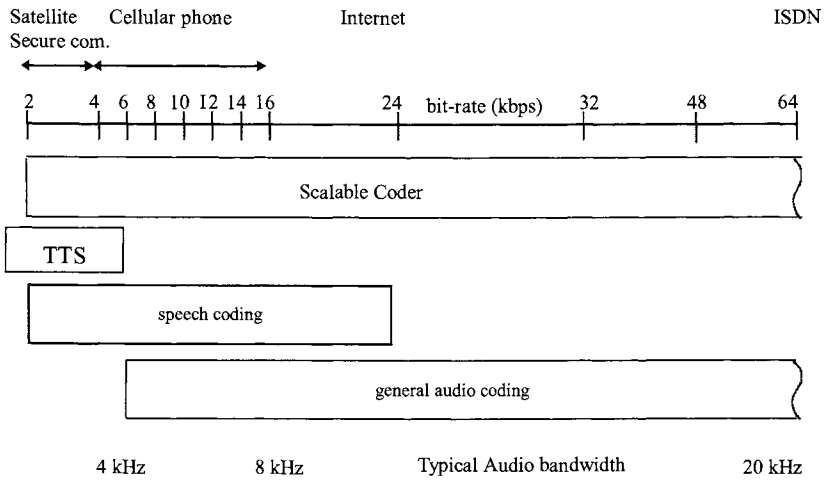


Figure 6.6: MPEG-4 audio coding framework. ©ISO/IEC 1998

6.5.1.1 Speech Coding

For bit rates between 2 to 4 kbits/s, Harmonic Vector eXcitation Coding (HVXC) is used; whilst at higher bit rates of between 4 to 24 kbits/s, Code Excited Linear Predictive (CELP) Coding is employed. HVXC can also operate at a lower bit rate of 1.2 kbits/s when coding in variable bit rate mode. In order to support narrowband and wideband speech, sampling rates of 8 and 16 kHz, respectively, are used.

6.5.1.2 Audio Coding

Transform coding techniques, such as Twin VQ and AAC, are employed to code audio at ≥ 6 kbits/s with a typical sampling rate of 8 kHz.

Fig. 6.6 shows the MPEG-4 audio coding framework.

6.5.1.3 Audio Scalability

Scalability in MPEG-4 can be in terms of:

- bit rate scalability which allows a bitstream to be parsed into a bitstream of lower bit rate to provide audio of lower quality;
- bandwidth scalability where part of the bitstream representing a part of the spectrum is discarded;

- encoder/decoder complexity scalability that allows encoder/decoder of different complexities to generate meaningful bitstreams.

Scalability tools can be applied to a combination of techniques, e.g., with CELP as a base layer and AAC as an enhancement layer.

6.5.2 Synthesized Sound

Decoders can synthesize sounds from structured inputs. Speech is converted from text input in the Text-to-Speech (TTS) coder, and more general sounds including music may be normatively synthesized with extremely low bit rate.

6.5.2.1 Text-to-Speech

MPEG-4 provides an interface for a TTS coder which allows the generation of intelligible synthetic speech from a text or a text with prosodic parameters. TTS coders operate with bit rates ranging from 200 bits/s to 1.2 kbits/s. It supports tools that allow synchronization to associated face animation, international language (for text) and international symbols (for phonemes).

6.5.2.2 Score Driven Synthesis

For audio, the decoding tools are driven by a synthesis language called Structured Audio Orchestra Language (SAOL) which defines the instruments of an orchestra downloadable in the bitstream. An instrument consists of a cluster of signal processing primitives implemented either by hardware or software that emulate some specific sounds generated by an actual acoustic instrument. Note that MPEG-4 does not standardize a method of synthesis but rather one of "describing" the synthesis.

Synthesis is controlled by the use of "scores" which are a time-sequenced set of commands that invokes various instruments at specific times to compose an overall music performance. The scores are described by a language known as Structured Audio Score Language (SASL) that can create new sounds or modify existing sounds. Careful control of the synthesis process enables the synthesis of a wide range of audio and sound effects, from footsteps, wind blowing, waterfalls, to conventional music and synthesized electronic music.

MPEG-4 also standardizes a simpler synthesis technique, the "wavetable bank format" for terminals with less functionality or applications not requiring the more sophisticated synthesis described above. This format allows

the generation of sound samples from a downloadable wavetable. Simple processing such as filtering, reverberation, chorus effects are also possible.

6.6 Coding of Natural Visual Objects

Visual objects can be either of natural or of synthetic origin. First, the objects of natural origin are described.

The tools for representing natural video in the MPEG-4 visual standard aim at providing standardized core technologies allowing efficient storage, transmission and manipulation of textures, images and video data for multimedia environments. These tools will allow the decoding and representation of atomic units of image and video content, called "video objects" (VOs). An example of a VO could be a talking person (without background) which can then be composed with other AVOs (audio-visual objects) to create a scene. Conventional rectangular imagery is handled as a special case of such objects.

In order to achieve this broad goal rather than a solution for a narrow set of applications, functionalities common to several applications are clustered. Therefore, the visual part of the MPEG-4 standard [4] provides solutions in the form of tools and algorithms for:

- efficient compression of images and video
- efficient random access to all types of visual objects
- extended manipulation functionality for images and video sequences
- content-based coding of images and video
- content-based scalability of textures, images and video
- spatial, temporal and quality scalability
- error robustness and resilience in error prone environments

6.6.1 Video Object Plane (VOP)

6.6.1.1 VOP Definition

The Video Object (VO) correspond to entities in the bitstream that the user can access and manipulate (cut, paste...). Instances of Video Object in given time are called Video Object Plane (VOP). The encoder sends together with the VOP, composition information (using composition layer

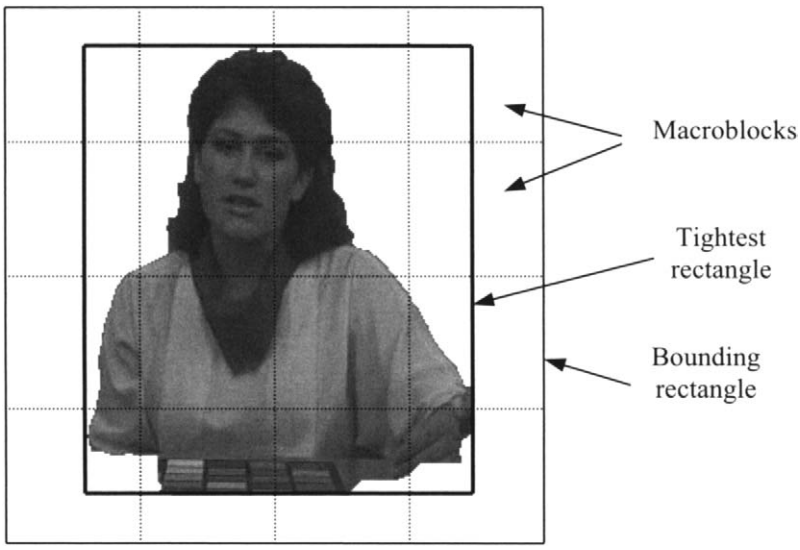


Figure 6.7: VOP Formation.

syntax) to indicate where and when each VOP is to be displayed. At the decoder side the user may be allowed to change the composition of the scene displayed by interacting on the composition information.

The VOP can be a semantic object in the scene: it is made of Y, U, V components plus shape information. When the sequence has only one rectangular VOP of fixed size displayed at fixed interval, it corresponds to the frame-based coding technique.

The exact method used to generate the VOP from the video sequences is not standardized in MPEG-4.

6.6.1.2 VOP Formation

The shape information is used to form a VOP. The VOP is formed by first drawing the tightest rectangle around the object. The rectangle is then extended to a bounding rectangle that contains a multiple of macroblocks as shown in Fig. 6.7. This ensures that the VOP contains a minimum number of macroblocks to represent the object.

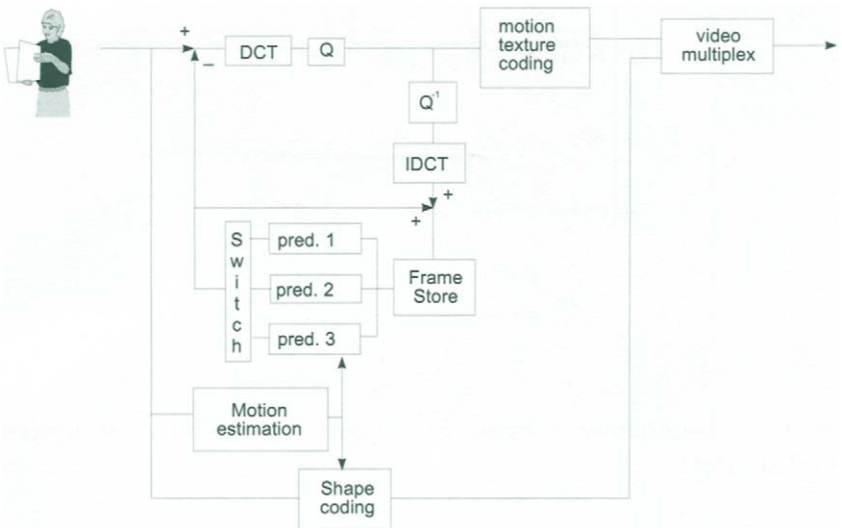


Figure 6.8: VOP encoder structure. ©ISO/IEC 1998

6.6.2 The Encoder

6.6.2.1 Overview

Fig. 6.8 presents a general overview of the VOP encoder structure. The same encoding scheme is applied when coding all the VOPs of a given session.

The encoder is mainly composed of two parts: the shape coder, and the traditional motion and texture coder applied to the same VOP. The phase between the luminance and chrominance samples of the bounding rectangle has to be correctly set according to the 4:2:0 format, as shown in Fig. 6.9. Specifically the top left coordinate of the bounding rectangle should be rounded to the nearest even number not greater than the top left coordinates of the tightest rectangle. Accordingly, the top left coordinate of the bounding rectangle in the chrominance component is that of the luminance divided by two.

The chrominance alpha plane is created from the luminance alpha plane by a conservative subsampling process. In the case of a binary alpha plane, this ensures that there is always a chroma sample where there is at least one luma sample inside the VOP.

Binary alpha plane: For each 2×2 neighborhood of luminance alpha

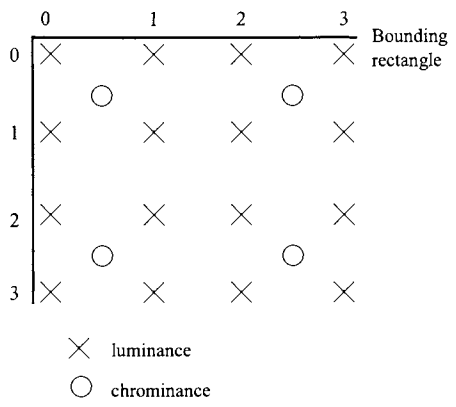


Figure 6.9: Luminance versus chrominance bounding box positioning. ©ISO/IEC 1998

pixels, the associated chroma alpha pixel is set to 255 if any of the four luminance alpha pixles are equal to 255.

Grayscale alpha plane: For each 2×2 neighborhood of luminance alpha pixels, the associated chroma alpha pixel is set to the rounded average of the four luminance alpha pixels.

6.6.3 Shape Coding

The shape information are in the form of binary or grey scale alpha planes as defined above. The binary alpha planes are encoded by modified context-based arithmetic encoding (CAE) whilst the grey scale alpha planes are encoded by motion compensated DCT similar to texture coding. An alpha plane is bounded by a rectangle that contains the shape of a VOP.

6.6.3.1 Binary Alpha Block Coding

The binary alpha plane contains 16×16 binary alpha blocks (BABs). The pixels in the BAB are raster-scanned and encoded by CAE. For I-VOPs, the BAB is encoded in INTRA mode whilst in P-VOPs, it may be encoded in INTRA or INTER mode.

The CAE Algorithm

The CAE encoding process begins by computing a context number of each pixel to be encoded. A probability table is then indexed according to the

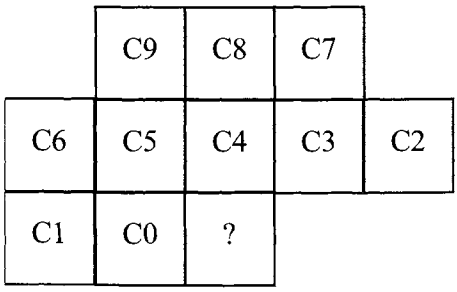


Figure 6.10: The INTRA template used in context computation. "?" denotes the pixel for which context is to be found. ©ISO/IEC 1998

computed context number. Finally the indexed probability is used to drive the arithmetic encoder. When the final pixel has been encoded, the process is terminated. The encoding process generates a single binary arithmetic codeword (BAC).

Context Computation

For INTRA coded BABs, the context for each pixel is given by

$$C = \sum_k c_k 2^k. \tag{6.1}$$

The positions of the pixels c_k are as illustrated in Fig. 6.10.

For INTER coded BABs, context computation exploits of the temporal redundancy provided by the motion-compensated BAB as depicted in Fig. 6.11.

BAB Borders

In calculating the context of each pixel of a BAB, pixels from neighboring BABs can be used. For INTRA and INTER cases, a border of 2 pixels in width is used as shown in Fig. 6.12. The top and left borders contains pixels from previously encoded and reconstructed BABs and the bottom and right borders contains zeroes which are unknown at decoding time.

BAB Encoding Decision

The BAB is encoded under all possible coding conditions both in INTRA and INTER modes. The coding condition which results in the shortest code is chosen to encode the BAB.

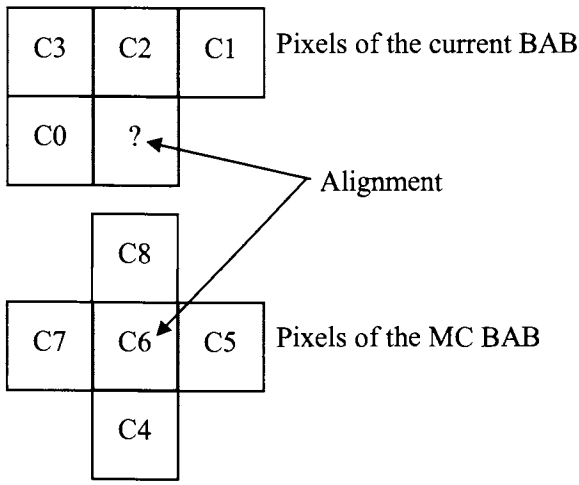


Figure 6.11: The INTER template used in context computation. "?" denotes the pixel for which context is to be found. ©ISO/IEC 1998

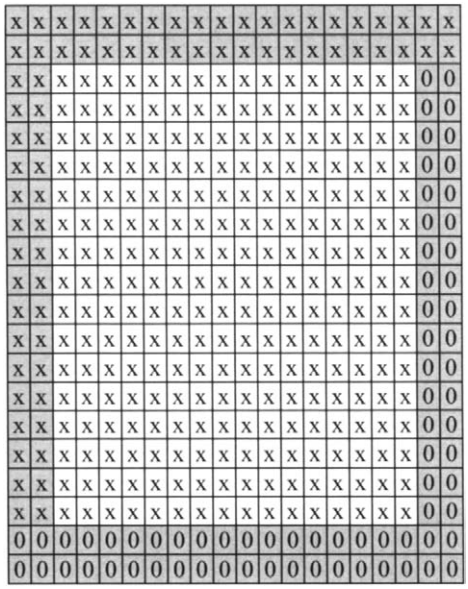


Figure 6.12: Current BAB and its borders (light grey area). ©ISO/IEC 1998

For I-VOPs, the BAB can be encoded in two ways:

- INTRA
- Transposition of the bordered BAB followed by INTRA

Similarly, for P-VOPs, the BABs can be enceode in the following ways:

- INTER
- Transposition of the bordered BAB and the MC BAB followed by INTER

The shortest codes from the INTRA and INTER modes are then compared and the coding condition which generates the shorter code is chosen.

6.6.3.2 Gray Scale Shape Coding

Gray scale alpha planes are employed for higher-quality content where each pixel belonging to a shape is assigned a value for its transparency. With this feature, transparency can differ from pixel to pixel of an object, and objects can be smoothly blended, either into a background or with other visual objects. An application of this is the superposition of a layer or layers of text or images (natural or synthetic) over a video sequence. This is particularly useful when a translucent effect is to be created, eg., subtitling, or composing synthetic objects with natural image background.

Support Function and Alpha Values Coding

Gray scale shape coding is performed by encoding the gray-level alpha plane in two separate parts: a binary alpha plane as its support function, and its alpha values as texture with arbitrary shape. The support is obtained by threshold the gray-level alpha plane by 0. The alpha values are encoded as 16×16 alpha macroblocks, the same way as luminanace in texture coding. The encoded alpha macroblock data are appended to the end of the corresponding texture macroblock for transmission.

Feathering and Translucency Coding

For most cases, when the gray-scale alpha mask is used, the texture within the alpha mask is fairly simple. It could consists of just a constant gray level or a binary alpha mask with values around the edges tapering from 255 to 0 to provide a gradual merging to the background. The process of smoothing the edge for merging to the background is known as feathering. It is done using two approaches: a feathering algorithm and a feathering filter.

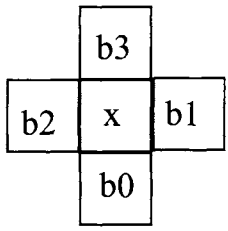


Figure 6.13: 3×3 filter kernel. ©ISO/IEC 1998

Feathering Algorithm

This algorithm tapers the alpha values of the pixels within a distance from the shape boundary linearly from the opaque alpha values to 0. Thus, the feathered alpha values are given by:

$$alpha = distance/(feather\_distance + 1) * opaque\_value \tag{6.2}$$

where *distance* is the distance of the pixel from the shape boundary, whereas $feature\_distance$ specifies the number of pixels to feather, and *opaque_value* is the value in the alpha mask.

6.6.3.3 Feathering Filter

A feathering filter is a 3×3 kernel as shown in Fig. 6.13 where x denotes the pixel to be feather and b1, b2 and b3 the surround pixels.

The filtering operation is as follows:

1. Label the alpha pixels of the input alpha mask to be 1 if it is 255, and 0 otherwise.

2. If $x = 0$, leave the pixel value intact. Otherwise, replace the pixel value with the value depending on its surrounding condition as given in Table 6.1.

3. The filtering operation is cascaded if the number of filtering iteration is more than 1. Note the the feathering filter can be different for each iteration and this is specified in the Video Object Layer.

Six modes of feathering operation can be identified in the Video Object Layer as represented by a 4-bit code as part of the VOL_descriptor:

Table 6.1: Feathering filter description table. ©ISO/IEC 1998

b3	b2	b2	b0	x'
0	0	0	0	x0
0	0	0	1	x1
0	0	1	0	x2
0	0	1	1	x3
0	1	0	0	x4
0	1	0	1	x5
0	1	1	0	x6
0	1	1	1	x7
1	0	0	0	x8
1	0	0	1	x9
1	0	1	0	x10
1	0	1	1	x11
1	1	0	0	x12
1	1	0	1	x13
1	1	1	0	x14

- no effects
- linear feathering
- constant alpha
- linear feathering and constant alpha
- feathering filter
- feathering filter and constant alpha

Linear filtering mode makes use of the feathering algorithm and feathering filter mode uses the feathering filter described above. For the linear feathering and feathering filter modes, the input can be a sequence of binary or grayscale alpha masks. For the constant alpha, linear feathering and constant alpha, and feathering filter and constant alpha modes, the input will be a sequence of grayscale alpha masks. If the mode selected is no effects, the default alpha mask compression algorithm specified will be utilized.

6.6.4 Motion Estimation and Compensation

Motion estimation is performed on a VOP basis. For the macroblocks on the VOP borders, motion estimation is modified from block matching to polygon matching. To achieve that, a special padding technique, i.e., the macroblock-based repetitive padding is used to pad the transparent region of the macroblocks in the reference VOP.

The absolute (frame) coordinate system is used for referencing all of the VOPs. At each particular time instance, a bounding rectangle is defined which includes the shape of that VOP. The left and top corner, in their absolute coordinates, of the bounding box is encoded in the VOP spatial reference. No alignment of VOP bounding boxes at different time instances is performed.

In addition to the motion estimation and compensation mode, two additional modes are supported, namely, unrestricted and advanced modes. In all three modes, the motion vector search range is $[-2^{f\_code+3}, -2^{f\_code+3} - 0.5]$ where $0 \leq f\_code \leq 7$. Unrestricted mode allows the motion vector to point outside the bounding box of the VOP. The advanced mode allows multiple motion vectors in one macroblock and overlapped motion compensation.

6.6.4.1 Padding Process

The padding process defines the values of luminance and chrominance samples outside the VOP for prediction of arbitrarily shaped objects. Fig. 6.14 shows a simplified diagram of this process.

A decoded macroblock $d[y][x]$ is padded by referring to the corresponding decoded shape block $s[y][x]$. The luminance component is padded per 16×16 samples, while the chrominance components are padded per 8×8 samples. A macroblock that lies on the VOP boundary (hereafter referred to as a *boundary macroblock*) is padded by replicating the boundary samples of the VOP towards the exterior. This process is divided into *horizontal repetitive padding* and *vertical repetitive padding.* The remaining macroblocks that are completely outside the VOP (hereafter referred to as *exterior macroblocks*) are filled by *extended padding.*

Horizontal and Vertical Padding

Horizontal repetitive padding is carried out by replicating each pixel at the boundary of a VOP horizontally to the left and/or right direction in order to fill the transparent region outside the VOP of a boundary macroblock. If there are two boundary pixels for filling a pixel position outside of a VOP,

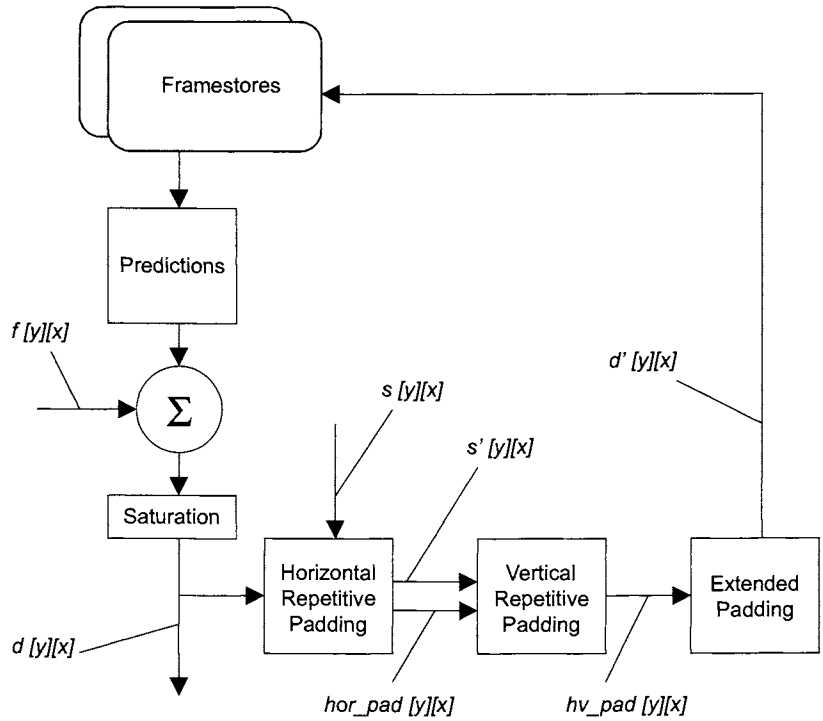


Figure 6.14: Simplified padding process. ©ISO/IEC 1998

the boundary pixel values are averaged. The remaining unfilled transparent horizontal samples are padded by a similar process as the horizontal repetitive padding but in the vertical direction. The samples already filled in are treated as if they were inside the VOP for the purpose of this vertical pass.

Extended Padding

Exterior macroblocks immediately next to boundary macroblocks are filled by replicating the samples at the border of the boundary macroblocks. If an exterior macroblock is next to more than one boundary macroblocks, If there are more one boundary macroblocks surrounding an exterior macroblock, the boundary macroblocks are numbered in priority according Fig. 6.15. The exterior macroblock is then padded by replicating upwards, downwards, leftwards, or rightwards the row of samples from the horizontal or vertical border of the boundary macroblock having the largest priority number. The remaining exterior macroblocks (not located next to any boundary macroblocks) are filled with 128.

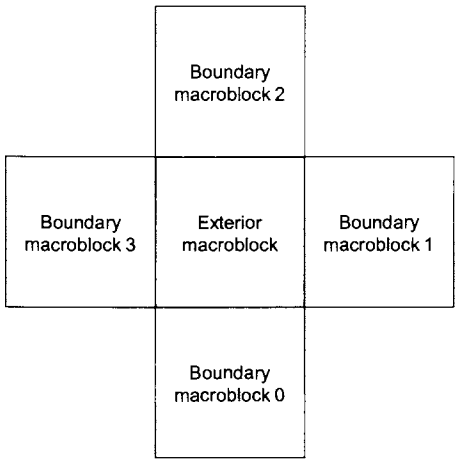


Figure 6.15: Priority of boundary macroblocks surrounding an exterior macroblock. ©ISO/IEC 1998

Padding for Chrominance Components

The above techniques for padding the luminance components are also applied to the chrominance components. The 8×8 chrominance blocks are generated by referring to a luminance shape block and decimating the shape block by a factor of 2 in the horizontal and vertical directions. The chrominance pixel is set to 1 if one of the 2×2 luminance shape samples is 1, otherwise it is set to 0.

Padding for Interlaced Macroblocks

The padding of interlaced macroblocks is the same as for the non-interlaced luminance and chrominance macroblocks except that it is performed for each field independently. A sample outside of a VOP is therefore filled with the value of the nearest boundary sample of the same field.

6.6.4.2 Basic Motion Estimation Techniques

Modified Block (Polygon) Matching

The principle of polygon matching is to use only the pixels within the VOP for motion estimation. To perform polygon matching, the bounding box of the VOP is first extended on the right-bottom corner to multiple of macroblocks, i.e., 16×16 for luminance VOPs, and 8×8 for chrominance

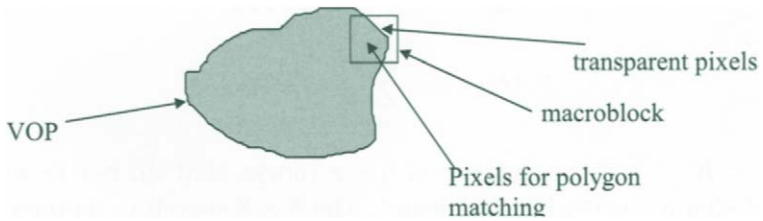


Figure 6.16: Polygon matching for an arbitrary shape VOP. ©ISO/IEC 1998

VOPs. The extended pixels and their associated alpha values are set to 0. The error criterion is the sum of absolute difference (SAD) computed using only the pixels with non-zero alpha value. The reference VOP is padded based on its own shape information. Fig. 6.16 illustrates an example.

Integer Pixel Motion Estimation

The motion estimation is performed by full search with a integer pixel accuracy within a maximum search area range specified by the *f_code*. The comparisons are made between the incoming block and the displaced block in the previous reconstructed VOP padded in a macroblock basis according to Section 6.6.4.1. The error measure is the SAD defined as:

$$SAD_N(x,y) = \sum_{i=1,j=1}^{N,N} |original - previous| * (Alpha_{original} \neq 0) \quad (6.3)$$

where $(x,y) = [-64,63]$, $n = 16$ or 8.

The SAD at the zero position $SAD_{16}(0,0)$ is reduced to favor the zero motion vector when there is no significant difference:

$$SAD_{16}(0,0) = SAD_{16}(0,0) - (N_B/2 + 1) \quad (6.4)$$

N_B is the number of pixels inside the VOP multiplied by $2^{(bits\_per\_pixel-8)}$. The (x,y) pair resulting in the lowest $SAD_{16}(x,y)$ is chosen as the 16×16 integer pixel motion vector, V0.

Likewise, the (x,y) pairs resulting in the lowest $SAD_8(x,y)$ are chosen to give the four 8×8 vectors V1, V2, V3 and V4. The 8×8-based SAD for the macroblock is

$$SAD_{K\times 8} = \sum_{1}^{K} SAD_8(x,y) \tag{6.5}$$

where $0 < K \leq 4$ is the number of 8×8 blocks that do not lie outside of the VOP shape. Instead of full search, the 8×8 search is centered around 16×16 vector, with a search window of ± 2 pixels.

If interlaced video is being encoded, four field motion vectors are calculated for every macroblock (in addition to existing 16×16 and 8×8 motion vectors described above) for each reference VOP. The field motion vectors correspond to the four combinations of current field (top or bottom) and reference field (top or bottom). The top field consists of even lines (0, 2, 4, ... , $H-2$), and the bottom field is composed of the odd lines (1, 3, 5, ldots, $H-1$, where H is the frame height). The full-pel field motion vector $(fx_{p,q}, fy_{p,q})$ is defined by the minimum sum of absolute differences $fSAD_{p,q}$ given by

$$\begin{aligned} fSADp,q = \min_{(x,y)\in S, y \text{ even}} & \sum_{j=0}^{7}\sum_{i=0}^{15} |C[i,2j+p] \\ & - OR[x_0+x+i, y_o+y+2j+q]| * (A[i,2j+p] \neq 0) \end{aligned} \tag{6.6}$$

where

(x_0, y_0)	Upper left corner coordinates of the current macroblock
p	Field in the current frame/VOP (0 for top; 1 for bottom)
q	Field in reference frame/VOP (0 for top; 1 for bottom)
$C[x,y]$	Current macroblock luminance samples
$A[x,y]$	Current macroblock alpha values
$OR[x,y]$	Reconstructed reference VOP luminance samples
S	Search region: $\{(x,y) : -2^{f\_code+3} \leq x,y < x^{f\_code+3}\}$

The field motion vectors are unrestricted in the sense that pixels referenced above might fall outside of the VOP bounding box but within the padded extension. If any pixel is required beyond the padded reference VOPs, then the candidate motion vector is eliminated from the search region, S.

The full-pixel SAD for an interlaced P-VOP macroblock is given by:

$$\begin{aligned} SAD_{inter} = min\{SAD_1 6(x,y), SAD_{k\times 8}, min(fSAD_{0,0}, fSAD_{0,1}) \\ + min(fSAD_{1,0}, fSAD_{1,1})\} \end{aligned} \tag{6.7}$$

INTRA/INTER Mode Decision

After the integer pixel motion estimation, the coder makes a decision on whether to use INTRA or INTER prediction in the coding. INTRA mode is chosen if:

$$A < (SAD_{inter} - 2 * N_B) \tag{6.8}$$

where

$$A = \sum_{i=1,j=1}^{16,16} |original - MB\_mean| * (!(Alpha_{original} == 0)) \tag{6.9}$$

and

$$SAD_{inter} = min(SAD_{16}(x, y), SAD_{k\times 8}) \tag{6.10}$$

$$MB_{mean} = (\sum_{i=1,j=1}^{N_c} original)/N_c \tag{6.11}$$

N_c is the number of pixels inside the VOP.

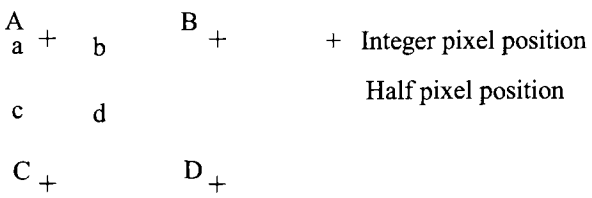
If INTRA mode is chosen, no further operations are necessary for the motion search. If INTER mode is chosen the motion search continues with half sample search around the V0 position, followed by quarter sample search if (quarter_sample $\neq$ 1).

Half Sample Search

Half sample search is performed using the previous reconstructed VOP on the luminance component of the macroblock, for 16×16 and 8×8 vectors as well as for 16×8 field motion vectors in the case of interlaced video. The search area is ± 1 half sample around the region pointed to by the motion vectors V0, V1, V2, V3, V4 or the field motion vector $(fx_{p,q}, fy_{p,q})$. The half sample values are calculated by bilinear interpolation horizontally and vertically as shown in Fig. 6.17 below:

The vector resulting in the best match during the half sample search is named MV. MV consists of horizontal and vertical components (MV_x, MV_y), both measured in half sample units.

For interlaced video, the half sample search used to refine the field motion vectors is conducted by vertically interpolating between lines of the same field. The field motion vectors are calculated in frame coordinates; that is the vertical coordinates of the integer samples differ by 2.



$$a = A$$
$$b = (A + B + 1 - rounding\_control)/2$$
$$c = (A + C + 1 - rounding\_control)/2$$
$$d = (A + B + C + D + 2 - rounding\_control)/4$$

Figure 6.17: Bilinear interpolation scheme. ©ISO/IEC 1998

Quarter Sample Search

For quanter sample search, an additional search step is performed for a search area of ± 1 quarter sample around the region pointed to by the best matched half sample motion vector MV. The quarter sample values are found by the same bilinear interpolation technique between surrounding half and integer samples respectively as shown in Fig. 6.18.

The vector resulting in the best match during the quarter sample search replaces the half sample vector MV. For quarter sample mode MV consists of horizontal and vertical components (MV_x, MV_y), both measured in quarter sample units.

Decision on 16×16 or 8×8 Prediction Mode

The decision to chose 16×16 or 8×8 prediction mode is based on the following criterion:

```
If SAD_{k×8}(x,y) < SAD_16(x,y) − (N_B/2 + 1)
   choose 8 × 8 prediction
else
   choose 16 × 16 prediction
```

Interlaced Video Prediction Mode Decision

In the case of interlaced video, the decision as to which prediction mdoe to use is by choosing the reference field that gives the smallest SAD (SAD_{top}

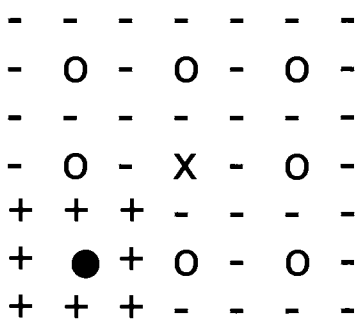


Figure 6.18: Location of best match integer ($\times$), half ($\circ$) and quarter ($-$) sample; ($\bullet$) marking the best match half sample, ($+$) the quarter sample search positions. ©ISO/IEC 1998

and SAD_{bottom}) from the field search. Therefore,

`if` $SAD_{16} \leq min(SAD_{16}, SAD_{K\times 8} + N_B/2 + 1, SAD_{top} + SAD_{bottom} + N_B/4 + 1)$
    16×16 `prediction`
`else if` $SAD_{K\times 8}+N_B/2+1 \leq min(SAD_{16}, SAD_{K\times 8}+N_B/2+1, SAD_{top}+SAD_{bottom} + N_B/4 + 1)$
    8×8 `prediction`
`else if` $SAD_{top}+SAD_{bottom}+N_B/4+1 \leq min(SAD_{16}, SAD_{K\times 8}+N_B/2+1, SAD_{top} + SAD_{bottom} + N_B/4 + 1)$
    `field based motion estimation`

Differential Coding of Motion Vectors

In inter coding, the motion vector is coded differentially by performing prediction based on the three neoghboring motion vectors as shown in Fig. 6.19. The differential coding is with reference to the reconstructed shape.

The prediction of the current motion vector is defined as:

$$\begin{aligned} P_x &= Median(MV1_x, MV2_x, MV3_x) \\ P_y &= Median(MV1_y, MV2_y, MV3_y) \end{aligned} \tag{6.12}$$

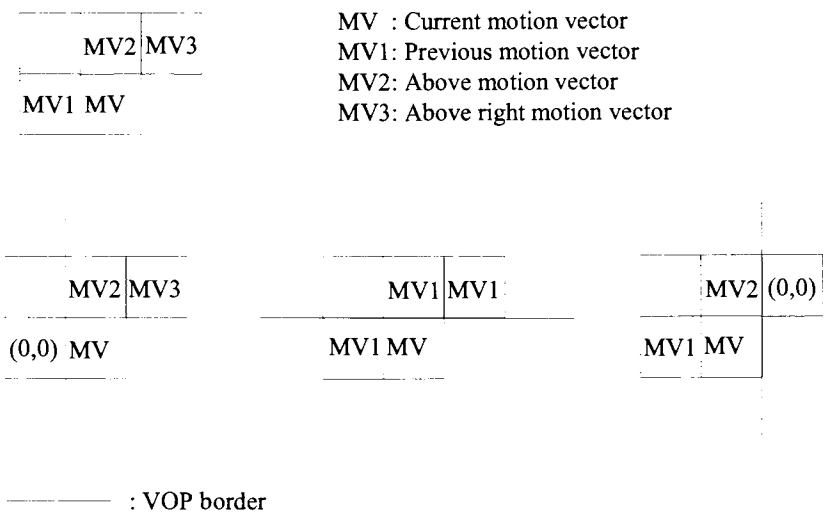


Figure 6.19: Motion vector prediction. ©ISO/IEC 1998

where $MV1$, $MV2$ and $MV3$ are the candidate predictors.

If error resilience is enabled, one dimensional prediction is used.

$$\begin{aligned} P_x &= MV1_x \\ P_y &= MV1_y \end{aligned} \tag{6.13}$$

The horizontal and vertical components of the differential motion vector are then given by:

$$\begin{aligned} MVD_x &= MV_x - P_x \\ MVD_y &= MV_y - P_y \end{aligned} \tag{6.14}$$

In the special case at the borders of the current VOP, the following decision rules are applied:

1. If only one candidate predictor is outside of the VOP, it is set to zero.
2. If only two candidate predictors are outside of the VOP, they are set to the third candidate predictor.
3. If all three candidate predictors are outside of the VOP, they are set to zero.

When interlaced video is encoded, if one or more of $MV1$, $MV2$ and $MV3$ refers to a field motion compensated macroblock, the value of MVi

is the average of the two field motion vectors. If a pixel offset finer than the motion vector resolution (half or quarter sample) is obtained by the average, it is replaced with a nearest pixel offset of the respective resolution.

If the current macroblock is a field motion compensated macroblock, then the same prediction motion vector (P_x, P_y) is used for both field motion vectors. Because the vertical component of a field motion vector has half the resolution of the horizontal component, the vertical differential motion vector encoded in the bitstream is

$$MVD_{yfield} = (MV_y - P_y)/2. \tag{6.15}$$

6.6.4.3 Unrestricted Motion Estimation/Compensation

Motion Estimation over VOP Boundaries

To improve the accuracy of motion estimation, the motion vector is allowed to point outside the decoded area of a reference VOP. For an arbitrarily shaped reference VOP, the decoded area refers to the area within the bounding box, padded as described in Section 6.6.4.1. The bounding box is extended multiples of 16×16 blocks in four directions (left, top, right and bottom) by 2^{f_code+3} pixels by repetitive padding. For the case of rectangular VOP, the rectangle is extended by the same amount in the same four directions. The target VOP remains the same except for extending to multiples of 16×16 blocks.

Modified block or polygon matching as described in Section 6.6.4.2 is applied to obtain the motion vectors.

Motion Compensation over VOP Boundaries

For motion compensation, when a sample referenced by a motion vector falls outside the decoded VOP area defined by the bounding box, an edge sample is used. This edge sample is the last full pixel position inside the decoded VOP area. Thus, the coordinates of the reference sample in the reference VOP are determined as follows:

$$\begin{aligned} x_{ref} &= \min[\max(x + dx, 0), x_{dim} - 1] \\ y_{ref} &= \min[\max(y + dy, 0), y_{dim} - 1] \end{aligned} \tag{6.16}$$

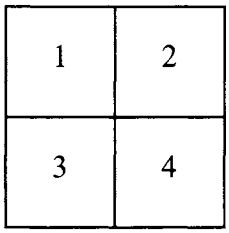


Figure 6.20: Numbering of luminance blocks in a macroblock. ©ISO/IEC 1998

where	(x, y)	are the coordinates of a sample in the current VOP,
	(x_{ref}, y_{ref})	are the coordinates of a sample in the reference VOP,
	(dx, dy)	is the motion vector, and
	(x_{dim}, y_{dim})	are the dimensions of the reference VOP.

6.6.4.4 Advanced Prediction Mode

Formation of the Motion Vectors

One, two or four motion vectors per macroblock may be used depending on the $MCBPC$ codeword and the *field_prediction* flag. If one motion vector MV is transmitted, all four block vectors take the same value as MV. When two field motion vectors are transmitted, each of the four block prediction motion vectors has the value equal to the average of the field motion vectors. If four motion vectors are transmitted for the current macroblock, MVD and MVD_{2-4} indicate the vector differences of luminance blocks 1 to 4 according to the block numbering in a macroblock as shown in Fig. 6.20 below. The motion vectors are obtained by adding predictors to MVD and MVD_{2-4} except that the candidate predictors $MV1$, $MV2$ and $MV3$ are redefined as illustrated in Fig. 6.21. If only one vector per macroblock is present, $MV1$, $MV2$ and $MV3$ are defined as for the 8×8 block numbered 1 in Fig. 6.20.

Motion vector MVD_{CHR} for both chrominance blocks is derived by calculating the sum of the K luminance vectors, that corresponds to K 8×8 blocks that do not lie outside the VOP shape, and dividing this sum by 2 in the case of half sample mode.

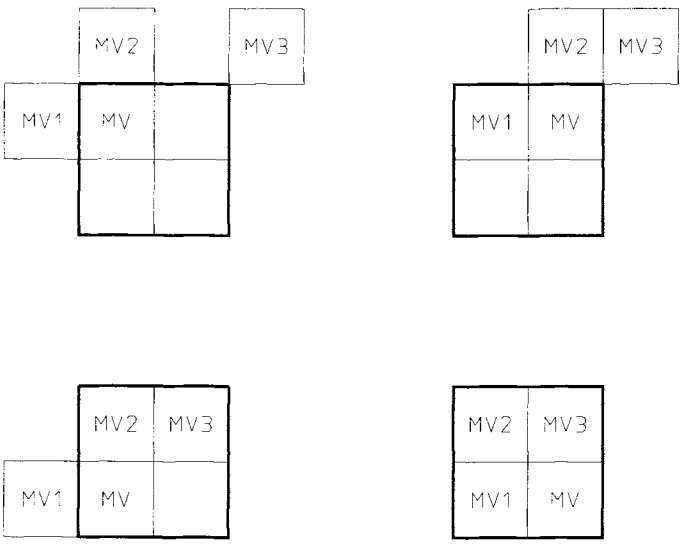


Figure 6.21: Redefinition of the candidate predictors $MV1$, $MV2$ and $MV3$ for each of the luminance blocks in a macroblock. ©ISO/IEC 1998

Overlapped Motion Compensation for Luminance Block

In overlapped motion compensation for luminance block, three motion vectors in the overlapped region (as illustrated in Fig. 6.22) are used:

- motion vector of the current block;
- motion vector of the block at the above or below the current block;
- motion vector of the block at the left or right of the current block.

For each 8×8 block in the macroblock, the motion vectors of the two nearest "remote" (left, above, right and below) blocks are used together with that of the current block. Let the motion compensated pixels of the blocks from the reference VOP be $q(i,j)$ for the current block, $r(i,j)$ for the above or bottom block, $s(i,j)$ for the left and right block. Thus

$$q(i,j) = p(i + MV_x^0, j + MV_y^0) \tag{6.17}$$

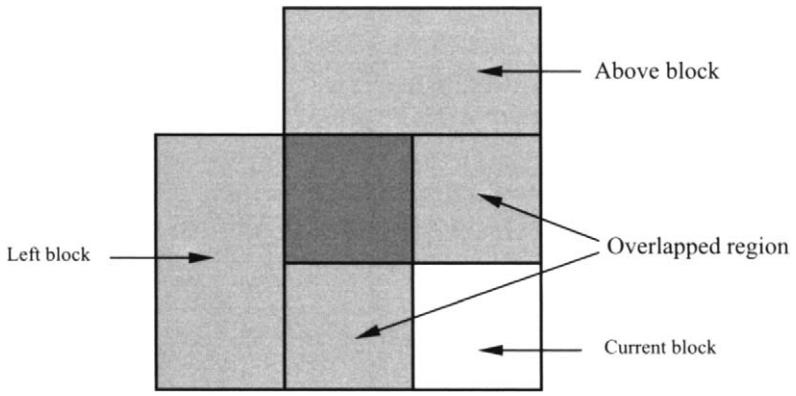


Figure 6.22: Overlapping blocks for motion compensation. ©ISO/IEC 1998

$$r(i,j) = p(i + MV_x^1, j + MV_y^1) \tag{6.18}$$

$$s(i,j) = p(i + MV_x^2, j + MV_y^2) \tag{6.19}$$

where

$p(i,j)$	is the pixel in the reference VOP;
(MV_x^0, MV_y^0)	is the motion vector of the current block;
(MV_x^1, MV_y^1)	is the motion vector of the block above or below the current block;
(MV_x^2, MV_y^2)	is the motion vector of the block at the left or right of the current block.

If MV_i^j points to a sub-sample position, the respective interpolation technique according to Section 6.6.4.2 is used.

The pixels in the prediction block $\hat{p}(i,j)$ are then given by:

$$\hat{p}(i,j) = [q(i,j) \times H_0(i,j) + r(i,j) \times H_1(i,j) + s(i,j) \times H_2(i,j) + 4]//4 \tag{6.20}$$

where $H_0(i,j)$, $H_1(i,j)$ and $H_2(i,j)$ are the weighting matrices as defined in Fig. 6.23.

Interlaced Motion Compensation

When field-based motion compensation is specified, two field motion vectors and corresponding reference fields are used to generate the prediction from

4	5	5	5	5	5	5	4
5	5	5	5	5	5	5	5
5	5	6	6	6	6	5	5
5	5	6	6	6	6	5	5
5	5	6	6	6	6	5	5
5	5	6	6	6	6	5	5
5	5	5	5	5	5	5	5
4	5	5	5	5	5	5	4

(a)

2	2	2	2	2	2	2	2
1	1	2	2	2	2	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	2	2	2	2	1	1
2	2	2	2	2	2	2	2

(b)

2	1	1	1	1	1	1	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	1	1	1	1	1	1	2

(c)

Figure 6.23: Weighting values for prediction with motion vector of current luminance block: (a)$H_0(i,j)$, (b) $H_1(i,j)$, (c) $H_2(i,j)$.

each reference VOP. For luminance motion compensation, the even lines of the macroblock are predicted from the top field motion vector using the reference field specified. The motion vector is specified in frame coordinates, i.e., full sample vertical displacements correspond to even integral values of vertical motion vector coordinate, a half sample vertical displacement is denoted by odd integral values and a quarter sample displacement by 0.5 values. The prediction of the odd luminance lines of the macroblock follows the same procedure above.

Chrominance motion compensation is perfomred field-wise. The even chrominance lines are predicted from the top field motion vector and the corresponding reference field, and the odd chrominance lines from the bottom field motion vector and the corresponding reference field. A chrominance motion vector is derived from the luminance motion vector by dividing each component by 2 followed by rounding.

6.6.5 Texture Coding

Basically the coding of the intra VOPs and the prediction error after motion compensation uses the 8×8 DCT coding scheme. In case of arbitrarily shaped VOP, the treatment of the macroblocks lying inside and on the VOP boundary is different. Those that lie completely inside the VOP boundary are coded using the H.263 coder. Whereas those intra macroblocks lying on the boundary are first padded as described in Section 6.6.5.1. For inter blocks, the region outside the VOP within the blocks are padded with zero. Padding is done for each of the luminance and chrominance blocks by using the original alpha values. The blocks are then coded same as the interior blocks. Transparent blocks are skipped and not coded. For blocks that lie outside the original shape, the intra blocks are padded with the value 128 for both the luminance and chrominance, whilst the inter blocks with the values of 0 for luminance and 128 for chrominance. Other blocks within the bounding box of a VOP are not coded at all.

6.6.5.1 Low Pass Extrapolation (LPE) Padding Technique

Before performing the DCT, for each intra block that has at least one transparent and one opaque pixel in its associated alpha information, the following block padding technique, referred to as *low-pass extrapolation* (LPE) padding, is applied to. The padding is performed in three steps.

1. Calculate the arithmetic mean value m of all block pixels $f(i, j)$ situated

within the object region R

$$m = \frac{1}{N} \sum_{(i,j) \in R} f(i,j) \tag{6.21}$$

where N is the number of pixels situated within the object region R. Division by N is done by rounding to the nearest integer.

2. Assign m to each block pixel situated outside of the object region R, i.e.

$$f(i,j) = m \qquad \text{for all}(i,j) \notin R. \tag{6.22}$$

3. Apply the following filtering operation to each block pixel $f(i,j)$ outside of the object region R, starting from the top left corner of the block and proceeding row by row to the bottom right pixel

$$f(i,j) = [f(i,j-1) + f(i-1,j) + f(i,j+1) + f(i+1,j)]/4 \tag{6.23}$$

Division is done by rounding to the nearest integer. If one or more of the four pixels used for filtering are outside the block, the corresponding pixels are not included into the filtering operation and the divisor 4 is reduced accordingly.

After this padding operation the resulting block is ready for DCT coding.

6.6.5.2 Adaptive Frame/Field DCT

MPEG-4 is capable of coding in frame or field DCT mode. The decision is based on the comparison of frame energy and field energy. That is, field DCT is used when

$$\begin{aligned} &\sum_{i=0}^{6} \sum_{i,j}^{1} 5(p_{2i,j} - p2i + 1, j)^2 + (p_{2i+1,j} - p2i + 2, j)^2 \\ &\quad > \sum_{i=0}^{6} \sum_{i,j}^{1} 5(p_{2i,j} - p2i + 2, j)^2 + (p_{2i+1,j} - p2i + 3, j)^2 \end{aligned} \tag{6.24}$$

where $p(i,j)$ is the spatial pixel value or prediction error before DCT.

When field DCT is used, the field macroblock is formed from the frame macroblock as shown in Fig. 6.24 below. The resulting macroblocks are then transformed, quantized and VLC encoded normally.

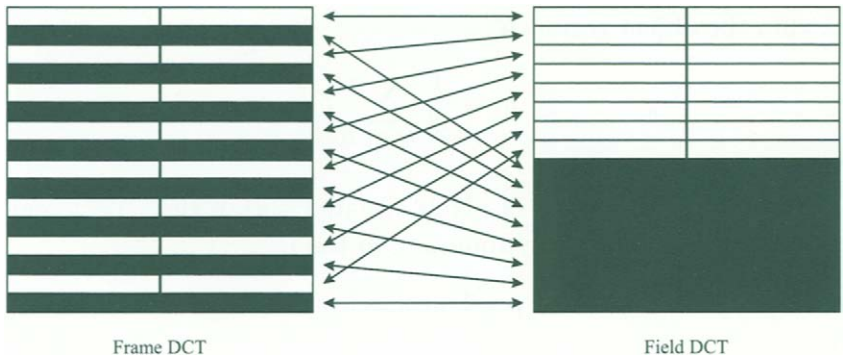


Figure 6.24: Formation of field macroblock from frame macroblock. ©ISO/IEC 1998

6.6.5.3 DCT

The $N \times N$ two dimensional (2-D) Discrete Cosine Transform (DCT) is defined as:

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos(\frac{(2x+1)u\pi}{2N}) cos(\frac{(2y+1)v\pi}{2N}) \tag{6.25}$$

with $x, y, u, v = 0, 1, 2, \ldots, N-1$, and

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u, v = 0 \\ 1 & \text{otherwise} \end{cases} \tag{6.26}$$

x, y are the spatial coordinates in the sample domain and u, v are the coordinates in the transform domain. The definition of the DCT is purely informative.

The $N \times N$ 2-D real-number inverse DCT (IDCT) is defined as:

$$F(x,y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u,v) cos(\frac{(2x+1)u\pi}{2N}) cos(\frac{(2y+1)v\pi}{2N}) \tag{6.27}$$

The integer IDCT is defined as:

$$f'(x,y) = round[f(x,y)] \tag{6.28}$$

The saturated integer IDCT is then defined as:

$$f''(x,y) = saturate[f'(x,y)] \tag{6.29}$$

where

$$C(u), C(v) = \begin{cases} -2^n & x < -2^n \\ 2^n - 1 & x > 2^n - 1 \\ x & -2^n \leq x \leq 2^n - 1 \end{cases} \tag{6.30}$$

and n is the number of bits per pixel.

The IDCT function $f[y][x]$ used in the decoding process may be any of several approximations of the saturated integer IDCT $f''(x,y)$, provided that it meets all of the following requirements:

- The IDCT function $f[y][x]$ used in the decoding process shall have values always in the range [-256, 255].

- The IDCT function $f[y][x]$ used in the decoding process shall conform to the IEEE Standard Specification for the implementation of 8×8 Inverse Discrete Cosine Transform, Standard 1180-1990, December 6, 1990.

- This clause applies only when input blocks of DCT coefficients cause all the 64 values output of the integer IDCT $f'(x,y)$ to be in the range [-384, 383]. When $f'(x,y) > 256$, $f[y][x]$ shall be equal to 255 and when $f'(x,y) < 257$, $f[y][x]$ shall be equal to -256. For all values of $f'(x,y)$ in the range [-257, 256] the absolute difference between $f[y][x]$ and $f''(x,y)$ shall not be larger than 2.

- Let F be the set of 4096 blocks $B_i[y][x]$, $i = 0, \ldots, 4095$ defined as follows:

$$b_i[y][x] = \begin{cases} i - 2048 & y, x = 0 \\ 0 & x, y \neq 0 \end{cases} \tag{6.31}$$

  For each block $B_i[y][x]$ that belongs to set F, an IDCT that conforms to this specification shall output a block $f[y][x]$ such that $f[y][x] - f''(x,y) = 0$ for all x and y.

6.6.5.4 SA-DCT & ΔDC-SA-DCT

When encoding a VOP of arbitrary shape, for the blocks which are completely within the shape, i.e. containing all opaque pixels, standard 8×8 DCT is applied. For those on the shape boundary, it is more efficient to employ DCT of arbitrary block size, known as *shape adaptive DCT* (SA-DCT) for inter-coded blocks. For intra-coded blocks, an extended version, ΔDC-SA-DCT is used.

Unlike the standard 8×8 DCT, SA-DCT and ΔDC-SA-DCT require the shape information provided by the binary alpha block. Only the opaque pixels within the shape boundary are transformed and coded thereby saving transmitted bit rate.

SA-DCT for Inter-coded Macroblocks

The SA-DCT is based on the odd or even orthonormal DCT basis functions. The procedure to calculate the SA-DCT of an arbitrary segment in a 8×8 block is illustrated in Fig. 6.25. First the segment is shifted vertically column by column to the upper edge of the block as in Fig. 6.25 (B). The length of each column N is then calculated. Depending on the length of the column, a one-dimensional N-DCT is performed on the pixels x_j of each of the columns to obtain the DCT coefficients X_j according to the following formula:

$$X_j = \sqrt{\frac{2}{N}} DCT_N x_j \tag{6.32}$$

where

$$DCT_N(p,k) = c_0 cos[p(k+0.5)\frac{\pi}{N}] \tag{6.33}$$

and

$$c_0 = \begin{cases} \sqrt{\frac{1}{2}} & p = 0; \\ 1 & \text{otherwise;} \end{cases} \quad \text{for } 0 \leq p,k \leq N-1 \tag{6.34}$$

The DCT coefficients are then shifted horizontally row by row to the left edge of the block as shown in Fig. 6.25 (E). The length of each row is found and a one-dimensional N-DCT is applied to each of the rows using the above formulae. This completes the computation of SA-DCT for an arbitrary segment in a 8×8 pixel block. Note that the number of coefficients is exactly equal to the number of pixels in the image segment. Also, the coefficients are located in a similar manner as in a standard 8×8 DCT block, i.e., the DC coefficient is located at the upper left corner with the

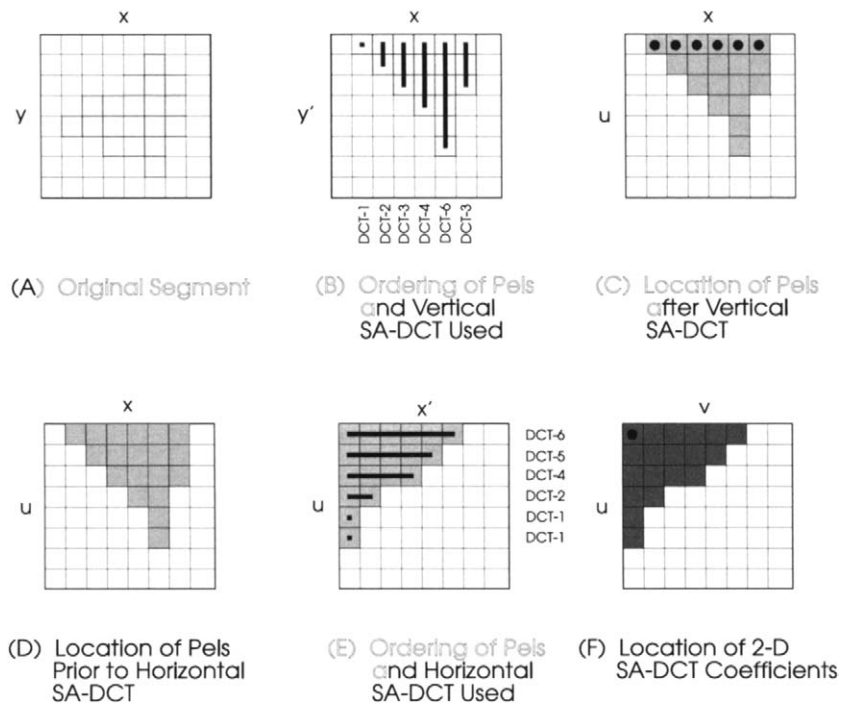


Figure 6.25: The procedure for performing a SA-DCT forward transform on a segment of arbitrary shape in a 8×8 block. ©ISO/IEC 1998

AC coefficients surrounding it, whose frequency content depends on their distance away from the DC cofficient.

Using the shape information obtained from the decoded BAB, inverse SA-DCT can be performed on the rows and columns of the transformed image segment according the following formula:

$$x_j = \sqrt{\frac{2}{N}} DCT_N^T X_j \tag{6.35}$$

where DCT_N^T is the transpose of DCT_N. The recovered pixel data together with the shape information are used to reconstruct the original image segment of arbitrary shape.

ΔDC-SA-DCT for Intra-Coded Macroblocks

For intra-coded macroblocks, ΔDC-SA-DCT, an extension of the SA-DCT is used to transform the arbitrarily shaped image segment within the blocks. Forward ΔDC-SA-DCT consists of the following steps:

1. The mean η of the pixels $x_{i,j}$ within the segment is first computed.
2. The mean is then subtracted from the pixels to generate zero-mean pixels $\hat{x}_{i,j}$ for the segment.
3. SA-DCT is applied to the zero-mean pixel data to yield the coefficients $X_{i,j}$.
4. The mean value is transmitted as the DC value of the SA-DCT coefficient matrix. For this the DC coefficient $X_{0,0}$ is redefined as

$$\hat{X}_{0,0} = 8.0\eta \tag{6.36}$$

$X_{0,0}$ which has been overwritten by $\hat{X}_{0,0}$ is called the ΔDC coefficient. Note that $X_{0,0}$ can be recomputed as a correction term during inverse ΔDC-SA-DCT in the decoder.

The inverse ΔDC-SA-DCT process follows the steps below:

1. After decoding the DC and AC coefficients, the mean value is extracted from the DC coefficient, i.e.,

$$\eta = \frac{\hat{X}_{0,0}}{8.0} \tag{6.37}$$

2. The DC coefficient is set to zero and together with the decoded AC coefficients form the coefficient matrix on which the inverse SA-DCT is performed.
3. Because of the zero-setting of the ΔDC coefficient, the sum of the inverse SA-DCT pixel values $\hat{x}_{i,j}$ is no longer zero. To correct this ΔDC error, a correction term is computed:

$$corr = \frac{sum}{\sum_{j=0}^{7} \sqrt{N_j}} \tag{6.38}$$

where

$$sum = \sum_{i=0}^{7} \sum_{j=0}^{7} a_{i,j} \hat{x}_{i,,j} \tag{6.39}$$

and

$$a_{i,j} = \begin{cases} 0 & \text{if } BAB(i,j) = 0 \\ 1 & \text{otherwise} \end{cases} \tag{6.40}$$

4. Finally, the inverse ΔDC-SA-DCT pixel values are given by:

$$x_{i,j} = a_{i,j}[\hat{x}_{i,j} + \eta - \frac{corr}{\sqrt{N_j}}] \tag{6.41}$$

1	2	6	7	15	16		
3	5	8	14	17			
4	9	13	18				
10	12						
11							
19							

Figure 6.26: Adaptive SA-DCT zig-zag scan for the active SA-DCT domain area of Fig. 6.25 (F). ©ISO/IEC 1998

Adaptive Scanning of the SA-DCT-Coefficients

Unlike the standard block-based 8×8 DCT, SA-DCT processes only a segment which forms only part of the block. Therefore, for SA-DCT we define an "active SA-DCT domain area", an example is the dark shaded area of Fig. 6.25 (F). For efficient coding of the SA-DCT coefficients using VLC, the scanning of the coefficients is modified as shown in Fig. 6.26 below for the example of Fig. 6.25 (F). Note that if the active SA-DCT domain area is rectangular, the SA-DCT degenerates into a standard 8×8 DCT and the scanning follows the standard zig-zag scanning defined in H.261, H.263, MPEG-1 and MPEG-2.

6.6.5.5 H.263 Quantization Method

The quantization parameter QP may take integer values from 1 to 31. The quantization stepsize is $2 \times QP$.

Quantization

For INTRA: $LEVEL = |COF|/(1 \times QP)$
For INTER: $LEVEL = (|COF| - QP/2)/(x \times QP)$

where COF is a transform coefficient to be quantized and $LEVEL$ is the absolute value of the quantized version of the transform coefficient. Clipping to [-127, 127] is performed for all coefficients except intra DC.

Dequantization

$$|COF'| = \begin{cases} 0 & \text{if } LEVEL = 0 \\ 2 \times QP \times LEVEL + QP & \text{if } LEVEL \neq 0, QP \text{ is odd} \\ 2 \times QP \times LEVEL + QP - 1 & \text{if } LEVEL \neq 0, QP \text{ is even} \end{cases}$$

where COF' is the reconstructed transform coefficient. The sign of COF is then added to obtain $COF' = Sign(COF) \times |COF'|$. Clipping to [-2048, 2047] is performed before IDCT.

The DC coefficient of an INTRA block is quantized as described below. 8 bits are used for the quantized DC coefficient.

Quantization: $LEVEL = COF//8$
Dequantization: $COF' = LEVEL \times 8$

6.6.5.6 MPEG Quantization Method

Quantization of Intra Macroblocks

Nonlinear Quantization of DC Coefficients

Note that his section is also valid for H.263 quantization.

The quantization strategy for intra DC coefficients is by scaling the coefficients with a non-linear function of the quantizer parameter Q_p. Separate DC Scalers for luminance (Type 1) and chrominance (Type 2) are used, with smaller value for chrominance than luminance. The characteristics of the DC Scalers as a function of Q_p is shown in Fig. 6.27.

The forward quantization is given by:

$$level = dc\_coef//dc\_scaler \tag{6.42}$$

whilst the reconstructed DC values are given by

$$dc\_rec = dc\_scaler * level \tag{6.43}$$

Quantization of AC Coefficients

The AC coefficients are first scaled by a intra quantizer matrix which by default is a flat matrix consists of 16 for all values. A user-downloaded non-flat quantizer matrix can alternatively be used.

The scaled AC coefficients are:

$$ac'[i][j] = (16 * ac[i][j])//w_1[i][j] \tag{6.44}$$

where $w_1[i][j]$ is the $[i][j]$th element of the default intra quantizer matrix. The resulting $ac'[i][j]$ is limited to the range of [-2048, 2047].

The quantized level is thus given by:

$$QAC[i][j] = \frac{ac'[i][j] + sign(ac'[i][j]) \times (3 \times Q_p//4)}{2 \times Q_p} \tag{6.45}$$

where $QAC[i][j]$ is limited to the range of [-127, 127].

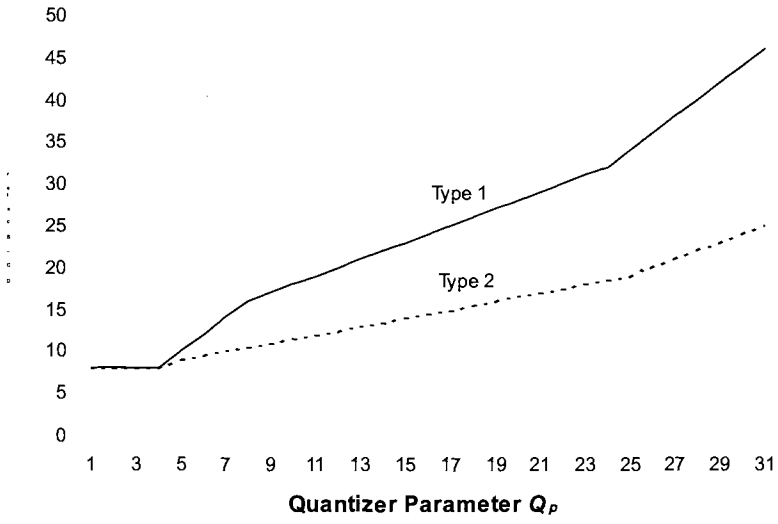


Figure 6.27: DC_Scaler characteristic as a function of quanitzer parameter Q_p. ©ISO/IEC 1998

Quantization of Non-Intra Macroblocks

Non-intra macooblocks in P- and B-VOPs are quantized with a uniform quantizer that has a dead-zone about zero. The default quantizer matrix is a flat matrix consists of 16 for all values. As in the case of quantization of intra AC coefficients, a user-downloaded non-flat quantizer matrix can alternatively be used.

The scaled AC coefficients are:

$$ac'[i][j] = (16 \times ac[i][j])//W_N[i][j] \tag{6.46}$$

where $W_N[i][j]$ is the $[i][j]$th element of the default non-intra quantizer matrix.

The quantized level is thus given by:

$$QAC[i][j] = \frac{ac'[i][j]}{2 \times Q_p} \tag{6.47}$$

where $QAC[i][j]$ is limited to the range of [-127, 127].

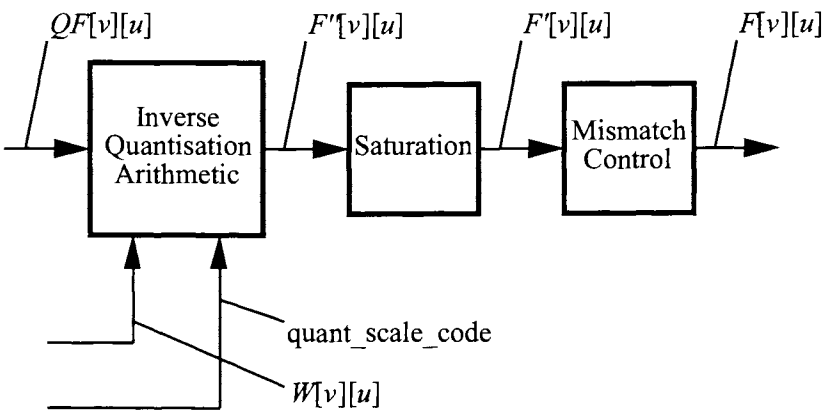


Figure 6.28: Inverse quantization process. ©ISO/IEC 1998

Inverse Quantization of Intra and Non-Intra Macroblocks

Fig. 6.28 illustrates the inverse quantization process. After the appropriate inverse quantisation arithmetic, the resulting coefficients $F''[v][u]$ are saturated to yield $F'[v][u]$ and then a mismatch control operation is performed to give the final reconstructed DCT coefficients $F[v][u]$.

Intra DC Coefficient

In intra-coded blocks, the reconstructed DC values $F''[0][0]$ are computed from the quantized values $QF[0][0]$ as follows:

$$F''[0][0] = dc\_scaler \times QF[0][0] \tag{6.48}$$

Other Coefficients

To reconstruct $F''[v][u]$ from $QF[v][u]$, the following equation is used:

$$F''[v][u] = \frac{2 \times QF[v][u] + k \times w[v][u] \times quantizer\_scale}{32} \tag{6.49}$$

In the the above expression, $k = 0, w[v][u] = w_1[v][u]$, for intra blocks; and $k = sign(QF[v][u]), w[v][u] = w_N[v][u]$, for non-intra blocks.

The coefficients resulting from the Inverse Quantisation Arithmetic are saturated to lie in the range [-2048, 2047]. Thus:

$$F''[v][u] = \begin{cases} 2047 & F''[v][u] > 2047 \\ F''[v][u] & -2048 \leq F''[v][u] \leq 2047 \\ -2048 & F''[v][u] < -2048 \end{cases} \tag{6.50}$$

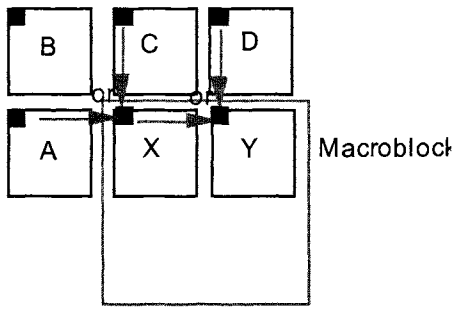


Figure 6.29: Previous neighboring blocks used in DC prediction. ©ISO/IEC 1998

Mismatch control is carried out to compensate for mismatch between DCT and IDCT. Note that only the last coefficient $F[7][7]$ is compensated. It is carried out according to the following procedure:

1. The sum of all coefficients is calculated:

$$sum = \sum_{v=0}^{v<8} \sum_{u=0}^{u<8} F'[v][u] \tag{6.51}$$

2. For all u and v except,$u = v = 7$, F[v][u] = F'[v][u].

3. If *sum* is even, a correction is made to $F[7][7]$, i.e.,

$$F[7][7] = \begin{cases} F'[7][7] - 1 & \text{if } F'[7][7] \text{ is odd} \\ F'[7][7] + 1 & \text{if } F'[7][7] \text{ is even} \end{cases} \tag{6.52}$$

6.6.5.7 Intra DC and AC Prediction for I-VOP and P-VOP

Adaptive DC Coefficient Prediction

The adaptive prediction of DC coefficient of the current block is based on the quantized DC (QDC) values of the immediately previous block on the same row and the block above it. Fig. 6.29 shows the neighboring blocks used in DC prediction.

DC coefficients are first divided by 8 to obtain the quantized values:

$$QDC = dc\_coef//8 \tag{6.53}$$

Let the predicted QDC value of the current block 'X' be $QPC_{X'}$, the prediction rule is as follows:

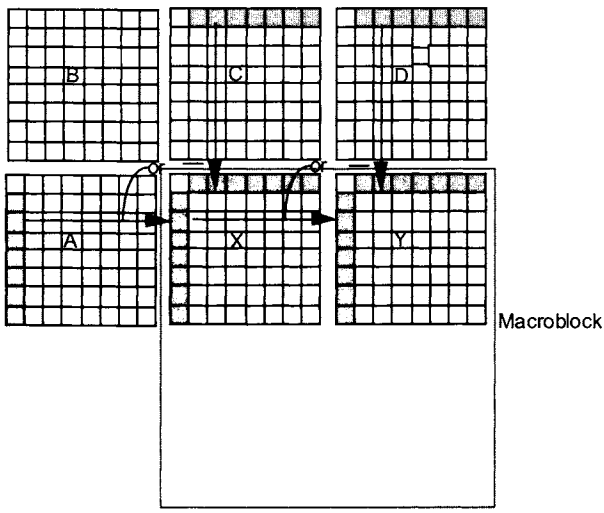


Figure 6.30: Previous neighboring blocks and coefficients used in AC prediction. ©ISO/IEC 1998

```
if |QDC_A − QDC_B| < |QDC_B − QDC_C|
    QDC_X' = QDC_C
else
    QDC_X' = QDC_A
```

The differential DC value is then obtained by subtracting the DC prediction, $QDC_{X'}$ from the QDC of block 'X', $QDC_{X'}$.

Adaptive AC Coefficient Prediction

Prediction of First Row or First Column of AC Coefficients

The prediction of the AC coefficients in the first row or the first column of the current block is based on the co-sited previous coded blocks. The direction of the DC coefficient prediction is used to determine the direction of the AC coefficient prediction. The AC coefficient prediction is illustrated in Fig. 6.30.

Q-step Scaling

To compensate for differences in the quantization of previous horizontally adjacent or vertically adjacent blocks used in AC prediction of the current

block, the prediction is scaled is scaled by the ratio of the current quantisation stepsize and the quantisation stepsize of the predictor block. Thus, if the AC prediction is from block 'A', the horizontal AC prediction of block 'X' is given by:

$$QAC_{i0X'} = \frac{QAC_{i0A} \times QP_A}{QP_X} \tag{6.54}$$

where QAC_{i0A} is the quantized horizontal AC coefficients of block 'A', QP_A and QP_B are QP values of blocks 'A' and 'X', respectively.

If the AC prediction is from block 'C', the vertical AC prediction of block 'X' is:

$$QAC_{0jX'} = \frac{QAC_{0jC} \times QP_C}{QP_X} \tag{6.55}$$

where QAC_{0jC} is the quantized vertical AC coefficients of block 'C', and QP_C is QP value of block 'C'.

If block 'A' or block 'C' are outside of the VOP, then the corresponding QP values are assumed to be equal to QP_X.

AC Prediction Enable/Disable Mode Decision

The decision to enable or disable the AC prediction mode is based on a criterion S calculated as follows for the case if the AC prediction is from block 'A':

$$S = \sum_{i=1}^{7} |QAC_{i0X}| - \sum_{i=1}^{7} |QAC_{i0X'}| \tag{6.56}$$

If it is from block 'C', the criterion S is given as follows:

$$S = \sum_{j=1}^{7} |QAC_{0jX}| - \sum_{j=1}^{7} |QAC_{0jX'}| \tag{6.57}$$

Next for for all blocks in the macroblock for which a common decision is to be made, a single $\sum S$ is calculated and the AC prediction is enabled/disabled according to the rules below:

```
if ∑S > 0
   enable AC prediction
else
   disable AC prediction
```

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

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

Figure 6.31: (a) Alternate-horizontal scan (b) Alternate-vertical (MPEG-2) scan. ©ISO/IEC 1998

Other simple rules apply to AC prediction are:

- If any of the blocks A, B or C are outside of the VOP boundary or do not belong to an intra coded macroblock, their QAC values are assumed to take a value of 0 and are used to compute prediction values.
- AC predictions are performed similarly for the luminance and each of the two chrominance components using the direction identified by the corresponding direction of DC prediction.
- The process for DC/AC prediction for alpha plane is similar to that of texture as described above.

6.6.5.8 VLC Encoding of Quantized Transform Coefficients

VLC Encoding of Intra Macroblocks

In addition to the zig-zag scan, two scans are employed depending on the DC prediction direction, as shown in Fig. 6.31.

For intra blocks, if the AC prediction is disabled, the zig-zag scan is selected for all blocks in a macroblock. If the prediction is from the horizontal adjacent block, alternate vertical scan (Fig. 6.31(a)) is used for the current block, otherwise alternate horizontal scan (Fig. 6.31(b)) is used.

For non-intra blocks, the transform coefficients are scanned according to the zig-zag scan.

A three-dimensional variable length code is used to code transform coefficients. An EVENT is a combination of three parameters:

LAST 0: There are more nonzero coefficients in the block.

1	2	6	7	15	16	28	29
3	5	8	14	17	27	30	43
4	9	13	18	26	31	42	44
10	12	19	25	32	41	45	54
11	20	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	62
36	37	49	50	58	59	63	64

Figure 6.32: Zig-zag scanning pattern. ©ISO/IEC 1998

1: This is the last nonzero coefficient in the block.
RUN Number of zero coefficients preceding the current nonzero coefficient.
LEVEL Magnitude of the coefficient.

The most commonly occurring combinations of (LAST, RUN, LEVEL) are coded with variable length codes given in [4]. The remaining combinations of (LAST, RUN, LEVEL) are coded with a 22 bit word consisting of:

ESCAPE 7 bit
LAST 1 bit (0: Not last coefficient, 1: Last nonzero coefficient)
RUN 6 bit
LEVEL 8 bit

The code words for these fixed length ESCAPE codes are described in [4].

For intra macroblock chroma AC Coefficients, the VLC used is the same as that used for intra AC luminance coefficients

VLC Encoding of Inter Macroblocks

For inter blocks, the 8×8 transform coefficients are scanned with the zig-zag scanning as illustrated in Fig. 6.32. The coding of the transform coefficients is using the same VLC codetable as in the coding of intra macroblocks above.

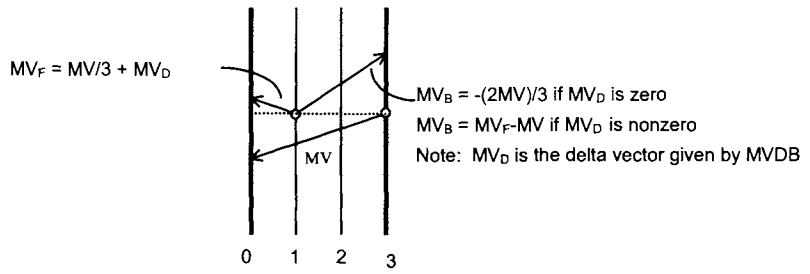


Figure 6.33: Progressive direct bidirectional prediction. ©ISO/IEC 1998

6.6.6 Prediction and Coding of B-VOPs

Prediction and coding of macroblocks in B-VOPs can use either the H.263 or the MPEG-1 approach. The main difference is in the amount of motion vector and quantization related overhead needed.

6.6.6.1 Direct Coding

Progressive Direct Coding

Progressive direct coding mode is used whenever the macroblock at the same location in the future anchor picture is coded as:

- a 16×16 (frame) macroblock,
- an intra macroblock or
- an 8×8 (advanced prediction) macroblock.

This coding mode uses direct bidirectional motion compensation derived by extending H.263 approach of employing P-picture macroblock motion vectors and scaling them to derive forward and backward motion vectors for macroblocks in B-picture. As per H.263, using B-frame syntax, only one delta motion vector is allowed per macroblock. Fig. 6.33 shows scaling of motion vectors.

The calculation of the forward and backward motion vectors follows the procedure in H.263. The only difference is that here we are dealing with VOPs instead of pictures, and instead of only a single B-picture between a pair of reference pictures, multiple B-VOPs are allowed between a pair of reference VOPs. As in H.263, the temporal referenceof the B-VOP relative to difference in the temporal reference of the pair of reference VOPs is used

to determine scale factors for computing motion vectors which are corrected by the delta vector MV_δ.

The forward and the backward motion vectors, MV_F and MV_B, given in half or quarter sample units are obtained as follows:

$$MV_F = \frac{TR_B \times MV}{TR_D} + MV_\delta \tag{6.58}$$

$$MV_B = \frac{(TR_B - TR_D) \times MV}{TR_D} \quad \text{if } MV_\delta = 0 \tag{6.59}$$

$$MV_B = MV_F - MV \quad \text{if } MV_\delta \neq 0 \tag{6.60}$$

where MV is the direct motion vector of a macroblock in P-VOP with respect to a reference VOP; TR_B is the difference in temporal reference of the B-VOP and the previous reference VOP; and TR_D is the difference in temporal reference of the temporally next reference VOP with temporally previous reference VOP, assuming B-VOPs or skipped VOPs in between.

The prediction blocks are generated using computed forward and backward motion vectors to obtain appropriate blocks from reference VOPs and averaging these blocks. Motion compensation is performed individually on 8×8 blocks to generate a macroblock irregardless of whether the direct prediction motion vectors are derived by scaling a single motion vector or four 8×8 block motion vectors. It should be noted that if if the next reference VOP is an I-VOP instead of a P-VOP, then the MV vectors are by default '0'.

The direct coding mode does not allow quantizer change and thus the quantizer value for previous coded macroblock is used.

Interlaced Direct Coding

The extension to interlaced video is used whenever the corresponding macroblock of the future anchor VOP is a field motion compensated macroblock.

In interlaced direct mode, the prediction macroblock is formed separately for the top and bottom fields. The four field motion vectors of a bi-directional field motion compensated macroblock are calculated from the motion vectors of the corresponding macroblock of the future anchor picture. Only one delta motion vector (used for both fields) is used for the field predicted macroblock.

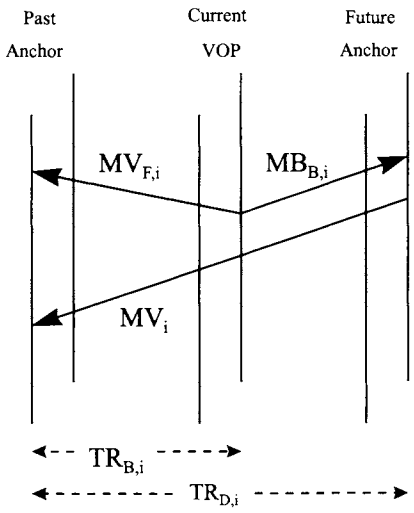


Figure 6.34: Interlaced direct bidirectional prediction. ©ISO/IEC 1998

The motion vectors for field i (top or bottom) are calculated by:

$$MV_{F,i} = \frac{TR_{B,i} \times MV_i}{TR_{D,i}} + MV_\delta \tag{6.61}$$

$$MV_{B,i} = \frac{(TR_{B,i} - TR_{D,i}) \times MV_i}{TR_{D,i}} \quad \text{if } MV_\delta = 0 \tag{6.62}$$

$$MV_{B,i} = MV_{F,i} - MV_i \quad \text{if } MV_\delta \neq 0 \tag{6.63}$$

where $MV_{F,i}$ is forward motion vector for field i whose reference field is the reference field of MV_i; $MV_{B,i}$ is backward motion vector for field i; MV_i is field motion vector for field i of the macroblock at the same location as the current macroblock in the future anchor VOP; $TR_{B,i}$ is temporal distance in fields between the past reference field for field i and field i of the current B-VOP; and $TR_{D,i}$ is temporal distance in fields between the past reference field and the future reference field for the current VOP's field i.

The calculation of $TR_{B,i}$ and TRD, i depends not only on the current field, reference field, and frame temporal references, but also on whether the current video is top field first or bottom field first, i.e.,

$$TR_{D,i} = 2(TR_{future} + TR_{past}) + \delta \tag{6.64}$$

$$TR_{B,i} = 2(TR_{current} + TR_{past}) + \delta \tag{6.65}$$

where TR_{future}, $TR_{current}$ and TR_{past} are the frame number of the future,

current and past frames in display order, and δ is 0 or ± 1 depending on whether the reference is top or bottom field.

6.6.6.2 Forward, Backward and Bidirectional Coding Modes

Forward coding mode uses forward motion compensation in the same manner as in MPEG-1/2 with the difference that a VOP is used for prediction instead of a picture. Only one motion vector in half or quarter sample units is employed for a 16×16 macroblock being coded. Chrominance vectors are derived by scaling of luminance vectors as in MPEG-1/2, but are restricted to half sample accuracy.

Backward coding mode is the same as above except backward motion compensation is used. *Bidirectional coding* mode is the same as above except *bidirectional* motion compensation is used. The above coding modes allow switching of quantizer from the one previously in use.

6.6.6.3 Mode Decisions

To select the coding mode for a macroblock in B-VOP, the motion compensated prediction macroblocks difference SAD (sum of absolute differences) is calcualted for each mode. The decision is according to the following rule:

```
if [SAD_direct − (0.5N_B + 1) ≤ min(SAD_forward, SAD_backward, SAD_bidirectional)]
   direct mode
else if [SAD_bidirectional ≤ min(SAD_forward, SAD_backward, SAD_bidirectional)]
   bidirectional mode
else if [SAD_backward ≤ min(SAD_forward, SA_backward, SAD_bidirectional)]
   backward mode
else
   forward mode
```

For interlaced B-VOPs, a macroblock can be coded using:

- direct coding;
- 16×16 motion compensated (including forward, backward and bidirectional modes); or
- field motion compensation (including forward, backward and bidirectional modes).

Table 6.2: Prediction motion vectors. ©ISO/IEC 1998

Function	PMV
Top field forward	0
Bottom field forward	1
Top field backward	2
Bottom field backward	3

The decision regarding coding mode of the macroblock is based on the minimum luminance SADs with respect to the decoded anchor pictures:

$$SAD_{direct} + b_1, SAD_{forward} + b_2, SAD_{forward,field} + b_3, SAD_{backward} + b_2,$$

$$SAD_{backward,field} + b_3, SAD_{average} + b_3, SAD_{average,field} + b_4$$

where the subscripts direct, forward, backward, average and field mean forward motion prediction, backward motion prediction, average (i.e., interpolated or bidirectional) motion prediction and field mode. The field SADs above are the sums of the top and bottom field SADs each with its own reference field and motion vector. The bias terms b_i's take on the values 0, or $\pm(N_B/2 + 1)$ depending on the motion prediction mode.

6.6.6.4 Motion Vector Coding

Only the motion vectors used for the selected prediction mode are coded. The motion vectors are coded differentially with the left neighboring vector used as predictor in the forward, backward and bidirectional modes. In the case where the current macroblock is located on the left edge of the VOP, or no vector is present in the left neighboring macroblock, the predictor is set to zero.

For interlaced B-VOP motion vector predictors, four prediction motion vectors (PMVs) are as given in Table 6.2. These PMVs are used for the different macroblock prediction modes as indicated in Table 6.3.

The PMVs used by a macroblock are set to the value of current macroblock motion vectors after being used. The prediction motion vectors are reset to zero at the beginning of each row of macroblocks. The predictors are not zeroed by skipped macroblocks or direct mode macroblocks.

For macroblocks coded in direct bidirectional mode no vector differences are transmitted. Instead, the forward and backward motion vectors are

Table 6.3: PMVs for different macroblock prediction modes. ©ISO/IEC 1998

Macroblock mode	PMVs used
Direct	none
Frame forward	0
Frame backward	2
Frame bidirectional	0,2
Field forward	0,1
Field backward	2,3
Field bidirectional	0,1,2,3

directly computed from the temporally consecutive P-vector as described in Section 6.6.6.1.

6.6.7 Generalized Scalable Coding

Scalable coding enables MPEG-4 to encode video at different resolutions and/or quality. A scalable encoder generates a bitstream that allows decoding appropriate subset of the bitstream to generate complete pictures of resolution and/or quality commensurate with the proportion of bitstream decoded and the capability and complexity of the decoder.

Two main types of scalability are allowed in MPEG-4, namely, spatial scalability and temporal scalability. As the names imply, spatial scalability offers scalability of the spatial resolution, and temporal scalability offers that of the temporal resolution. Each type of scalability consists of two layers: the base layer and the enhancement layer, with the base layer provides the basic resolution whilst the enhancement layer enhances the resolution of the base layer. In the case of spatial scalability, the enhancement layer increases the spatial resolution of the video, while in temporal scalability, the enhancement layer provides a higher temporal resolution for the video.

Traditionally, these scalabilities are applied to rectangular frames of video, but nowadays, many applications necessitate not only traditional frame based scalabilities but also scalabilities of VOPs of arbitrary shapes. MPEG-4 supports rectangular VOPs (frames) in both the spatial and temporal scalabilities, but only temporal scalability is capable to handle arbitrarily shaped VOPs at this moment.

A generalized scalable codec structure is shown in Fig. 6.35. The input to the Scalability PreProcessor are VOPs of arbitrary shape (rectangular

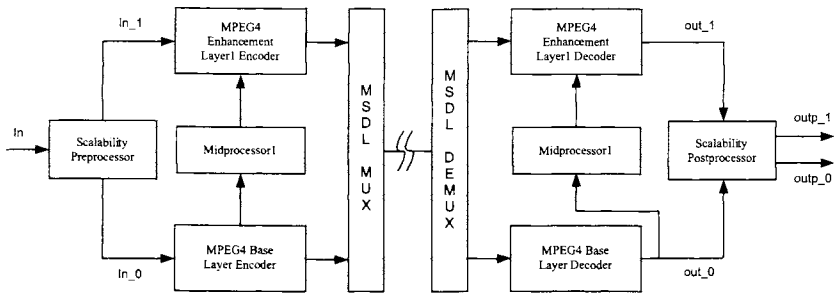


Figure 6.35: A generalized scalable codec structure. ©ISO/IEC 1998

shape is a special case). If spatial scalability is to be carried out, spatial downsampling is performed on the input VOPs to generate the base layers (in_0) as input to the Base Layer Encoder which performs a non-scalable encoding. The encoder could be one of the standard video encoders, such as H.263, H.261, MPEG-1 or MPEG-2. The reconstructed VOPs are then upsampled by the Midprocessor and used as prediction for the original VOPs (in_1) in the Enhancement Layer Encoder. The output bitstreams of the Enhancement Layer Encoder and the Base Layer Encoder are then multiplexed by the MSDL Multiplexer for transmission. The MSDL Demultiplexer, Base Layer Decoder, Enhancement Layer Decoder, Midprocessor and Scalability Postprocessor in the receiver peroforms the reverse operations to retrieve the resulting outputs for the Base Layer (outp_0) and the Enhancement Layer (out_0).

In the case of temporal scalability, the Scalability PreProcessor demultiplexes the VOP into two streams with different temporal resolutions, i.e., in_0 and in_1 as inputs to the Base Layer Encoder and the Enhancement Encoder, respectively. The MidProcessor does not perform any subsampling but merely allows the decoded base-layer VOPs to pass through and used for temporal prediction in encoding the enhancement layer. The output bitstreams of the Enhancement Layer Encoder and the Base Layer Encoder are then multiplexed for transmission. The operation of the Base Layer Decoder and Enhancement Layer Decoder are the reverse of their corresponding encoders. The PostProcessor does not carry out any conversion but temporally multiplexes the base and enhancement layers to produce higher temporal resolution enhancement layer VOPs.

6.6.7.1 Spatial Scalability Encoding

In spatial scalability, the base layer is encoded at a smaller spatial resolution than the enhancement layer. For example, if the base layer is encoded at QCIF resolution, the enhancement layer will be encoded at CIF resolution. Therefore, if QCIF resolution is required, only base layer will be decoded. However, if CIF resolution is required, both base and enhancement layers need to be decoded.

The down sampling process is performed at the Scalability PreProcessor. The down sampling process for the factor of 2 is only described here, however down sampling for any factor is allowed. The encoding process of the base layer is the same as non-scalable encoding process, such H.263, MPEG-1 or MPEG-2.

Upsampling Process

For spatial prediction of the enhancement layer, the reconstructed base-layer VOP is upsampled in the MidProcessor. The upsampling is performed in two steps, viz vertical upsampling and horizontal upsampling as described below.

Vertical Upsampling

The vertical upsampled image $vert\_pict[y][x]$ is obtained from the base-layer image $d_{lower}[y][x]$ using linear interpolation according to the following formula:

$$vert\_pic[y_h][x] = (16 - phase) \times d_{lower}[y_1][x] + phase \times d_{lower}[y_2][x] \tag{6.66}$$

where

$$\begin{aligned} y_h &= \text{output sample coordinate in vert\_pic} \\ y_1 &= y_h \times vertical\_sampling\_factor\_m / vertical\_sampling\_factor\_n \\ y_2 &= \begin{cases} y_1 + 1 & \text{if } y_1 < video\_object\_layer\_height - 1 \\ y_1 & otherwise \end{cases} \\ phase &= [16 * ((y_h * vertical\_sampling\_factor\_m) / vertical\_sampling\_factor\_n)] \end{aligned}$$

where $video\_layer\_height$ is the height of the reference VOL.

Samples which lie outside the lower layer reconstructed frame which are required for upsampling are obtained by border extension of the lower layer reconstructed frame.

NOTE: The calculation of phase assumes that the sample position in the enhancement layer at is spatially coincident with the first sample position of the lower layer. It is recognized that this is an approximation for the chrominance component if the chroma_format = 4:2:0.

Horizontal Upsampling

Horizontal upsampling is performed using a similar procedure as the vertical upsampling. The horizontal upsampled image $hor\_pict[y][x]$ is obtained from the vertically upsampled image $ver\_pict[y][x]$ using linear interpolation according to the following formula:

$$hor\_pic[y_h][x] = (16 - phase) \times vert\_pic[y][x_1] + phase \times vert\_pic[y][x_2]//256 \tag{6.67}$$

where

$$\begin{aligned} x_h &= \text{output sample coordinate in } hor\_pic \\ x_1 &= x_h \times horizontal\_sampling\_factor\_m / horizontal\_sampling\_factor\_n \\ x_2 &= \begin{cases} x_1 + 1 & \text{if } y_1 < video\_object\_layer\_width - 1 \\ x_1 & otherwise \end{cases} \\ phase &= [16 * ((x_h * horizontal\_sampling\_factor\_m) \\ &\qquad / horizontal\_sampling\_factor\_n)] \end{aligned}$$

where $video\_layer\_width$ is the width of the reference VOL.

Samples which lie outside the lower layer reconstructed frame which are required for upsampling are obtained by border extension of the lower layer reconstructed frame.

Encoding of Enhancement Layer

The VOP in the enhancement layer is encoded either as P-VOP or B-VOP. The relationship between VOP in the base layer and that of the enhancement layer is illustrated in Fig. 6.36. The VOP which is temporally coincident with the base-layer I-VOP is encoded as P-VOP. The VOP which is temporally coincident with the base-layer P-VOP is encoded as B-VOP. In spatial scalability, a decoded unsampled VOP in the base layer is used as a reference for prediction of the enhancement layer. The temporally coincident VOP in the reference layer (base layer) must be coded before the encoding of the VOP in the enhancement layer.

In the case of P-VOP, the INTRA/INTER mode decision is the same as that in the coding of the base layer. For Inter mode, the motion vector is always set to 0 and is not coded.

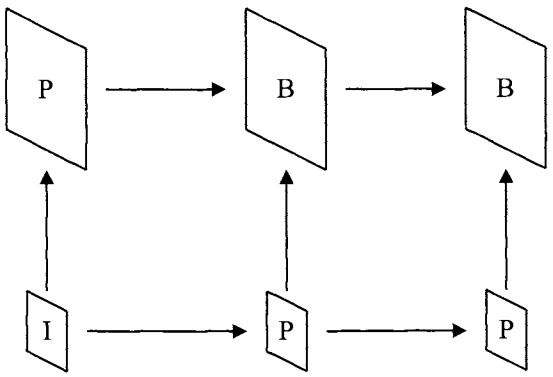


Figure 6.36: Prediction references for enhancement layer. ©ISO/IEC 1998

In the case of B-VOP, the backward prediction reference is set to P-VOP which is the temporally coincident VOP in the base layer, and the forward prediction reference is set to P-VOP or B-VOP which is the most recent decoded VOP of the enhancement layer. A macroblock in B-VOP in forward, backward or bidirectional modes and never in direct mode. The decision is which mode is best is to compare the SAD of the three modes and select the minimum as follows:

```
if [SAD_forward ≤ min(SAD_forward, SAD_backward, SAD_bidirectional)]
  direct mode
else if [SAD_bidirectional ≤ min(SAD_forward, SAD_backward, SAD_bidirectional)]
  bidirectional mode
else
  backward mode
```

6.6.7.2 Temporal Scalability Encoding

Temporal scalability can be applied to arbitrarily shaped VOPs, in addition to the traditional rectangular VOPs. In Object-based Temporal Scalability (OTS), the frame rate of the selected object in the enhancement layer is increased to provide a smoother motion than the same object in the base layer.

Type I

There are two types of temporal scalability. Fig. 6.37 shows Type 1 temporal scalability where VOL0 (Video Object Layer 0) is the entire frame containing the object and the backgroung, while VOL1 represents the object of VOL0. VOL0 is coded with a lower frame rate than VOL1. Prediction of VOL1 can be by (1) forward prediction with VOL0 as reference to form P-VOPs in the enhancement layer as shown in Fig. 6.37 (a), or (2) bidirectional prediction with VOL0 as reference to form B-VOPs in the enhancement layer as shown in Fig. 6.37 (b). In both cases, two additional shape data, a forward shape and a backward shape, are encoded to perform the background composition.

Type II

The other type of temporal scalability is Type II where the VO0 (Video Object 0) contains only the background and has no scalability layer. VO1 contains a particular object and consists of two scalability layers, VOL0 and VOL1 as shown in Fig. 6.38. VOL0 is considered the base layer and is coded at a lower frame rate than VOL1, the enhancement layer.

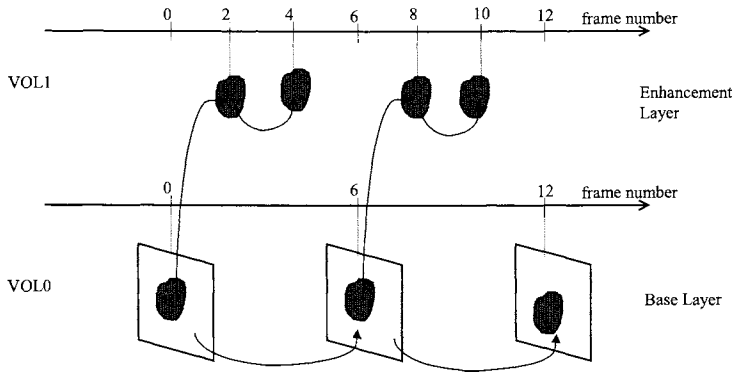
Enhancement Types

There are 2 types of enhancements for scalability, described by the enhancement_type flag. The base layer VOL0 for both enhancement types is coded with a lower a spatial of temporal resolution. At the enhancement layer, enhancement_type flag distinguished how it is to be enhanced. When this flag is 1, the enhancement layer increases the picture quality of a partial region of the base layer. For example, in Fig. 6.39, the temporal resolution or the spatial resolution of VOL1 (i.e., the car) is enhanced. When this flag is 0, the enhancement layer increases the picture quality of the entire region of the base layer. For example, in Fig. 6.39, the temporal or spatial resolution of VOL1 is enhanced. The region represented by VOL1 depends on the definition of VOL0.

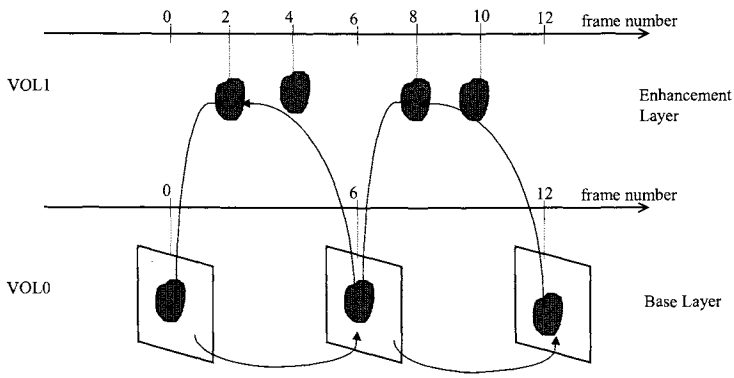
NOTE: All scalability modes can be applied in the case of extension to 8-bit video. It is not necessary that enhancement and base layers are specified as having the same number of bits per pixel.

6.6.8 Sprite Coding

Sprite is a still image present in all scenes in a video segment. A classic example of a sprite is the background of a video sequence generated by a panning camera. The sprite generated by the panning sequence when



(a) Prediction of enhancement layer to form P-VOPs.



(b) Prediction of enhancement layer to form B-VOPs.

Figure 6.37: Type I temporal scalability. ©ISO/IEC 1998

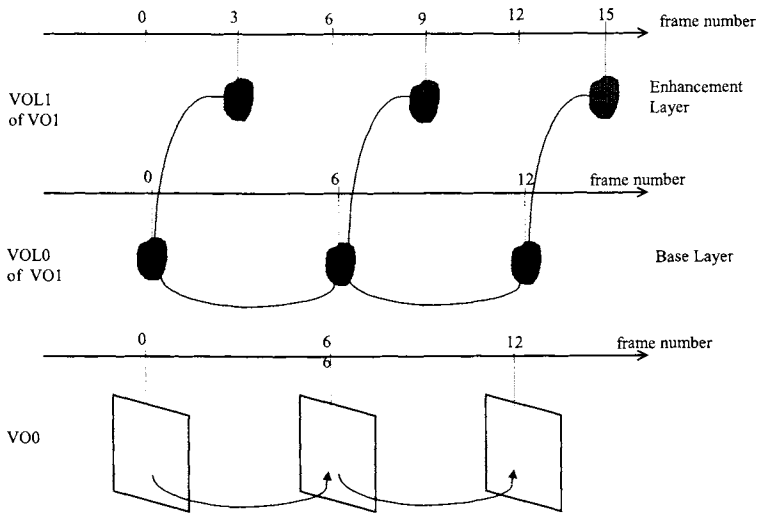


Figure 6.38: Type II temporal scalability. ©ISO/IEC 1998

composed can be transmitted as a large still image separately from the foreground object. This assumes the foreground objects can be segmented from the background and the sprite image can be extracted from the sequence prior encoding. In this way, the transmitted bit rate is reduced enormously as the sprite needs only to be transmitted once as the first frame of the sequence.

In the receiver, the background can be reconstructed based on the sprite using the global motion parameters describing the camera motion transmitted in subsequent frames. The foreground objects are transmitted separately as arbitrary-shaped video objects. Fig. 6.40 shows an exmaple of sprite coding of video sequence.

In sprite-based coding, two types of sprites are used, namely, (1) off-line static sprites, and (2) on-line dynamic sprites. The following describes them in more details.

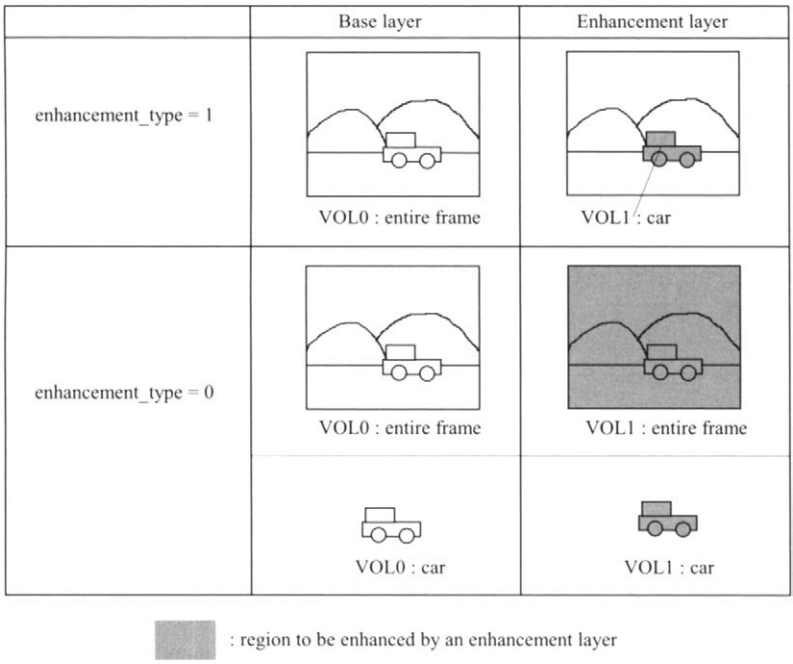


Figure 6.39: Enhancement types for scalability. ©ISO/IEC 1998

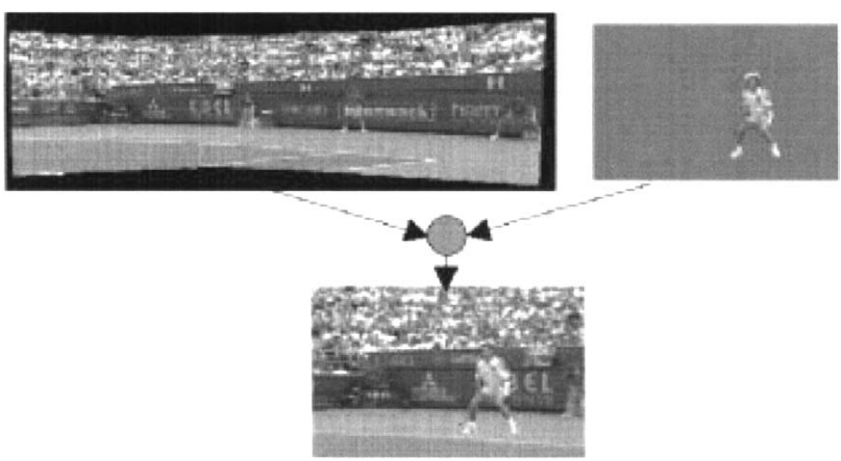


Figure 6.40: Sprite coding of video sequence. ©ISO/IEC 1998

6.6.8.1 Off-line Static Sprites

Off-line Stripe Generation

Off-line sprites, also known as static sprites are built off-line prior to encoding assuming the entire video object from which the sprite is derived is available. They can be directly copied, warped and cropped to generate a particular rendition of the sprite at a particular instant in time. For each VOP in the original video sequence, the global motion field is estimated using one of the following transform methods:

- stationary transform;
- translational transform;
- isotropic transform;
- affine transform; or
- perspective transform.

Each transformation is defined as a set of coefficients or the motion trajectories of some reference points. While the former representation is convenient for performing the transformation, the latter is required for encoding the transformations. Using the global motion parameters, the VOP is registered with the sprite by warping and blending the VOP to the sprite coordinate system. The number of reference points needed to encode the warping parameters determines the transform to be used for warping.

Off-line static sprites are particularly suitable for synthetic video objects and natural video objects undergoing rigid motion when a wallpaper-like rendering is appropriate.

Static Sprite Coding

As static sprite is a still image, the shape and texture of static sprite are treated as an I-VOP and therefore coded as such. Since sprites consists of information needed to reconstruct the background of multiple frames of a video sequence, they are typically much larger than a single frame of the video sequence. Transmitting this large amount of information as the first frame takes time and therefore a significant latency is incurred at the start of the display of a video sequence when large sprites are used.

There are two approaches one can adopt to reduce the latency incurred when large sprite are transmitted:

1. First transmit only portion of the sprite needed to reconstruct the first few frames and transmit the remaining pieces when the decoder requires them subject to the availability of bandwidth.

2. First transmit a low resolution or coarsely quantized sprite to enable the reconstruction of the first few frames and transmit the residual information to progressively build up the image quality as the bandwidth becomes available.

The above two techniques can be employed independently or in combination.

According to the sprite coding syntax, the size of sprite, the location offset of the initial piece of the sprite and the shape information for the entire sprite are transmitted at the Video Object Layer (VOL), while the transmission of the remaining portions of the sprite is done at the Video Object Plane (VOP). At the VOP, the remaining portions of the sprite are sent in small pieces along with the trajectory points. During each frame period, there may be one or more pieces of the sprite being transmitted along with size, location, and the corresponding trajectory points information where for simplicity sake, the size and location information are constrained to be of multiples of 16. The process continues until all the pieces are transmitted. Note that the encoder has the responsibility to ensure the timely delivery of pieces in a way that regions of the sprite are always present at the decoder before they are needed.

The functionality of the syntax also provides for the transmission of the sprite at a lower resolution at times of timing and bandwidth constraint and improves the quality by sending the residual information later. These residual information may be sent in place of or along with other sprite pieces at anytime subject to the bandwidth and timing constraints. The encoder can make the quality update process more efficient by determining the regions to be updated beforehand and send only the residual information when needed.

The global motion information obtained using the transformations as described above are used to represent the warping information instead of the transform coefficients. Specifically, we define a set of reference points $(x_r(n), y_r(n))$ in the current VOP to be coded. The corresponding sprite points $(x'_r(n), y'_r(n))$ in the sprite or in the reference VOP are computed using the global motion parameters estimated by global motion estimation. The sprite points $(x'_r(n), y'_r(n))$ are quantized to half-pel accuracy. The set of reference and sprite points defines the quantized transform. This process is illustrated in Fig. 6.41.

Motion vectors of the reference points which are the corner points of

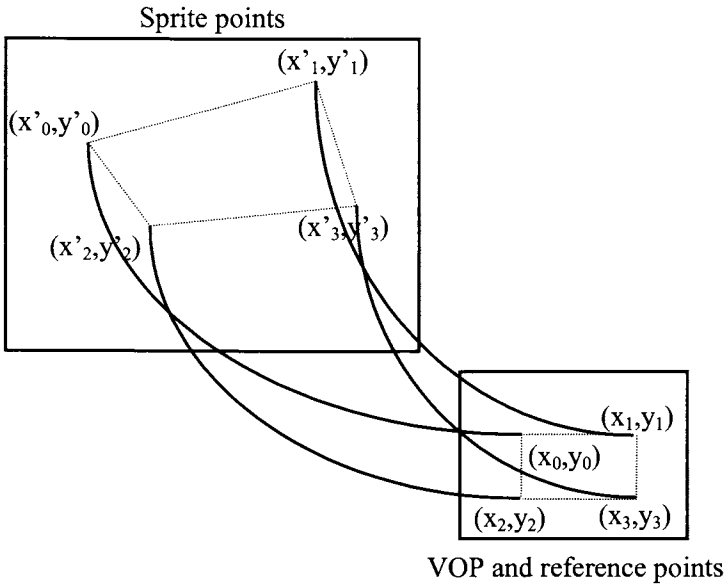


Figure 6.41: Warping of reference points to sprite points. ©ISO/IEC 1998

the bounding rectangle are coded as differential motion vectors. They are transmitted as the global motion parameters for each VOP and are at half-pixel resolution. The actual translation values are retrieved by dividing the decoded values by 2.

To reconstruct the VOP from the sprite, we scan the pixels of the current VOP and compute the corresponding location of this pixel in the sprite using the qnantized transformation described above.

6.6.8.2 On-line Dynamic Sprites

On-line Stripe Generation

On-line sprites or dynamic sprites are generated on-line during coding in both the encoder and the decoder. In on-line sprite coding, the current VOP is used as the reference, from which global motion estimation is performed between successive VOPs. The stripe is updated for each input VOP by being warped with respect to the current VOP coordinates using the estimated motion parameters between two consecutive VOPs. The current sprite is then built by blending the current VOP onto the newly aligned

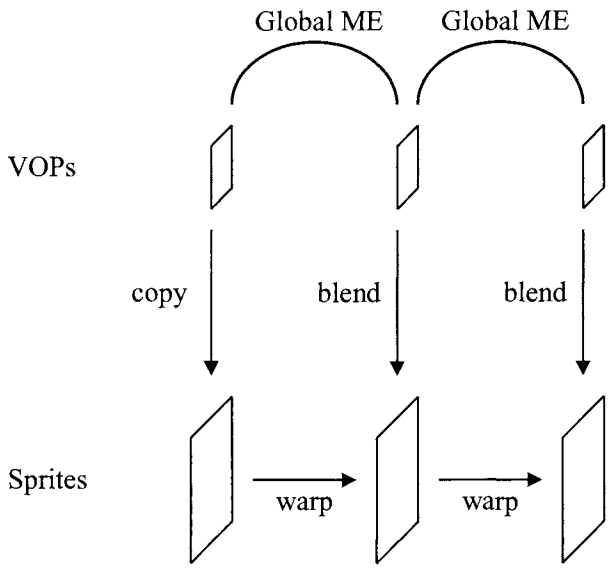


Figure 6.42: On-line dynamic sprite generation process. ©ISO/IEC 1998

sprite. Fig. 6.42 depicts the sprite generation process.

Dynamic Sprite Coding

In the case of dynamic sprites, the sprite is used for predictive coding. The prediction of a MB using the sprite is obtained using the warping parameters and a transform function. The procedure is as follows:

- the coordinates of the MB are scanned;
- using the transform function, the coordinates of the warped pixels in the sprite are found;
- the prediction of the pixel values is obtained by using bilinear transformation.

As the global motion estimation using the transformation produces pixel-wise motion vectors, the candidate motion vector predictor from the reference MB is obtained as the average value of the pixel-wise motion vectors in motion vector coding for MBs in sprite-VOPs. However, there may be regions where sprite content is undefined, therefore padding may be needed as for normal VOPs.

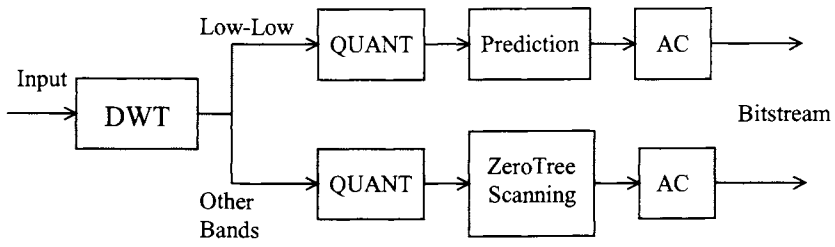


Figure 6.43: Block diagram of the wavelet encoder. ©ISO/IEC 1998

Shape coding in sprite-VOPs is the same as that in P-VOPs.

6.6.9 Still Image Texture Coding

The coding of still images employs zerotree wavelet coding technique. This technique enables the coding of still image textures with a high efficiency and spatial/SNR scalability at fine granularity which can be selected at a wide range of possible levels.

6.6.9.1 The Encoder Structure

Fig. 6.43 shows the structure of the wavelet encoder. The input is decomposed into various subbands by the discrete wavelet transform (DWT). The low-low band is quantized and coded by predictive coding scheme while the other bands are zerotree wavelet coding technique. Both the outputs of the predictive and wavelet coders are then entropy-coded by adaptive arithmetic coder (AC).

6.6.9.2 Discrete Wavelet Transform

The two-separable wavelet decomposition is performed using a Daubechies (9,3) tap biorthogonal filter with the filter coefficients given by Table 6.4. A group delay of 1 and -1 sample is applied to the highpass analysis and highpass synthesis filter, respectively.

Before applying the wavelet decomposition, symmetric extensions are performed at the leading and trailing of the texture data sequences to satisfy the perfect reconstruction criterion of wavelet filtering. Downsampling by a factor of 2 is carried out at each level of decomposition to preserve the total number of samples in the image.

Table 6.4: Coefficients of Daubechies (9,3) tap biorthogonal filter. ©ISO/IEC 1998

Lowpass filter	Highpass filter
0.03314563036812	-0.35355339059327
-0.06629126073624	0.70710678118655
-0.17677669529665	-0.35355339059327
0.41984465132952	
0.99436891104360	
0.41984465132952	
-0.17677669529665	
-0.06629126073624	
0.03314563036812	

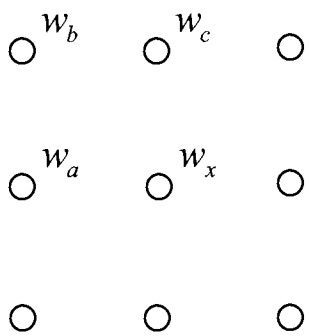


Figure 6.44: Coding of lowest subband coefficients. ©ISO/IEC 1998

6.6.9.3 Coding of the Lowest Subband

The lowest subband (i.e., low-low band) is the most important subband and is encoded independently from other subbands. The encoding technique used is a simple predictive coding scheme, the differential pulse code modulation (DPCM). Quantization of the wavelet coefficients is by an uniform midrise quantizer. The quantized coefficient w_x is predicted from its three nearest neighbors w_a, w_b, and w_c as illustrated in Fig. 6.44.

The prediction rule is as follows:

```
if (|w_a − w_b| < |w_a − w_c|)
    ŵ_x = w_c
    w_x = w_x − ŵ_x
```

```
else
```

$$\hat{w}_x = w_a$$
$$w_x = w_x - \hat{w}_x$$

The coefficients after DPCM are then encoded using an adaptive arithmetic coder. The minimum and maximum values of the coefficients are found. The minimum value is subtracted from all the coefficients to limit their lower bound to zero. The AC model is initiated with an uniform distribution with the maximum value as seeds. The coefficients are then scanned and encoded adaptively by the AC.

6.6.9.4 Zerotree Coding of the Higher Subbands

A multiscale zerotree coding scheme is employed to achieve a wide range of scalability levels as shown in Fig. 6.45. The wavelet coefficients of the first layer are first quantized with the quantizer Q0. The quantized coefficients are zerotree scanned and the significant maps and the coefficients are entropy coded with the AC producing output BS0. The quantized wavelet coefficients of the first layer are also reconstructed and subtracted from the original coefficients forming the coefficients of the second layer. These coefficients are quantized by the quantizer Q1, zerotree scanned and entropy coded producing the output BS1. The quantized wavelet coefficients of the second layer are also reconstructed and subtracted from the original coefficients forming the coefficients of the third layer. The process is repeated until the final Nth layer is reached where $N + 1$ defines the number scalability layers.

Zerotree Scanning

As a result of the wavelet subband decomposition, there exists a parent-child relationship, i.e., high correlation, between wavelet coefficients at the same location across different subbands. With reference to Fig. 6.46, a wavelet tree can be constructed as we scan from the parent in the lowest subband to the higher subbands as indicated by the dotted line.

Zerotree is formed at any node of the wavelet tree if the coefficient is zero and all the node's children are also zero. This is based on the principle if a wavelet coefficient in a lower subband is insignificant, because of the high correlation between parent and children, then all the coefficients in the same location in the higher will also likely to be insignificant.

The wavelet trees are coded by scanning each tree from the root in the lowest subband through the children in the higher subbands, and assigning

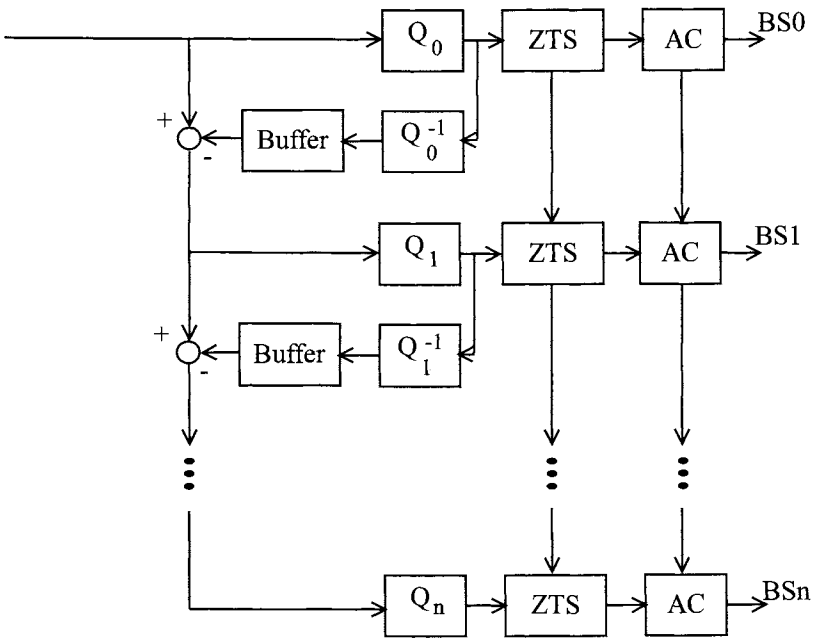


Figure 6.45: Multiscale zerotree coding scheme. ©ISO/IEC 1998

one of three symbols to each node, namely, zerotree root, valued zerotree root, or value. A zerotree root is the coefficient at the root of a zerotree. Zerotrees need not be scanned anymore since all the coefficients are zero. A valued zerotree root is a node where the coefficient has a nonzero amplitude, and all four children are zerotree roots. Scanning terminates at a valued zerotree. A value identifies a coefficient with amplitude either zero or nonzero, but also with some nonzero children. The symbols and the quantized coefficients are encoded using an adaptive arithmetic coder.

6.6.9.5 Quantization

Two quantization schemes are employed levels, i.e., multilevel quantization and bi-level quantization. To achieve a wide range of scalability levels, a multilevel quantizer is used where the quantization levels are defined by the encoder. Different quantization step sizes can be specified for each level of scalability. All higher subband quantizers are uniform midrise quantizers with a dead zone twice the quantizer step size. The multilevel quantization scheme provides a flexible tradeoff between levels and types of scalability, complexity and coding efficiency for any application.

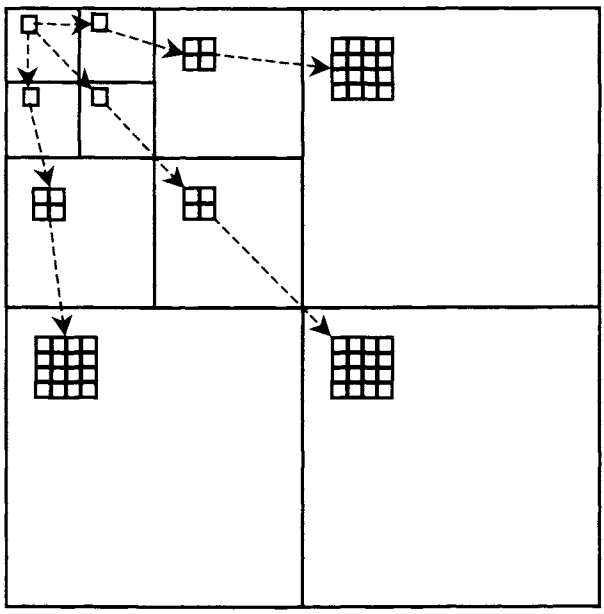


Figure 6.46: The parent-child relationship of wavelet coefficients. ©ISO/IEC 1998

In order to achieve the finest granularity of SNR scalability, a bi-level quantization scheme is used for all the quantizers. This is also a uniform midrise quantizer with a dead zone twice the quantization step size. The coefficients that are outside the dead zone are quantized with a 1-bit accuracy. The number of quantizers is equal to the maximum number of bitplanes in the wavelet coefficient representation.

6.6.9.6 Entropy Coding

The zerotree symbols and quantized wavelet coefficients are entropy-coded using an adaptive arithmetic coder with a three-symbol alphabet. Therefore, at least three different tables, namely, type, valz and valnz, must be codet at the same time. The arithmetic coder must track at least three probability models, one for each table. There may be two more models to track, one for non-zero quantized coefficients of the low-low band and one for the non-zero quantized coefficients of the other three low resolution bands. For each wavelet coefficient, first the coefficient is quantized, then its type and value are calculated, and lastly these values are arithmetic coded.

The probability model of the arithmetic coder is initialized with an uniform distribution and switched appropriately for each table.

6.7 Coding of Synthetic Objects

Synthetic objects can be generated by computer graphics, or formed from natural objects by using a parametric description of the objects. It is the latter type of synthetic objects that MPEG-4 has its focus. In its current version, MPEG-4 provides standards for:

Parametric descriptions of

- a synthetic description of human face and body
- animation streams of the face and body

Static and dynamic mesh coding with texture mapping Texture coding for view dependent applications

6.7.1 Facial Animation

Animation of the face, i.e., the shape, texture and expressions of the face, is controlled by the Facial Description Parameter (FDP) sets and/or Facial Animation Parameter (FAP) sets. The positions of the various feature points on the face as defined in MPEG-4 are shown in Fig. 6.47. Initially, the face object contains a generic face with a neutral expression. Upon receiving the animation parameters, the face can be rendered to produce animation of different facial expressions, movements and speech utterances. Together with the definition parameters, the generic face can be transformed into faces of different shapes and textures. If required, a complete face model, e,g., a wireframe model, can be downloaded via the FDP set.

Note that the face models are not normative. MPEG-4 only standardizes the coding of description and animation parameters when decoded can drive an unlimited range of models. In cases where custom models and specialized interpretation of the FAPs are needed, the Systems Binary Format for Scenes (BIFS) provides the following features to support face animation:

1. FDPs in BIFS - downloadable model data to configure a baseline face model pre-stored in the terminal into a particular face or to install a specific face model at the beginning of a session;

2. Face Animation Table (FAT) within FDPs - downloadable functional mapping from the incoming FAPs to feature control points in the face mesh to control facial movements;

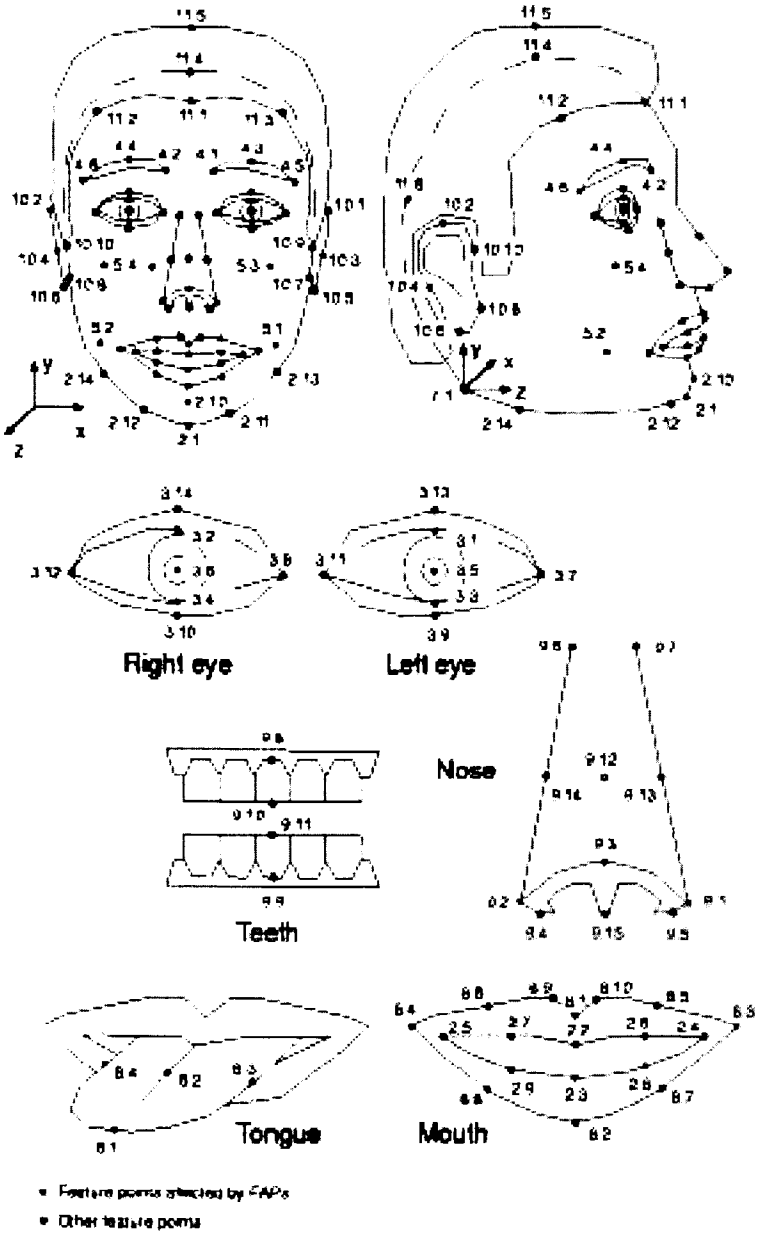


Figure 6.47: Description of the feature points in MPEG-4 [5]

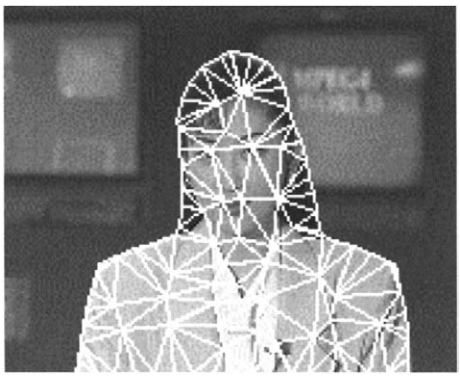


Figure 6.48: An example of 2-D mesh modeling on a video object. ©ISO/IEC 1998

3. Face Interpolation Technique (FIT) in BIFS - downloadable definitions of mapping of incoming FAPs into a total set of FAPs before their applications to feature points which can be used for complex cross-coupling of FAPs to link their effects, or to interpolate FAPs using those available in the terminal.

The above BIFS features enables the customization of face models including the calibration of an established model or the downloading of a new model with its shape, texture and color.

6.7.2 Body Animation

Decoding and scene description of body animation follows the same approach in face animation described in Section 6.7.1 above and is designed to work in an integrated way with it.

6.7.3 2-D Animated Meshes

Triangular mesh has long being used for 3-D object shape modeling and rendering. When projected onto the 2-D image plane, a 2-D mesh results. A 2-D dynamic mesh refers to 2-D mesh geometry and the associate motion information of all its node points at a particular instant of time. Fig. 6.48 shows a 2-D mesh on a human face.

6.7.3.1 Mesh Tracking

Using the dynamic mesh, the image features can be tracked forward in time by non-uniform sampling of the motion field at the node points between the reference mesh and the current mesh. The 2-D mesh may be regular or adapted to the image content, which in the case is known as content-based mesh. Method of selection and tracking of the node points are not subject to standardization.

6.7.3.2 Texture Mapping

Mapping of the texture within the triangular meshes is carried out by first deforming the triangular patches in the reference frame according to the movement indicated by the motion vectors at the node points, and warping the texture within each patch in the reference frame onto the corresponding patches in the current frame. Affine mapping seems to be the popular choice for triangular meshes as it is a linear mapping which can model translation, rotation, scaling, reflection and shear, and at the same time, preserve straight lines at low computational complexity. Moreover, the degree of freedom given by the three motion vectors match well with the six parameters of the affine mapping which means that a continuous affine motion field can be reconstructed from the 2-D motion field represented by the three motion vectors at the node points.

One big advantage of the 2-D mesh modeling is it only requires a single view without range data. However, it is easily extendable to 3-D mesh if the additional information is available. In summary, 2-D mesh representation is able to model the shape and motion of a VOP in a unified framework. It also enables the following content-based functionalities:

Video Object Manipulation

- *Augmented reality*: Merging computer animated video images with natural video images to create enhanced display information. The computer-generated images must be tracked to have perfect synchronization with the moving natural images.
- *Synthetic-object-transfiguration*: Replacing a natural video object in a video clip by another video object extracted from another natural video clip or transfigured from a still image object using the motion information of the object to be replaced.
- *Spatio-temporal interpolation*: Creating intermediate frames by motion-compensated temporal interpolation e.g., in frame rate up-conversion.

Video Object Compression

- Using 2-D mesh modeling, large compression can be achieve by transmitting only texture maps of selected key frames and animate (interpolate) those in intermediate frames using the motion information of the selected frames.

Content-Based Video Indexing

Mesh representation

- enables animated key snapshots for a moving visual synopsis of objects;
- provides accurate object trajectory information that can be used to retrieve visual objects with specific motion; and
- provides vertex-based object shape representation for shape-based object retrieval.

6.8 Error Resilience

MPEG-4 provides error robustness and resilience to allow accessing image or video information over a wide range of storage and transmission media. The error resilience tools developed for MPEG-4 can be divided into three major categories: resynchronization, data recovery, and error concealment.

6.8.1 Resynchronization

The use of variable length codewords in the MPEG-4 bitstream means that there may be loss of synchronization when an error occurs in the bitstream. Resynchronization tools attempt to re-establish synchronization between the decoder and the bitstream after a residual error or errors have been detected. Generally, the data between the synchronization point prior to the error and the first point where synchronization is re-established, is discarded. If the resynchronization approach is effective at localizing the amount of data discarded by the decoder, then the ability of other types of tools which recover data and/or conceal the effects of errors is greatly enhanced.

The resynchronization technique adopted by MPEG-4 is the video packet approach based on providing periodic resynchronization markers throughout the bitstream. In other words, the length of the video packets are not based on the number of macroblocks, but instead on the number of bits contained

in that packet. If the number of bits contained in the current video packet exceeds a predetermined threshold, then a new video packet is created at the start of the next macroblock and a resynchronization marker is inserted. This marker is distinguishable from all possible VLC code words as well as the VOP start code. Header information is also provided at the start of a video packet. Contained in this header is the information necessary to restart the decoding process.

It should be noted that when utilizing the error resilience tools within MPEG-4, some of the compression efficiency tools are modified. For example, all predictively encoded information must be confined within a video packet so as to prevent the propagation of errors.

In conjunction with the video packet approach to resynchronization, a second method called fixed interval synchronization has also been adopted by MPEG-4. This method requires that VOP start codes and resynchronization markers (i.e., the start of a video packet) appear only at legal fixed interval locations in the bitstream. This helps to avoid the problems associated with start codes emulations. That is, when errors are present in a bitstream it is possible for these errors to emulate a VOP start code. In this case, when fixed interval synchronization is utilized the decoder is only required to search for a VOP start code at the beginning of each fixed interval. The fixed interval synchronization method extends this approach to be any predetermined interval.

6.8.2 Data Recovery

After synchronization has been re-establish, data recovery tools attempt to recover data that in general would be lost. These tools are not simply error correcting codes, but instead techniques which encode the data in an error resilient manner. For instance, one particular tool that has been endorsed by the Video Group is Reversible Variable Length Codes (RVLC). In this approach, the variable length code words are designed such that they can be read both in the forward as well as the reverse direction, as depicted in Fig. 6.49. Obviously, this approach reduces the compression efficiency achievable by the entropy encoder. However, the improvement in error resiliency is substantial.

6.8.3 Error Concealment

Error concealment is an important component of any error robust video codec. The effectiveness of a error concealment strategy is highly dependent on the performance of the resynchronization scheme. Basically, if the

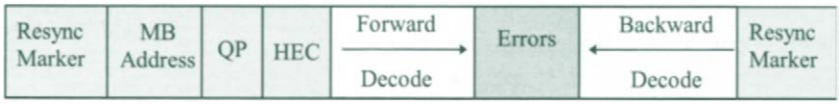


Figure 6.49: Example of Reversible Variable Length Code. ©ISO/IEC 1998

Table 6.5: Summary of error resilience modes of operation. ©ISO/IEC 1998

Mode	VOP Type	Resync Marker	RVLC	Data Partitioning (Motion Marker)
Low Complexity	I-VOP	Mandatory	Optional	Not used
	P-VOP	Mandatory	Optional	Not used
Medium Complexity	I-VOP	Mandatory	Optional	Mandatory
	P-VOP	Mandatory	Optional	Mandatory
High Complexity	I-VOP	Mandatory	Mandatory	Mandatory
	P-VOP	Mandatory	Mandatory	Mandatory

resynchronization method can effectively localize the error then the error concealment problem becomes much more tractable. For low bitrate, low delay applications the current resynchronization scheme provides very acceptable results with a simple concealment strategy, such as copying blocks from the previous frame.

An additional error resilient mode that further improves the ability of the decoder to localize an error is data partitioning. This approach requires that a second resynchronization marker be inserted between motion and texture information. If the texture information is lost, this approach utilizes the motion information to conceal these errors by motion compensating the previous decoded VOP.

6.8.4 Modes of Operation

There are several modes of opertion providing various options depending on the complexity of the scene to be encoded. Essentially there are three modes of operation, namely, low complexity mode, medium complexity mode and the high complexity mode. A summary of these various modes is given in Table 6.5.

Table 6.6: Suggested Resynchronization Marker Spacing. ©ISO/IEC 1998

Bit Rate (kbit/s)	Spacing (bits)
0-24	480
25-48	736
49-128	1500
128-512	TBD
512-1000	TBD

6.8.5 Error Resilience Encoding Tools

6.8.5.1 Resynchronization Markers

The resynchronization markers should be inserted by the encoder before the first macroblock after the number of bits output since the last resync_marker field exceeds a predetermined value. The value used for this spacing is dependent on the anticipated error conditions of the transmission channel and compressed data rate. Suggested values for this spacing are provided in Table 6.6. These values are obtained experimentally and provide good results for a variety of error conditions encountered in error-prone channels such as wireless fading channels. It is highly recommended that users of MPEG-4's error resilient tools adjust this spacing of the resynchronization markers to fit the error conditions of their particular channel.

As shown in Fig. 6.49, in addition to the *resync_marker* field, the encoder also inserts a field indicating the current macroblock address (MB address), a field indicating the current quantization parameter (QP) and a Header Extension Code (HEC). This additional information is provided to the decoder enabling it to determine which VOP a resync packet belongs to in case the VOP start_code is lost

6.8.5.2 Data Partitioning

When the data partitioning is used, then in addition to the *resync_marker* fields, MB address, QP and HEC, a *motion_marker* field is inserted after the motion data (before the beginning of the texture data). This *motion_marker* field is unique from the motion data and enables the decoder to determine when all the motion information has been received correctly.

6.8.5.3 Reversible VLCs

The use of reversible VLCs enables the decoder to recover additional texture information in the presence of errors. This is accomplished by first detecting the error and searching forward to the next *resync_marker*, once this point is determined the texture data can be read in the reverse direction until an error is detected. When errors are detected in the texture data, the decoder can use the correctly decoded motion vector information to perform motion compensation and conceal these errors.

6.8.5.4 Decoder Operation

When an error is detected in the bitstream, the decoder should resynchronize at the next suitable resynchronization point. Where a VOP header is missed or received with obvious errors, this should be the next VOP *start_code*. Otherwise, the next resynchronization point in the bitstream should be used.

Under the following error conditions, the baseline decoder should resynchronize at the next resynchronization point in the bitstream:

1. An illegal VLC is received.
2. More than 64 DCT coefficients are decoded in a single block.
3. Inconsistent resynchronization header information (i.e., QP out of range, $MBN(k) < MBN(k-1)$, etc.)
4. Resynchronization marker is corrupted.

Under the following error conditions, the decoder should resynchronize at the next VOP header:

- VOP start code corrupted.

For other resynchronization techniques, conditions for error detection and resynchronization should be as close as possible to those outlined above. Missing blocks should be replaced with the same block from the previous frame.

References

[1] ISO/IEC 14496-2, "Information technology - generic coding of audio-visual objects (final draft of international standard)," Dec. 1998.

[2] R. Koenen, "Overview of MPEG-4 standard," in *ISO/IEC JTC1/SC29/WG11 MPEG98/N2459, Atlantic City, USA*, Oct. 1998.

[3] ISO/IEC, "Managing intellectual property identification and protection within MPEG-4," in *ISO/IEC JTC1/SC29/WG11 MPEG97/N1918, Fribourg, Switzerland*, Oct. 1997.

[4] MPEG Video Group, "MPEG-4 video verification model version 11.0," in *ISO/IEC JTC1/SC29/WG11 MPEG98/N2172, Tokyo, Japan*, Mar. 1998.

[5] F. Lavagetto, R. Pockaj, and M. Costa, "MPEG-4 compliant calibration of 3d head models," in *Int. Picture Coding Symposium, PCS'99, Portland, Oregon, USA*, Apr. 1999, pp. 217–220.