ITU-T Video Coding Standards

Deepak Turaga and Tsuhan Chen

Electrical and Computer Engineering, Carnegie Mellon University

{dturaga, tsuhan}@ece.cmu.edu

0.1 Introduction

Standards define a common language that different parties can use, so that they can communicate with one another. Standards are thus, a prerequisite to effective communication. Video coding standards define the bitstream syntax, the language that the encoder and the decoder use to communicate. Besides defining the bitstream syntax, video coding standards are also required to be efficient, in that they should support good compression algorithms as well as allow the efficient implementation of the encoder and decoder.

In this chapter we are introducing the ITU-T video coding standards with the focus being on the latest version, H.263. This version is also known as H.263 Version 2, or H.263+, as opposed to an earlier version of H.263. Whenever we say H.263 in this chapter, we mean the latest version.

This chapter is organized as follows. Section 0.2 defines what a standard is and the need for standards. It also lists some of the prevalent standards and organizations involved in developing them. Section 0.3 talks of the fundamental components of video coding standards in general and also some specifics regarding the H.263 standard. The basic concepts of motion compensation, transform coding and entropy coding are introduced. The section concludes with a overall block diagram of the video encoder and decoder. Section 0.4 is specific to H.263 and describes the optional modes that are available with it. These options are further grouped into options for better picture quality, options for added error resilience and options for scalabilities. There are some other miscellaneous options that are also described. There is also some discussion on the levels of preferred support and other supplemental information. Section 0.5 is the conclusion and includes some general remarks and further sources of information.

0.2 Fundamentals of Standards

Multimedia communication is greatly dependent on good standards. The presence of standards allows for a larger volume of information exchange, thereby benefiting the equipment manufacturers and service providers. It also benefits customers, as now they have a greater freedom to choose between manufacturers. All in all, standards are a prerequisite to multimedia communication. Standards for video coding are also required to be efficient for the compression of video content. This is because a large number of bits are required for the transmission of uncompressed video data.

The H.263 version 2 standard [1] belongs to a category of standards called voluntary standards. These standards are defined by volunteers in open committees and are agreed upon based on the consensus of all the committee members. They are driven by market needs and try to stay ahead of the development of technologies. H.263 is the latest in the series of low bit rate video coding standards developed by ITU-T and was adopted in 1996 [3]. It combined the features of MPEG and H.261[2] (an earlier standard developed in 1990) for very low bit rate coding. H.263 version 2 or H.263+ was adopted in early 1998 and is the current prevailing standard from ITU-T. This standard is the focus of this chapter and whenever we say H.263 we are referring to H.263 version 2.

Another major organization involved in the development of standards is the International Organization for Standardization (ISO). Both these organizations have defined different standards for video coding. These different standards are summarized in Table 1. The major differences between these standards lie in the operating bit-rates and the applications they are targeted for. Each standard allows for operating at a wide range of bit-rates, hence each can be used for a range of applications. All the standards follow a similar framework in terms of the coding algorithms, however there are differences in the ranges of parameters and some specific coding modes.

Table 1

Video Coding Standards Developed by different Organizations

Standards Organization	Video Coding Standard	Typical Range of Bit Rates	Typical Applications
ITU-T	H.261	p×64 kbits/s, p=1…30	ISDN Video Phone
ISO	IS 11172-2 MPEG-1 Video	1.2 Mbits/s	CD-ROM
ISO	IS 13818-2 MPEG-2 Video[1]	4-80 Mbits/s	SDTV, HDTV
ITU-T	H.263	??	PSTN Video Phone
ISO	CD 14496-2 MPEG-4 Video	24-1024 kbits/s	A wide range of applications
ITU-T	H.26L	< 64 kbits/s	A wide range of applications

[1] ITU-T also actively participated in the development of MPEG-2 Video. In fact, ITU-T H.262 refers to the same standard and uses the same text as IS 13818-2.

For a manufacturer to build a standard compliant codec, it is very important to look at the bitstream syntax and to understand what each layer corresponds to and what each bit represents. This approach is, however, not necessary to understand the process of video coding. In order to have an overview of the standard, it suffices to look at the coding algorithms that generate the standard compliant bitstream. This approach emphasizes an understanding of the various components of the codec and the functions they perform. Such an approach helps in understanding the video coding process as a whole. This chapter focuses on the second approach.

0.3 Basics of Video Coding

Video coding involves not only translation to a common language, but also tries to achieve compression. This compression is achieved by trying to eliminate redundancy in the video data. There are two kinds of redundancies present in video data. The first kind of redundancies is spatial, while the second kind is temporal. Spatial redundancy refers to the correlation present between different parts of a frame. Removal of spatial redundancy, thereby involves looking within a frame and is hence referred to as Intra Coding. Temporal redundancies, on the other hand are the redundancies present between frames. At a sufficiently high frame rate it is quite likely that successive frames in the video sequence, are very similar. Hence, removal of such temporal redundancy involves looking between frames and is called Inter Coding. Spatial redundancy is removed through the use of Transform Coding techniques. Temporal redundancy is removed through the use of Motion Estimation and Compensation techniques.

0.3.1 Source Picture Formats and Positions of Samples

In order to implement the standard, it is very important to know the picture formats that the standard supports and positions of the samples in the pictures. The samples are also referred to as pixels (picture elements) or pels. Source picture formats are defined in terms of the number of pixels per line, the number of lines per picture and the pixel aspect ratio. H.263 allows for the use of five standardized picture formats. These are the CIF (Common Intermediate Format), QCIF (Quarter-CIF), sub-QCIF, 4CIF and 16CIF. Besides these standardized formats it also allows support for custom picture formats that can be negotiated. Details about the 5 standardized picture formats are summarized in Table 2.

Table 2

Standard picture Formats Supported by H.263

	Sub-QCIF	QCIF	CIF	4CIF	16CIF
No. of Pixels per Line	128	176	352	704	1408
No. of Lines	96	144	288	576	1152
Uncompressed Bit Rate (at 30 Hz)	4.4 Mb/s	9.1 Mb/s	37 Mb/s	146 Mb/s	584 Mb/s

The pixel aspect ratio is defined in the recommendations for H.261 as 12:11. Using this, it can be seen that all the standard picture formats defined in the table above cover an area that has an aspect ratio of 4:3.

Each sample or pixel consists of three components, a luminance or Y component and two chrominance or C_B and C_R components. The values of these components are as defined in [4]. As an example, “black” is represented by Y = 16, while “white” is represented by Y = 235, while the values of C_B and C_R lie in the range 16 to 240. C_B and C_R values of 128 represent zero color difference or a gray region. The picture formats shown in Table 2 define the resolution of the Y component. As it is known that the human eyes are less sensitive to the chrominance components, these components typically have only half the resolution, both horizontally and vertically, of the Y component. This is referred to as the 4:2:0 format. Each C_B or C_R pel lies at the center of four neighboring Y pels. This is shown in Figure 1. The block edges can lie in between rows or columns of Y pels.

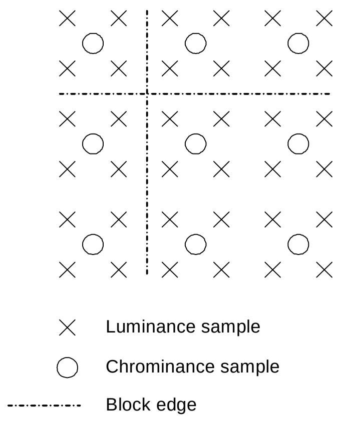


Figure 1. Positions of luminance and chrominance samples

As was mentioned before H.263 allows support for negotiable custom picture formats. Custom picture formats can have any number of pixels per line and any number of lines in the picture. The only constraint applied is that the number of pixels per line should be a multiple of 4 in the range [4,..2048] and the number of lines per picture should also be a multiple of 4 in the range [4,..1152]. Custom picture formats are also allowed to have custom pixel aspect ratios and this is shown in Table 3.

Table 3

Different Pixel Aspect Ratios Supported by H.263

Pixel Aspect Ratio	Pixel Width : Pixel Height
Square	1:1
CIF	12:11
525-type for 4:3 picture	10:11
CIF for 16:9 picture	16:11
525-type for 16:9 picture	40:33
Extended PAR	m:n, m and n are relatively prime

These pictures or frames occur at a certain rate to make the video sequence. The standard specifies that all decoders and encoders should be able to use the standard CIF picture clock frequency (PCF). The PCF is 30000/1001 frames per second for CIF. It is also allowed for decoders and encoders to have custom PCF, even higher than 30 frames per second.

0.3.2 Blocks, Macroblocks and Groups of Blocks

H.263 uses block based coding schemes. In these schemes, the pictures are sub-divided into smaller units called blocks that are processed one by one, both by the decoder and the encoder. These blocks are processed in the scan order as shown in the following figure

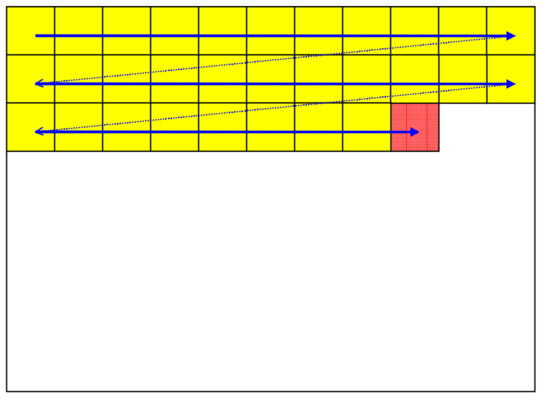


Figure 2. Scan order of blocks

A block is defined as a set of 8x8 pixels. As the chrominance components are downsampled, each chrominance block corresponds to four Y blocks. The collection of these 6 blocks is called a macroblock (MB). A MB is treated as a unit during the coding process.

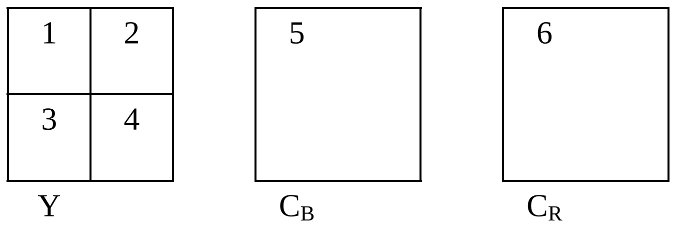


Figure 3. Blocks in a Macroblock

A number of MBs are grouped together into a unit called a Group of Blocks (GOB). The H.263 allows for a GOB to contain one or more rows of MBs. This shown in Figure 4

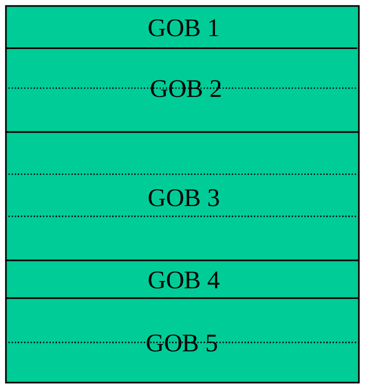


Figure 4. Example GOB Structure for a QCIF picture

The optional slice structured mode allows for the grouping of MBs into slices, which may have arbitrary number of MBs grouped together. More about the slice structured mode is in Section 0.4.2.1.

0.3.3 Compression Algorithms

Compression involves removal of spatial and temporal redundancy. The H.263 standard uses the Discrete Cosine Transform to remove spatial redundancy and motion estimation and compensation to remove temporal redundancy. These techniques are discussed in the following sections.

0.3.3.1 Transform Coding

Transform coding has been widely used to remove redundancy between data samples. In transform coding, a set of data samples is first linearly transformed into a set of *transform coefficients*. These coefficients are then quantized and entropy coded. A proper linear transform can de-correlate the input samples, and hence remove the redundancy. Another way to look at this is that a properly chosen transform can concentrate the energy of input samples into a small number of transform coefficients, so that resulting coefficients are easier to encode than the original samples.

The most commonly used transform for video coding is the discrete cosine transform (DCT)[5] [6]. Both in terms of objective coding gain and subjective quality, DCT performs very well for typical image data. The DCT operation can be expressed in terms of matrix multiplication:

$$\mathbf{Y} = \mathbf{C}^T \mathbf{X}\mathbf{C}$$

where $\mathbf{X}$ represents the original image block, and $\mathbf{Y}$ represents the resulting DCT coefficients. The elements of $\mathbf{C}$, for an 8×8 image block, are defined as

$$C_{mn} = k_n \cos\left[\frac{(2m+1)n\pi}{16}\right] \text{ where } k_n = \begin{cases} 1\big/(2\sqrt{2}) & \text{when } n = 0 \\ 1/2 & \text{otherwise} \end{cases}$$

After the transform, the DCT coefficients in $\mathbf{Y}$ are quantized. Quantization involves loss of information, and is the operation most responsible for the compression. The quantization step size can be adjusted based on the available bit rate and the coding modes chosen. Except for the intra DC coefficients that are uniformly quantized with a step size of 8, a “dead zone” is used while quantizing all other coefficients. This is done in order to remove noise around zero. The input-output relations for the two cases are shown in Figure 5.

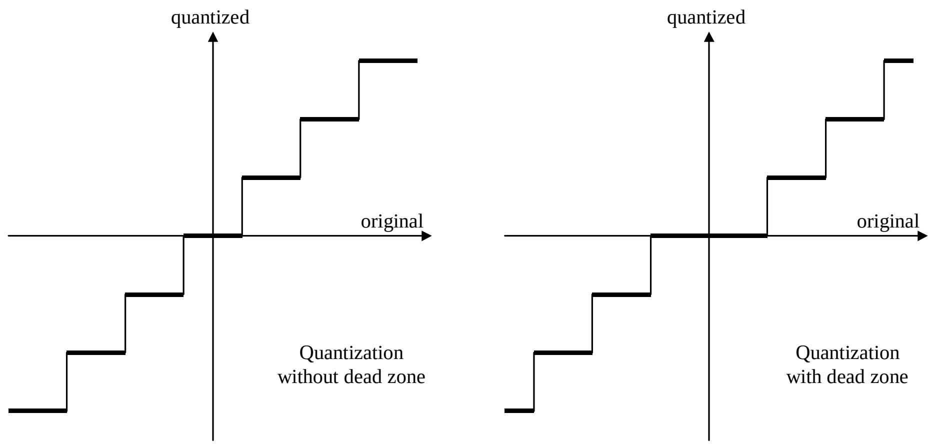


Figure 5. Quantization with and without "dead zone"

The quantized 8×8 DCT coefficients are then converted into a one-dimensional (1D) array for entropy coding. Figure 5 shows the scan order used in H.261 for this conversion. Most of the energy concentrates on the low frequency coefficients, and the high frequency coefficients are usually very small and are quantized to zero before the scanning process. Therefore, the scan order in Figure 6 can create long runs of zero coefficients, which is important for efficient entropy coding, as we will discuss in the next paragraph.

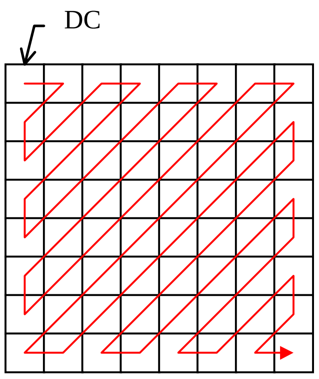


Figure 6. Scan order of the DCT coefficients

The resulting 1D array is then decomposed into segments, with each segment containing some (this number may be zero) zeros followed by a nonzero coefficient. Let an *event* represent the three values (*run*, *level*, *last*). “Run” represents the number of zeros; “level” represents the magnitude of the nonzero coefficient following the zeros and “last” is an indication of whether the current non-zero coefficient is the last non-zero coefficient in the block. A Huffman coding table is built to represent each event by a specific codeword, i.e., a sequence of bits. Events that occur more often are represented by shorter codewords, and less frequent events are represented by longer codewords. So, the table is often called a variable length coding (VLC) table. This coding process is sometimes called “run-length coding.” An example of the VLC table is shown in the Table 4. The transform coefficients in this table correspond to input samples chosen as the residues after motion compensation, which will be discussed in the following section.

Table 4

Partial VLC Table for DCT coefficients

LAST	RUN	|LEVEL|	VLC CODE
0	0	1	10s
0	0	2	1111 s
0	0	3	0101 01s
0	0	4	0010 111s
0	0	5	0001 1111 s
0	0	6	0001 0010 1s
0	0	7	0001 0010 0s
0	0	8	0000 1000 01s
0	0	9	0000 1000 00s
0	0	10	0000 0000 111s
0	0	11	0000 0000 110s

In the above table the third column represents the magnitude of the level. The sign bit added at the end of the VLC code, represented as s takes care of the sign of the level. It can be seen from the table that more frequently occurring symbols (for instance symbols with smaller magnitudes) are assigned fewer bits than the less frequently occurring symbols. It is reasonable to assume that smaller magnitude symbols occur more frequently than the large magnitude symbols, because most of the time we code the residue found after motion compensation (discussed in the next section) and this residue does tend to have small magnitudes.

0.3.3.2 Motion Compensation

Motion compensation involves removing the temporal redundancy present in video sequences. When the frame rate is sufficiently high, there is a great amount of similarity between neighboring frames. It is more efficient to code the difference between frames, rather than the frames themselves. An estimate for the frame being coded is obtained from the previous frame and the difference between the prediction and the current frame is sent. This concept is similar to Predictive Coding and Differential Coding techniques.

Most video sequences have moving objects in them and this motion is one of the reasons for the difference between successive frames. If there were no motion of objects between frames, these frames would be very similar. The basic idea behind motion compensation is to estimate this motion of objects and to use this information to build a prediction for successive frames. H.263 supports block based motion estimation and compensation. Motion compensation is done at the MB level, except in the advanced modes, when it is done at the block level. The process involves looking at every MB in the current frame and trying to find a match for it in the previous frame. Each MB in the current frame is compared to 16x16 areas in a specified search space in the previous frame, and the best matching area is selected. This area is then offset by the displacement between its position and the position of the current MB. This forms a prediction of the current MB. In most cases, it is not possible to find an exact match for the current MB. The prediction area is usually similar to the MB and the difference or residue between the two is small. Similarly, predictions for all the MBs in the current frame are obtained and the prediction frame is constructed. The residue frame is computed and is expected to have a much smaller energy than the original frame. This residue frame is then coded using the transform coding procedure. More information about motion compensation can be found in [7] and [8]. The process of motion compensation is highlighted in Figure 7.

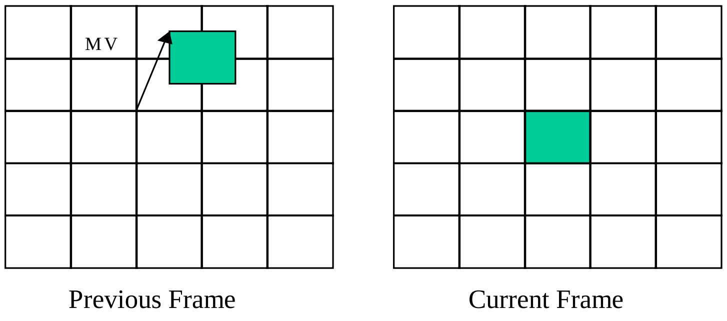


Figure 7. Motion Compensation

In the above figure, the gray block on the right corresponds to the current block being coded and the gray area on the left represents the best match found for the current block, in the previous frame. A major part of the encoding process involves finding these best matches or equivalently the offsets between the position of the best match and the position of the current block. This process of searching for the best matches is called Motion Estimation.

These offsets between the position of the best match and the position of the current MB are called motion vectors. H.263 allows these motion vectors to have non-integral values. For instance the motion vector for a MB may have the value (1.5, -4.5), which means that the best match for the current block has pixels that lie at non-integral positions. All pixels in the previous frame are at integer pixel positions. Hence pixels at non-integral positions have values that are computed from the original pixel values using bilinear interpolation. The pixel positions and their interpolated values are shown in Figure 8.

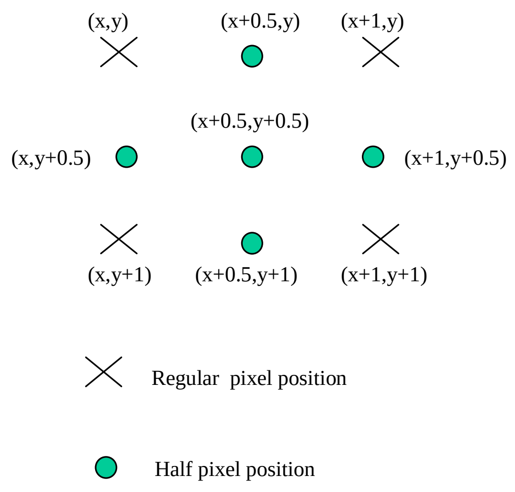


Figure 8. Half pixel positions

It is possible that for a particular MB, we cannot find a good match, in which case the residue error between the best match and the MB itself may have as much energy as the original MB. For such cases, it is better to not do motion compensation. The encoder has the flexibility to decide for each MB, whether it wants to do motion compensation or transform coding. However, in most cases, a saving is accomplished in the bits to code the residue. Besides the residue we also need to send the information regarding construction of the prediction frame to the decoder. This information is basically the motion vectors. The decoder can then use these to reconstruct the prediction frame and add to it the residue (this is transform coded) to obtain the current frame. In order to avoid a large increase in the bitstream because of these motion vectors, these are also differentially coded. The motion vectors of three neighboring MBs (one to the left, one above and one above right) are used as predictors for the motion vector of the current MB. The prediction is formed by taking the median of these three motion vectors. The prediction error between the actual motion vector and the predicted value in the horizontal direction and the vertical direction is coded. This is shown in Figure 9.

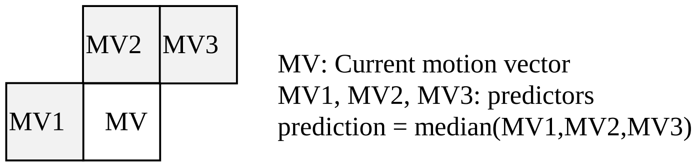

MV2
MV3
MV1
MV
MV: Current motion vector
MV1, MV2, MV3: predictors
prediction = median(MV1,MV2,MV3)

Figure 9. Prediction of motion Vectors

Special cases are needed to take care of MBs for which the predictors lie outside the picture or GOB boundary. These special cases are shown in Figure 10.

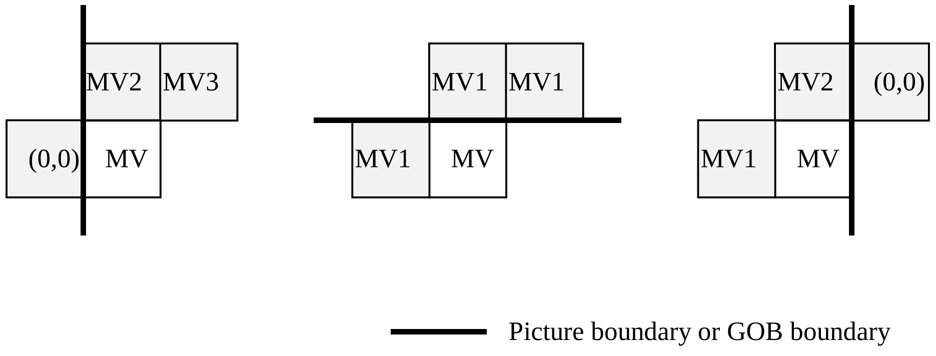


Figure 10. Special Case Motion Vector prediction

Whenever one of the prediction MBs lies outside the picture or GOB boundary, it is replaced by (0,0), however, when two MBs lie outside, they are replaced by the motion vector of the third MB. This is done to avoid having two of these motion vectors replaced by zeros, in which case the final value got after the median operation will be (0,0). The prediction error in motion vectors is also coded using a VLC table. Part of the table is shown in Table 5.

Table 5

Part of the VLC Table for Motion Vector Differences

Vector	differenc	Codes
-5	27	0000 0100 11
-4.5	27.5	0000 0101 01
-4	28	0000 0101 11
-3.5	28.5	0000 0111
-3	29	0000 1001
-2.5	29.5	0000 1011
-2	30	0000 111
-1.5	30.5	0001 1
-1	31	0011
-0.5	31.5	011
0		1
0.5	-31.5	010
1	-31	0010
1.5	-30.5	0001 0
2	-30	0000 110
2.5	-29.5	0000 1010
3	-29	0000 1000
3.5	-28.5	0000 0110
4	-28	0000 0101 10
4.5	-27.5	0000 0101 00
5	-27	0000 0100 10

It can be seen from Table 5 that there are two motion vector difference values that correspond to the same codeword. These can be separated by the knowledge that the final decoded motion vector should lie in the range [-16,15.5]. The exploitation of this fact leads to additional savings in terms of bit rate.

0.3.3.3 Summary

All these basic techniques form part of the baseline options that are specified by H.263. There are other advanced negotiable options, which we will discuss in the next section. These basic coding algorithms may be put together to form a block diagram representation of the video encoder and decoder. This is shown in the Figure 11.

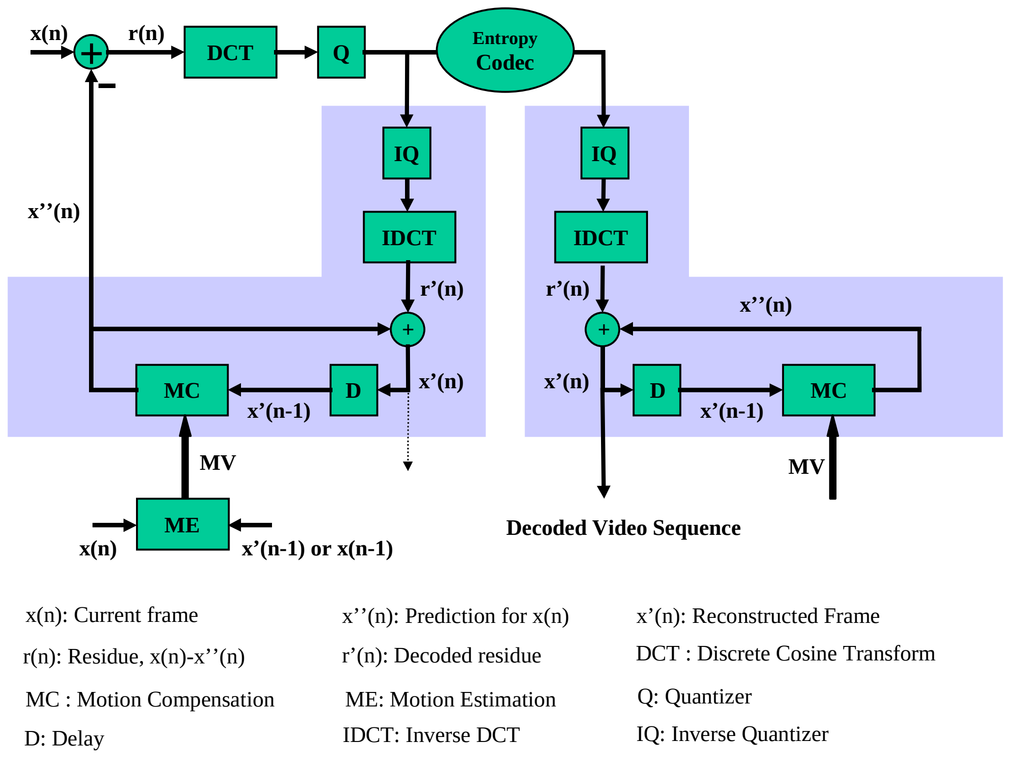


Figure 11. Block Diagram of Encoder and Decoder

Based on some experiments conducted by us, using baseline H.263 provides us with perceptually good quality for simple talking head sequences at ~ 400 bytes per frame and sequences with moderate motion at ~ 1000 bytes per frame. In terms of compression ratios this translates to 100 for low motion sequences and 40 for moderate motion sequences.

0.4 Negotiable Options

H.263 allows for several advanced negotiable options. At the start of a communication session, the decoder signals the encoder regarding what all options it can support. If the encoder also supports those options these may be turned on. These advanced options are provided for better coding efficiency, error resilience and include some enhancements for scalabilities. These are described in the following section.

0.4.1 Coding Modes for Efficiency or Improved Picture Quality

The modes described in this section try to improve the coding efficiency by using better coding algorithms or try to improve the quality of the decoded pictures.

0.4.1.1 Unrestricted Motion Vector (UMV) Mode

In the baseline prediction mode of H.263 all motion vectors are restricted to the picture boundaries. This restriction is removed in this mode and pixels outside of the picture boundaries are referred to. Clearly these are not originally defined. These are obtained through the process of extrapolation. When a pixel referenced by a motion vector lies outside the coded picture area, an edge pixel is used instead. Another feature of this mode is the fact that it allows for an extension to the motion vector range. In the baseline mode, motion vectors are restricted to [-16, 15.5]. In the UMV mode, this range is extended to [-31.5, 31.5]. A larger motion vector range is very useful when the scene being encoded is high motion (especially when there is great camera motion) or the frame rate is low. In both cases large amounts of motion can occur between successive frames and hence the best match is likely to be further displaced from the current block, necessitating a large search range.

0.4.1.2 Syntax Based Arithmetic Coding (SAC)

This option deals with the process of entropy coding and decoding i.e. converting symbols to bits and back. Instead of using the Huffman-like Entropy Coding VLC tables, the encoder and decoder use arithmetic coding to generate the bitstream. Arithmetic coding is more efficient than the VLC and results in savings in the size of the bitstream. Based on some experiments that we conducted, we have found that SAC results in savings of around 10-20% for intra frames and around 1-3% for inter frames. The use of arithmetic coding generates a bitstream different from the VLC generated bitstream, however, the quality of the reconstructed pictures is the same. Some more information about Arithmetic coding can be obtained from [9].

0.4.1.3 Advanced Prediction Mode

This mode contains two basic features. The first feature supported by this mode is the use of four motion vectors per MB. The second feature supported is Overlapped Block Motion Compensation (OBMC). OBMC involves using motion vectors of neighboring blocks to reconstruct a block, thereby leading to an overall smoothing of the image and removal of blocking artifacts. Both these modes are discussed in more detail in the subsections. Like in the UMV mode, this mode also allows motion vectors to cross picture boundaries. Pixels outside the coded area are obtained by extrapolating, as in the UMV case. The extension of motion vector range, however, is not automatically turned on.

0.4.1.3.1 Four Motion Vectors per Macroblock

Each luminance block in the MB is allowed to have it's own motion vector. This allows greater flexibility in obtaining a best match for the MB. It is now possible to find really good matches for each of the four parts of the MB and hence, when these parts are put together, a much better prediction for the MB is obtained. Having four motion vectors also means more motion vector data has to be sent to the decoder. If the savings in the residue for the MB are offset by the extra bits needed to send the four motion vectors, then there is no point in sending four motion vectors. Hence, the encoder needs to be able to intelligently decide whether it wants to send one motion vector or four motion vectors for every MB. With four motion vectors per MB it is no longer possible to code the motion vector difference using the same scheme as in the baseline case. A new set of predictors is defined in the standard. These are shown in Figure 12.

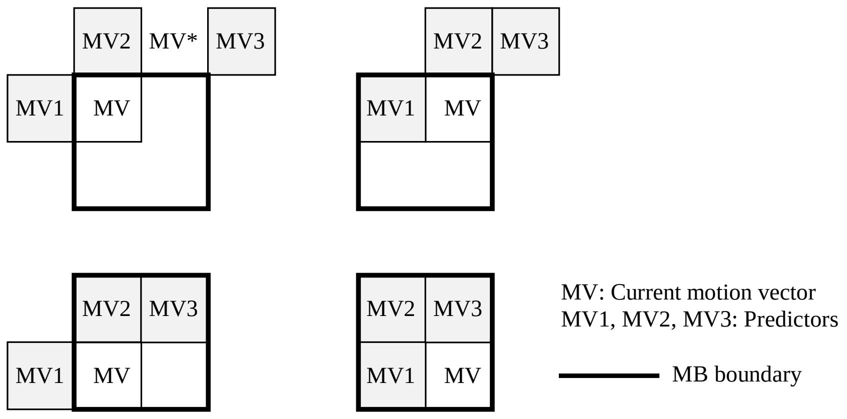


Figure 12. Redefinition of Motion Vector Prediction

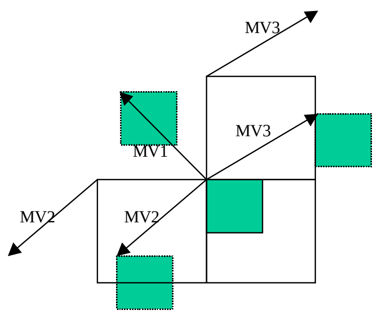
The predictors are chosen such that none of them are redundant. For instance if MV* was chosen as a predictor for the upper left block of the MB, it would probably make the choice of MV1 redundant. This is because it is quite likely that MV2 and MV* are close to one another, as they come from the same MB. Hence, the information obtained from MV1 is suppressed by the median operation. Such redundancy is avoided by picking a motion vector form another MB.

0.4.1.3.2 Overlapped Block Motion Compensation (OBMC)

Every pixel in the final prediction region for a block is obtained as a weighted sum of three values [10]. These values are obtained using the motion vector of the current block and two out of four remote motion vectors. The remote motion vectors are the motion vectors of the block above, the block below, the block to the left and the block to the right of the current block. The three values that are linearly combined to get the final pixel value are:

a) The pixel in the previous frame at a displacement given by the motion vector of the current block.
b) Two pixels in the previous frame at displacements given by the two remote motion vectors.

The weights are predefined in the standard. The choice of two out of four remote motion vectors is made by the location of the pixel in the current block. Each pixel picks the remote vectors of the two neighboring blocks closest to it. For instance, a pixel in the top left part of the block picks the remote motion vectors as the motion vectors of the blocks above and to the left of the current block.



MV1: Motion Vector of current block MV2: Motion Vector of block to the left

MV3: Motion Vector of block above

Figure 13. OBMC for upper left half of block

The overall effect of using the neighboring motion vectors to make a prediction is to smooth the prediction. This reduces blockiness and leads to better predictions, thereby finally resulting in a smaller bitstream.

0.4.1.4 PB Frame Mode

A PB frame is a unit that is made of two frames, one of which is called the P picture and the other is called the B picture. Both these pictures are decoded together and are treated as a single entity. This mode supports three different pictures: the I picture, the P picture and the B picture. An I picture is Intra coded and hence is not predicted from any other picture. A P picture is predicted from the previous P picture or from an I picture, like Inter coded frames in the baseline mode. A B picture is called so, because parts of it may be bidirectionally predicted. The B picture is obtained by looking at forward motion vectors, backward motion vectors and delta motion vectors. The forward and backward motion vectors for the B picture are derived from the motion vectors of the P picture. Forward and backward motion vectors are calculated such that the motion of a block across the sequence of P-B-P pictures appears smooth. As an example consider a block in the P picture that is displaced by 4 in the horizontal direction from the previous P picture. This corresponding B picture block should be displaced by 2 as compared to the previous P picture and –2 as compared to the following P picture, in the horizontal direction. These values correspond to the forward and backward motion vectors. This assumes that the B picture lies exactly between two successive P pictures in time reference, e.g. if no frames are skipped and the original frames are coded as a sequence of P-B-P pictures. Another assumption made in the discussion above is that half the motion occurs between the first P picture and the B picture and the remaining half of motion occurs between the B picture and the second P picture. This may not necessarily be true. There may be accelerated motion across these frames. The delta motion vectors are introduced to take care of such acceleration effects.

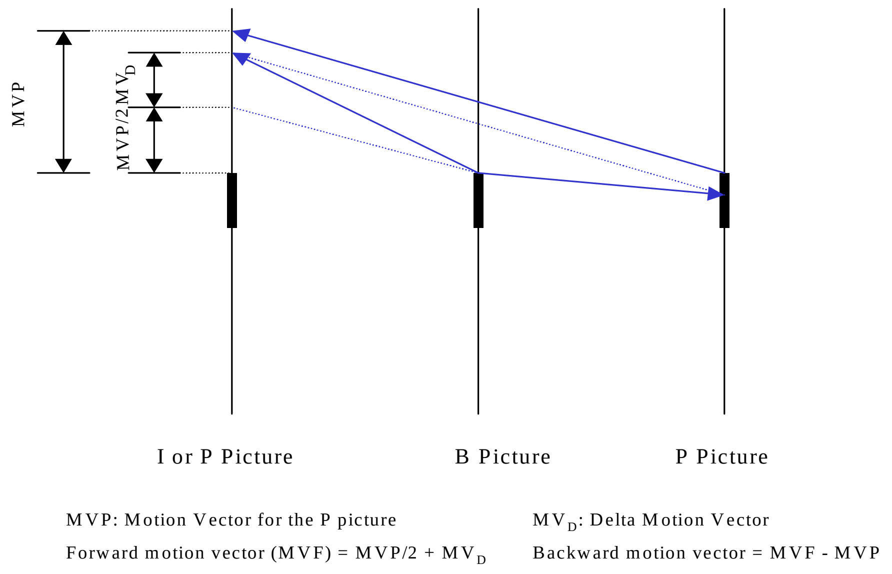

MVP
MVP/2
MV_D
I or P Picture
B Picture
P Picture
MVP: Motion Vector for the P picture
Forward motion vector (MVF) = MVP/2 + MV_D
MV_D: Delta Motion Vector
Backward motion vector = MVF - MVP

Figure 14. Example of use of Delta Motion Vectors for 1-D case

The prediction scheme for a B block is shown in Figure 15.

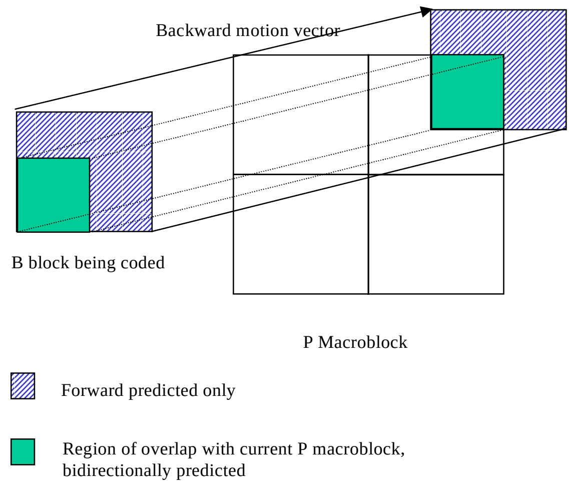


Figure 15. B block prediction

This kind of prediction scheme results in less bit rate overhead for the B picture. For relatively simple sequences at low bit rates, the picture rate can be doubled with this mode with minimal increase in the bit rate. However, for sequences with heavy motion, PB-frames do not work as well as the B picture scheme as used in MPEG 2. Also, as compared to the baseline, the use of PB-frame mode increases the end-to-end delay. So it may not be suitable for two-way interactive communication.

0.4.1.5 Advanced Intra Coding Mode

This mode attempts to improve the efficiency while coding Intra MBs in a given frame using:

a) Intra block prediction using neighboring Intra blocks.

b) A separate VLC for the Intra coefficients.

c) Modified Inverse Quantization for Intra coefficients.

An Intra coded block (the 8×8 DCT block) is predicted from an intra coded block above or an intra coded block to the left of it. There are special cases made when the neighboring blocks are either absent or not intra coded. First the DC coefficient is predicted from both the block above and the block to the left. Following this either the first row of the block may be predicted from the first row of the block above or the first column of the block may be predicted from the first column of the block to the left. It is also possible that only the DC coefficient is predicted. The AC coefficients not in the first row or first column are never predicted. This way of predicting tries to exploit the knowledge of whether the block has stronger horizontal frequency or vertical frequency components. If the block has strong horizontal frequency components, then the first row of AC coefficients is predicted from the block above and all other AC coefficients are predicted as zero. A similar thing is done with the first column of AC coefficients, when the block has stronger vertical frequency components. This is shown in Figure 16.

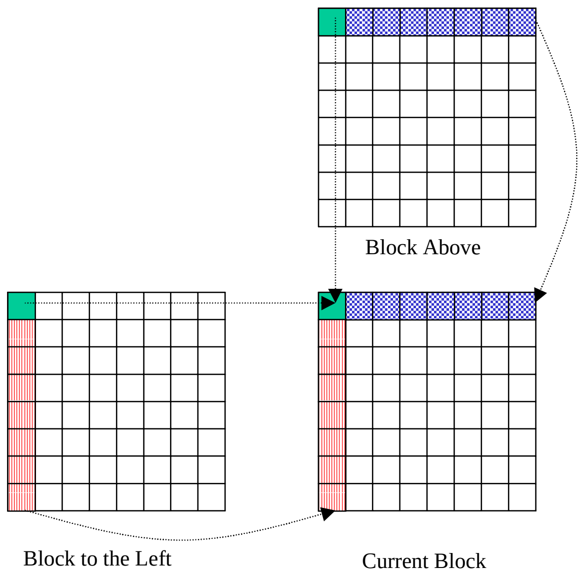


Figure 16. Neighboring blocks for prediction of DCT coefficients

The residue between the predicted coefficients and the actual coefficients is then scanned in different orders. These different scanning orders, besides the original zigzag scanning are defined to further exploit this prediction scheme. A horizontal scan is used if the block has stronger horizontal frequency components, otherwise a vertical scan is used. These scan orders are shown in Figure 17.

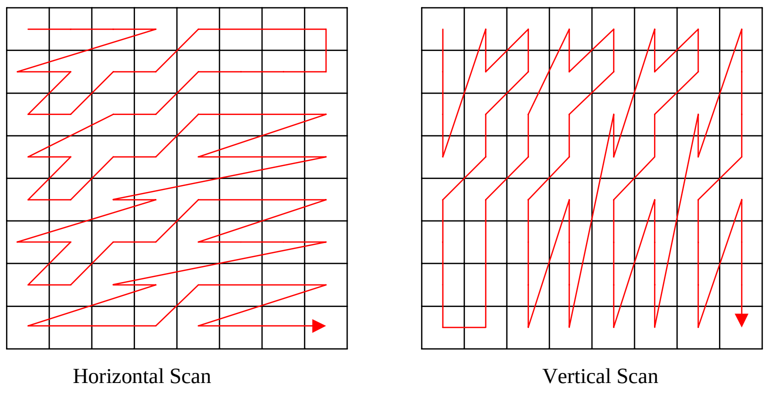


Figure 17. DCT Scanning patterns for Advanced Intra Coding

After the scanning, the array of coefficients is sent to the entropy encoder. If the prediction is good, there is a saving in the number of bits for the block.

Besides the prediction introduced above, there are additional improvements available to utilize this prediction. Firstly, an alternate intra VLC table is defined. This table has similar entries as the baseline VLC for intra coefficients, with the difference being that the run and level are interpreted differently. For instance, in this mode intra DC coefficients are not handled separately from the other coefficients any longer. As an example, a value of zero is allowed for the intra DC coefficient and this only leads to the increase in the run before the next non-zero coefficient. Secondly, this mode allows for an alternate inverse quantization procedure. This is also defined to take advantage of the fact that we are now coding a residue instead of coding actual intra coefficients.

0.4.1.6 Alternate Inter VLC Mode

During the process of inter coding it is assumed that the residues have significantly less energy than the original blocks. To take advantage of this fact, the VLC tables used to convert these residue blocks or inter blocks are different from the tables for the intra blocks. For instance, inter VLC tables use fewer bits for large runs and low levels, as these symbols occur more frequently. On the other hand intra VLC tables assign fewer bits to large levels and small runs, as these are the more frequently encountered cases. It is, however, possible that there are significant changes that occur in the scene, causing inter blocks to have energy comparable to the intra blocks. As said before, for this range of values, the intra VLC tables are more efficient. This mode allows the encoder to use the intra VLC for inter blocks so that such cases can be taken care of efficiently. The encoder is allowed to use this mode only when the decoder can deduce that these inter block coefficients were coded using the intra VLC table. The decoder can deduce this information in the following way. While using the inter VLC table to decode an inter block the decoder may encounter illegal values, such as more than 64 coefficients for the block. When this occurs, the decoder can deduce that the intra VLC table was used for the inter block and proceed to decode the block accordingly. The use of different VLC tables in different modes is highlighted in Figure 18.

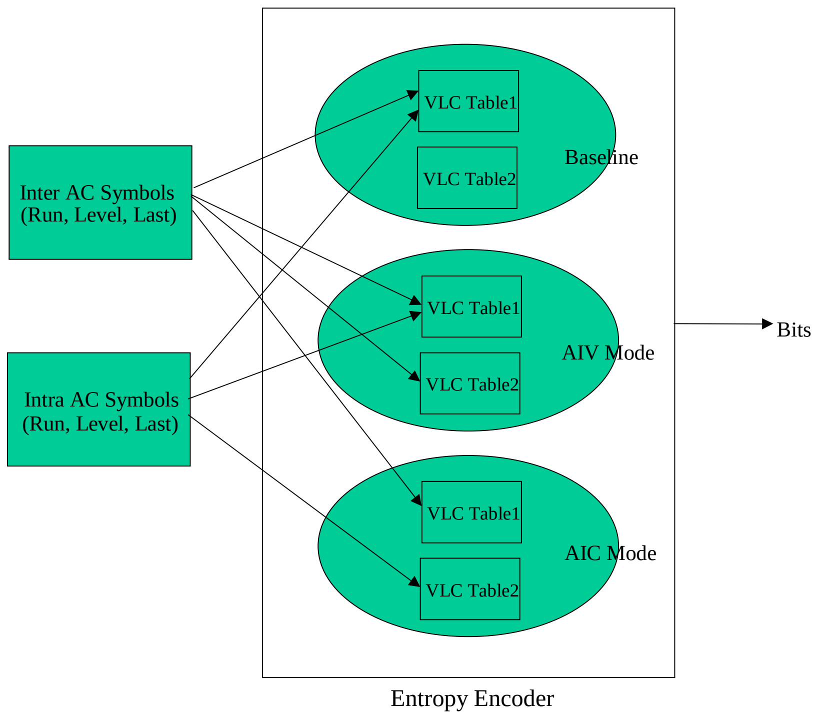


Figure 18. Use of different VLC tables for different modes

In the above figure Table 1 refers to the Inter VLC tables, while Table 2 refers to the Intra VLC tables. We can see that in the baseline mode, the AC coefficients for both the Inter and Intra coded blocks use the Table 1. The AIV mode allows the encoder to use the Intra table for the Inter AC symbols for better efficiency in cases of high motion or scene changes, while the AIC mode allows the encoder to use Inter VLC for Intra AC coefficients as these are predicted from the neighboring blocks.

0.4.1.7 Modified Quantization Mode

The quantizer operation is modified by this mode. This mode has four key features. The first feature allows the encoder a greater flexibility in controlling the quantization step size. This allows the encoder greater bit-rate controlling ability. The second feature allows the encoder to use a different quantization parameter for chrominance components and luminance components. Thus, the encoder can use a much finer step size for the chrominance components, thereby reducing any chrominance artifacts. The third feature extends the DCT coefficient range, so that any possible true coefficient can be represented. The fourth feature tries to eliminate coefficient levels that are very unlikely, thereby improving the ability to detect any errors and also to reduce the coding complexity.

0.4.1.8 Deblocking Filter Mode

The deblocking filter is an optional block edge filter that is applied to I and P frames in the coding loop. This filter is applied to 8x8 block edges to reduce blocking artifacts and improve perceptible picture quality. The selection of this mode requires the use of four motion vectors per MB and an extension of the motion vector range as in the UMV case. The effect that this filtering operation has is similar to the effect produced by the OBMC mode described in advanced prediction. Unlike the OBMC, that is done while making the prediction frame, this filtering operation is performed on the reconstructed image data. The filter operates on sets of four pixel values on a horizontal or vertical line. This is shown in the following figure, where A, B, C and D are the four pixel values with A and B belonging to one block (called block 1) and C and D belonging to block 2.

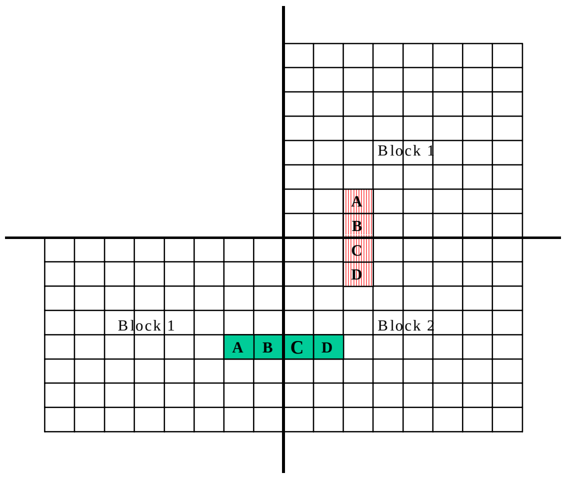


Figure 19. Examples of positions of filtered pixels

These four boundary pixels are replaced by four other values A1, B1, C1 and D1 such that the boundary between block 1 and 2 is smoothed. Simplistically, this is done by trying to move the pixel values closer together. All the horizontal filtering for a block is done before any vertical filtering. This filtering operation is limited to the picture or segment boundaries. Some special cases arise because of some MBs not being coded. On the whole, this filtering operation tries to get rid of blockiness in the frame and hence provides a better quality of decoded pictures.

0.4.1.9 Improved PB Frame Mode

The PB frames constrain the motion vectors of the B picture to be estimated from the motion vectors of the P picture part of the same frame. This kind of a scheme performs poorly for large or complex motion scenarios when the corresponding prediction, obtained for the B pictures, is not good. This improved mode allows for distinct forward and backward motion vectors. This is different from the PB frame case when the forward and backward motion vectors were both derived from the motion vector of the P picture and hence were closely related to one another. This allows a better prediction for the B pictures as in the case with the B frames in MPEG. There are three different ways of coding a MB of the B picture in the improved PB frame mode. The bi-directional prediction scheme is very similar to what occurs in the regular PB frame mode, with the delta motion vectors set to zero. The forward prediction scheme involves predicting the MB using only the previous frame as reference. The backward prediction option involves using bi-directional prediction whenever the backward motion vector points inside the already reconstructed P MB, otherwise only forward prediction is used. The improved PB frame option tries to allow a greater independence to the prediction of a B picture MB and hence allows for improved performance when there is high or complex motion in the scene.

0.4.2 Enhancements for Error Robustness

The following modes help in increasing the error resilience and are of great use if the encoder and decoder communicate across an unreliable or lossy channel.

0.4.2.1 Slice Structured Mode

This is one of the enhancements in the standard to help in improving error resilience. When this mode is turned on, the frames are subdivided into many slices, instead of the regular GOBs. A slice is a group of consecutive MBs in scanning order, with the only constraint being that the slice starts at a MB boundary and that a MB can belong to exactly one slice. There are two sub-modes in this mode. The first is the rectangular slice sub-mode in which the slice is constrained to have a rectangular shape and all slices are transmitted strictly in order. This means that if a slice has MBs that occur before (in terms of the scanning order) the MBs in another slice, then it is transmitted before the other slice. The second sub-mode is called the arbitrary slice ordering sub-mode. In this sub-mode slices can be transmitted in any order. This grouping of MBs into slices instead of GOBs allows the encoder a lot of flexibility and advantages as enumerated below.

a) It is more convenient to partition the frame into slices rather than into rows of MBs. For instance, the background may be separated from the foreground by grouping MBs appropriately. In fact different objects in the scene can be partitioned out. When in arbitrary slice ordering sub-mode, these may be transmitted separately also. This is quite useful, especially in applications when we need lip synchronization.

b) Slices also help with error resilience. When combined with the independent segment decoding mode, data dependencies are prevented from crossing slice boundaries. This can help confine errors to slices and prevent them from corrupting the entire frame.

c) Slice boundaries can also be used as resynchronization points in case of bit errors or packet losses. With the GOBs we send a header every n rows of MBs. This however, does not translate to uniform spacing of the headers in the bitstream. A uniform spacing of headers in the bitstream may be achieved by grouping the MBs into arbitrary slices. This is illustrated in Figure 20.

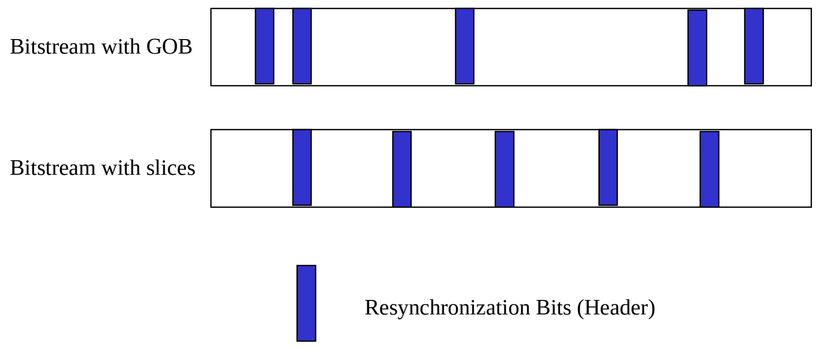


Figure 20. Uniform Resynchronization points with arbitrary slices

0.4.2.2 Reference Picture Selection Mode

This option allows for selection of any of the previously decoded frames (within 255 frames of the current frame, or when used with custom picture clock frequency, within 1023 frames of the current frame) as a reference frame to generate the prediction for the current frame. This is very different from the baseline case, when only the frame decoded immediately before the current frame may be used as a reference. This capability of selecting different frames as reference is most useful when the decoder can send information back to the encoder. So, the decoder needs to be able to inform the encoder which frames it received correctly and which frames were corrupted during the transfer. The encoder can then use this information to use only those frames as reference that the decoder has acknowledged as being received correctly. This capability of selecting different frames as reference need not be restricted to the frame level. For instance, every GOB of the current frame may be predicted from regions of different reference frames. This is shown in Figure 21.

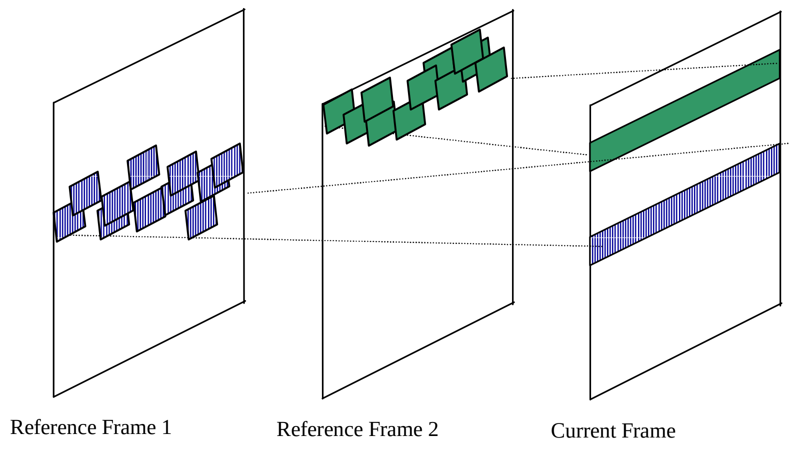


Figure 21. Different GOBs predicted from different reference frames

Thus if a GOB of the previously decoded frame is corrupted, then while predicting GOBs of the current frame, that need to use data from that particular GOB, a different frame needs to be selected. This option requires the decoder to use some backward channels in order to send information to the encoder. (There is a special mode called the Video Redundancy Coding mode when the decoder does not need to provide this feedback. In this mode the encoder encodes multiple representations of the same frame and when the decoder receives adjacent pictures with the same temporal reference, something like a frame number, it should decode the first one and discard all the rest.) The decoder also needs to initially inform the encoder about the amount of memory it has available. Based on this information the encoder decides on the maximum number of previously decoded pictures that it needs to store and how far back in time a frame can be and still be selected as reference. The information regarding which frame is selected as reference is encoded in the bitstream. There is an increase in the size of the bitstream and there is an overhead involved in the forward and backward communication between the encoder and decoder. This overhead is tolerated because it leads to a better error resilience. The encoder and decoder can work together to ensure the quality of the recovered frames, even when they communicate over error-prone networks.

0.4.2.3 Independent Segment Decoding Mode

This mode is another enhancement for improved error resilience. It tries to remove data dependencies across the video picture segment boundaries. A video picture segment may be a slice (only the rectangular slice mode is allowed with ISD) or a number of consecutive GOBs. When this mode is turned on, then the segment boundaries are treated as the picture boundaries. So, if this option is selected with the baseline options, then each picture segment is decoded independently, as if it were the whole picture. This means that no operation can reference data outside the segment. This option may also be used with options like UMV or advanced prediction in which case data outside the segment boundaries is referred to. In such cases, data outside the segment boundaries is derived through extrapolation of the segment, as was done in the case of the entire frame.

0.4.3 Scalability Related Enhancements

If a bitstream is scalable, it allows for the decoding of the video sequence at different quality levels. This is done by partitioning the pictures into several layers. Pictures are partitioned into the base layer (the lowest layer) and enhancement layers. There are three different pictures used for scalability and these are called the B, EI and EP pictures. Each picture has two numbers associated with it. The first number refers to the layer to which the picture belongs (ELNUM). The second number is called the reference layer number and this refers to the layer from which the picture was predicted (RLNUM). There are three basic methods of achieving scalability. These are temporal, signal to noise ratio (SNR) and spatial enhancements.

Temporal scalability is achieved through the use of bidirectionally predicted B pictures. These B pictures are independent entities unlike in the PB frame mode or the Improved PB frame mode, where the B pictures are syntactically intermixed with a successive P or EP picture. B pictures are not used as reference for any other pictures; thereby some of these may be dropped without adversely affecting following pictures. Figure 22 illustrates the structure of P and B pictures.

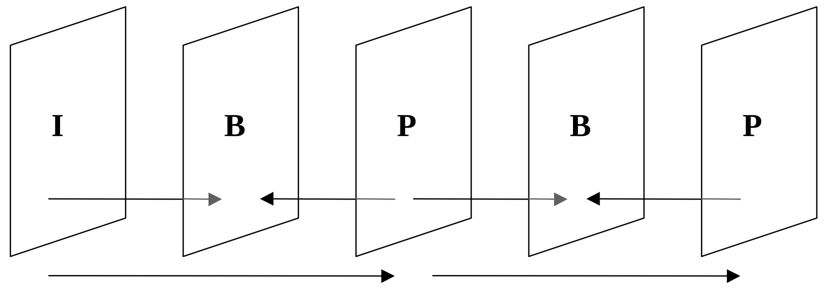


Figure 22. B picture prediction dependencies

During the process of compression, some information about the picture is lost and the encoded picture is not exactly the same as the original picture. The SNR scalability mode allows the codec to send this difference as an enhancement layer and the presence of this enhancement layer helps in increasing the SNR of the video picture. If the enhancement layer information is predicted only from the layer below, this picture is called an EI picture. The enhancement layer pictures that are predicted both from the base layer as well as the prediction from the previous enhancement layer frame are called EP pictures. This scheme is shown in Figure 23.

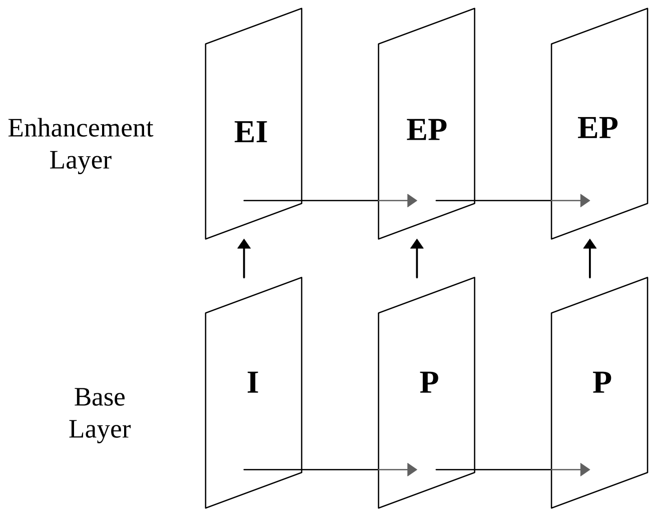


Figure 23. Illustration of EI and EP pictures

Spatial scalability is closely related to SNR scalability, with an additional operation performed before prediction of the enhancement layer pictures. The base layer pictures are interpolated by a factor two (i.e. expanded) either in one direction (X or Y) or in both. This interpolated picture is then used as reference for prediction. If the interpolation is done in both directions, then the predicted picture is called the 2-D spatial enhancement layer picture; otherwise it is called the 1-D spatial enhancement layer picture. This option hence requires the use of custom picture formats and aspect ratios. This is illustrated by the fact that when a 1-D spatial enhancement picture (in the horizontal direction) is built from a QCIF frame, it has a size 352×144, which is not a standard size. All these three methods may be used together also, i.e. the standard allows for multi-layer scalability. Scalability is very useful for better error resilience support.

0.4.4 Other Enhancement Modes

There are two other modes that are listed in the standard. These are the reference picture resampling and reduced resolution update modes.

0.4.4.1 Reference Picture Resampling

This option allows the resampling of the previously decoded reference picture to create a warped picture that can be used for predicting the future frames. This is very useful if the current picture has a different source format from the previously decoded reference picture. Resampling defines the relation between the current frame and the previously decoded reference frame. In essence, resampling specifies the alteration in shape, size and location between the current frame and the previously decoded reference frame. This resampling is defined in terms of the displacement of the four corners of the current picture area to get to the warped picture area. Four motion vectors are introduced to represent these four displacements. These motion vectors basically describe how to move the corners of the current picture area to map it to the warped picture area. The pixels are assumed to have a unit height and width and the positions of the centers of the pixels are identified. Following the position determination, horizontal and vertical displacements are computed for every pixel in the current picture area. These displacements are computed using the position of the pixel and the four conceptual motion vectors and they specify the locations of pixels in the warped area. After the location has been determined, bilinear interpolation is used to actually predict the pixel value at that pixel location. This new warped picture area is extrapolated so that the final height and width are divisible by 16. This new warped picture area is used as a reference frame for the current picture. A figure describing this is shown below.

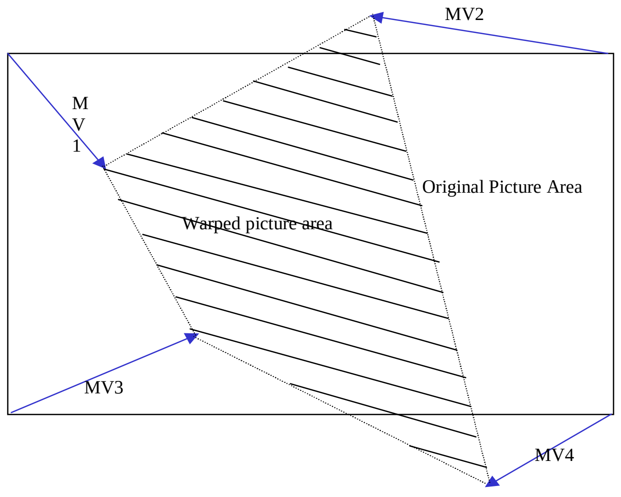
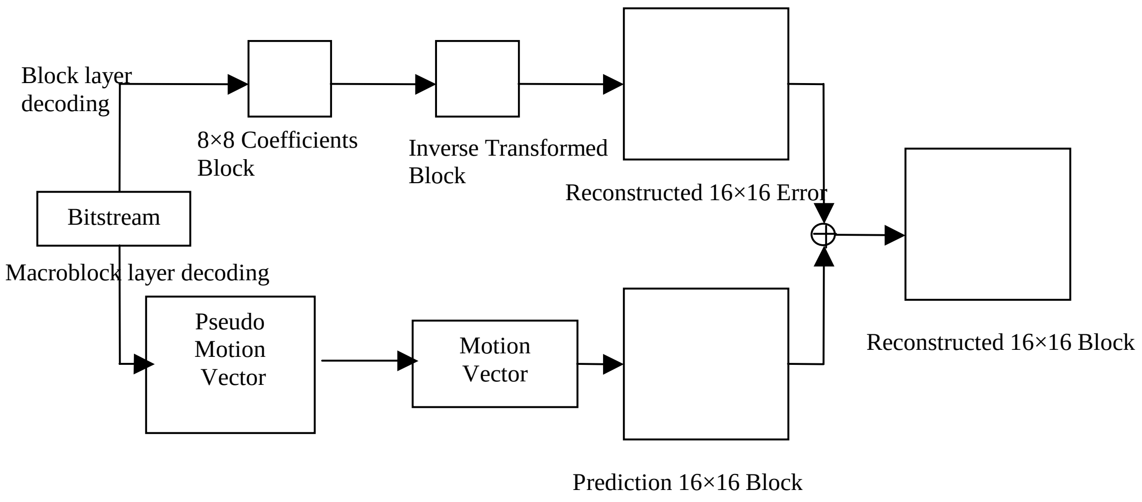


Figure 24. Conceptual Motion Vectors used for warping

This mode can be used to adaptively alter the resolution of the pictures being coded.

0.4.4.2 Reduced-Resolution Update Mode

This mode allows the encoder to send the residue or update information for a coded frame with reduced resolution, while keeping the finer detail information in the higher resolution reference image. The final frame can be reconstructed at the higher resolution from these two parts without significant loss of detail. Such an option is very useful when coding a very active or high motion scene. In this mode, MBs are assigned a size 32x32 (correspondingly blocks are 16x16) and hence per picture there are quarter the number of MBs as before. All motion vectors are estimated corresponding to these new larger MBs or blocks, depending on whether we desire one or four motion vectors for the MB. As against this, the transform coded residues for each of these 16x16 blocks are thought of as representing an 8x8 area in a reduced resolution frame. The decoder does motion compensation using the motion vectors corresponding to these large MBs and blocks.



Each of the residue blocks is upsampled to get residues for the higher resolution blocks. These are then added to produce the final high resolution block. This process is illustrated in Figure 25.

Figure 25. Reduced Resolution Update Block decoding

As the residue frame is at a lower resolution, the encoder uses a smaller number of bits while encoding it. This means that the bitstream that is sent to the decoder is now smaller. Hence, this mode can be used to increase the coding picture rate while maintaining a sufficient subjective quality.

0.4.5 Supplemental Enhancement Information Specification

The encoder sends supplemental enhancement information in a special field of the picture layer. It is not necessary for the decoder to have the ability to support these enhanced capabilities. When a decoder does not support these enhanced capabilities, it can just discard this field without affecting the quality of the decoded sequence. This mode differentiates between the decoded image and the displayed image. The displayed image is the image that is currently displayed, while the decoded image is the current image being decoded. Some of the enhanced capabilities require this distinction. For instance one of the options is the freeze picture request, in which case the displayed image is not changed to the successive decoded images, unless the option is turned off or a time out occurs. This means that a particular frame can be held on the display for a specified time although the decoder continues to decode further pictures. Similarly there are other modes that request partial picture freeze request (a particular area is frozen while the rest is updated), or resizing of a frozen partial picture area or release of the partial picture freeze mode. Besides the options to freeze parts of the image, there are other options that allow the current picture to be treated as a snapshot of the video content for external use.

There are other features among which is the Chroma Keying Information. This option allows the pixels in the decoded pictures to be transparent and semi transparent. If a pixel is defined to be transparent, it is not displayed and instead a background picture is displayed. This background picture is either a prior reference picture or an externally controlled picture. If a pixel is labeled semi-transparent, then it is displayed as a blend between the current value and the background pixel value. The last enhanced capability indicated by this mode is the extended function type. The use of this extended function is to provide a means for ITU to define a larger number of backward compatible functions later.

0.4.6 Levels of Preferred Mode Support

All the optional modes introduced in the section above are useful, however it is not necessary that all manufacturers would like to support all of them. A set of preferred mode combinations for operation is defined in the standard. Such a definition helps in identifying which combinations of modes are likely to be more widely accepted and also provides a guideline for the order in which modes should be supported in decoders. Manufacturers can thus make codecs that do not support all optional modes, but be assured of being able to communicate with several other codecs at some syntax better than baseline. These preferred mode combinations are grouped into levels based on the effect each mode has on the improvement in subjective video quality, overall delay and complexity of the codec. A decoder that supports a certain level should be able to support all modes not only in that level, but also in levels below it. Level 1, which is the lowest level, is composed of advanced intra coding mode, deblocking filter, full frame freeze and modified quantization. Level 2 includes, in addition, support for unrestricted motion vector mode, slice structured mode and reference picture resampling. Level 3, the highest level, includes supporting advanced prediction, improved PB frames mode, independent segment decoding and alternate inter VLC mode.

0.5 Conclusion

This chapter provides a basic overview of the process of video coding in general and the H.263 standard in particular. Through this chapter, we also hope we have conveyed the need for video coding standards. We also wish to state that although standards completely specify the bitstream between encoder and decoder, they allow flexibility in the way the bitstream is produced. For instance, the standard does not specify how motion estimation should be done. Encoders can be optimized in many ways, in terms of speed or quality of decoded video etc., without losing standard compliance. Similar to the standard, there are other documents drafted by ITU-T called Test Model Near-Term (TMN) [14]. These documents talk about specific encoder algorithms and how to efficiently produce standard compliant bitstreams. They are however, only guidelines, and need not be followed. There is further work being done to develop a third generation of H.263 syntax called H.263++ and H.263L. More information about the H.263 standard and the ongoing efforts is available at the web site for ITU-T http://www.itu.int and the site for the Video Coding Experts Group at ftp://standard.pictel.com/video-site.

References

[1] ITU-T SG16, Gary Sullivan, ed., “Draft text of Recommendation H.263 Version 2 (“H.263+”) for Decision,” January, 1998.

[2] ITU-T Recommendation H.261: “Video codec for audiovisual services at p x 64 kbit/s,” Geneva, 1990, revised at Helsinki, March 1993.

[3] ITU-T Recommendation H.263: “Video coding for low bitrate communication,” March 1996.

[4] ITU-R Recommendation BT.601-4: “Encoding parameters of digital; television for studios.”

[5] Ahmed, N., Natarajan, T., and Rao, K. R., “Discrete cosine transform,” *IEEE Transactions on Computers*, C-23: 90-93, 1974.

[6] Rao, K. R., and Yip, P., *Discrete Cosine Transform*, Academia Press, New York, 1990.

[7] Netravali, A. N., and Robbins, J. D., “Motion-compensated television coding: Part I,” *Bell Systems Technical Journals*, v. 58(3), pp. 631-670, March 1979.

[8] Netravali, A. N., and Haskell, B. G., *Digital Pictures*, Plenum Press, New York and London, 1995, 2nd ed.

[9] Witten, I. H., Neal, R. M., and Cleary, J.G., “Arithmetic coding for data-compression,” Communications of the ACM, v30 (6), pp. 520-540, June 1987.

[10] Orchard, M. T., and Sullivan, G. J., “Overlapped block motion compensation - an estimation-theoretic approach,” *IEEE Transactions on Image Processing*, v3 (5), pp. 693-699, Sept 1994.

[11] ITU-T Recommendation H.324: “Terminal for low bitrate multimedia communication,” March 1996.

[12] ITU-T Recommendation H.223: “Multiplexing protocol for low bitrate multimedia communication,” 1995.

[13] ITU-T Recommendation H.245: “Control protocol for multimedia communication,” 1995.

[14] “Video Codec Test Model, Near-Term, Version 10 (TMN10) Draft 1”, Document Q15-D-65 d1, April 1998.

Figure 22. B picture prediction dependencies

Figure 23. Illustration of EI and EP pictures

Figure 24. Conceptual Motion Vectors used for warping

Figure 25. Reduced Resolution Update Block decoding