An Introduction to MPEG Video Compression

John Wiseman

Introduction

MPEG video compression is used in many current and emerging products. It is at the heart of digital television set-top boxes, DSS, HDTV decoders, DVD players, video conferencing, Internet video, and other applications. These applications benefit from video compression in the fact that they may require less storage space for archived video information, less bandwidth for the transmission of the video information from one point to another, or a combination of both. Besides the fact that it works well in a wide variety of applications, a large part of its popularity is that it is defined in two finalized international standards, with a third standard currently in the definition process.

It is the purpose of this paper to introduce the reader to the basics of MPEG video compression, from both an encoding and a decoding perspective. The workings of the basic building blocks such as the discrete cosine transform and motion estimation are covered, but detailed explanations of the MPEG syntax are not. MPEG-2 is a superset of MPEG-1, but in general this paper treats the common ground of the two standards, as the differences tend to be understood better by the more advanced reader.

Example Video Compression Calculation

One of the formats defined for HDTV broadcasting within the United States is 1920 pixels horizontally by 1080 lines vertically, at 30 frames per second. If these numbers are all multiplied together, along with 8 bits for each of the three primary colors, the total data rate required would be approximately 1.5 Gb/sec. Because of the 6 MHz. channel bandwidth allocated, each channel will only support a data rate of 19.2 Mb/sec, which is further reduced to 18 Mb/sec by the fact that the channel must also support audio, transport, and ancillary data information. As can be seen, this restriction in data rate means that the original signal must be compressed by a figure of approximately 83:1. This number seems all the more impressive when it is realized that the intent is to deliver very high quality video to the end user, with as few visible artifacts as possible. This paper will show some of the basic techniques that make this video compression possible.

MPEG Video Basics

The acronym MPEG stands for Moving Picture Expert Group, which worked to generate the specifications under ISO, the International Organization for Standardization and IEC, the International Electrotechnical Commission. What is commonly referred to as "MPEG video" actually consists at the present time of two finalized standards, MPEG-1[1] and MPEG-2[2], with a third standard, MPEG-4, in the process of being finalized at the time this paper was written. The MPEG-1 & -2 standards are similar in basic concepts. They both are based on motion compensated block-based transform coding techniques, while MPEG-4 deviates from these more traditional approaches in its usage of software image construct descriptors, for target bit-rates in the very low range, < 64Kb/sec. Because MPEG-1 & -2 are finalized standards and are both presently being utilized in a large number of applications, this paper concentrates on compression techniques relating only to these two standards. Note that there is no reference to MPEG-3. This is because it was originally anticipated that this standard would refer to HDTV applications, but it was found that minor extensions to the MPEG-2 standard would suffice for this higher bit-rate, higher resolution application, so work on a separate MPEG-3 standard was abandoned.

MPEG-1 was finalized in 1991, and was originally optimized to work at video resolutions of 352x240 pixels at 30 frames/sec (NTSC based) or 352x288 pixels at 25 frames/sec (PAL based), commonly referred to as Source Input Format (SIF) video. It is often mistakenly thought that the MPEG-1 resolution is limited to the above sizes, but it in fact may go as high as 4095x4095 at 60 frames/sec. The bit-rate is optimized for applications of around 1.5 Mb/sec, but again can be used at higher rates if required. MPEG-1 is defined for progressive frames only, and has no direct provision for interlaced video applications, such as in broadcast television applications.

MPEG-2 was finalized in 1994, and addressed issues directly related to digital television broadcasting, such as the efficient coding of field-interlaced video and scalability. Also, the target bit-rate was raised to between 4 and 9 Mb/sec, resulting in potentially very high quality video. MPEG-2 consists of *profiles* and *levels*. The profile defines the bitstream scalability and the colorspace resolution, while the level defines the image resolution and the maximum bit-rate per profile. Probably the most common descriptor in use currently is Main Profile, Main Level (MP@ML) which refers to 720x480 resolution video at 30 frames/sec, at bit-rates up to 15 Mb/sec for NTSC video. Another example is the HDTV resolution of 1920x1080 pixels at 30 frame/sec, at a bit-rate of up to 80 Mb/sec. This is an example of the Main Profile, High Level (MP@HL) descriptor. A complete table of the various legal combinations can be found in reference[2].

MPEG Video Layers

MPEG video is broken up into a hierarchy of layers to help with error handling, random search and editing, and synchronization, for example with an audio bitstream. From the top level, the first layer is known as the video sequence layer, and is any self-contained bitstream, for example a coded movie or advertisement. The second layer down is the group of pictures, which is composed of 1 or more groups of intra (I) frames and/or non-intra (P and/or B) pictures that will be defined later. Of course the third layer down is the picture layer itself, and the next layer beneath it is called the slice layer. Each slice is a contiguous sequence of raster ordered macroblocks, most often on a row basis in typical video applications, but not limited to this by the specification. Each slice consists of macroblocks, which are 16x16 arrays of luminance pixels, or picture data elements, with 2 8x8 arrays of associated chrominance pixels. The macroblocks can be further divided into distinct 8x8 blocks, for further processing such as transform coding. Each of these layers has its own unique 32 bit start code defined in the syntax to consist of 23 zero bits followed by a one, then followed by 8 bits for the actual start code. These start codes may have as many zero bits as desired preceding them.

Intra Frame Coding Techniques

The term *intra coding* refers to the fact that the various lossless and lossy compression techniques are performed relative to information that is contained only within the current frame, and not relative to any other frame in the video sequence. In other words, no temporal processing is performed outside of the current picture or frame. This mode will be described first because it is simpler, and because non-intra coding techniques are extensions to these basics. Figure 1 shows a block diagram of a basic MPEG video encoder for intra frames only. It turns out that this block diagram is very similar to that of a JPEG still image video encoder, with only slight implementation detail differences. The potential ramifications of this similarity will be discussed later in this paper. The basic processing blocks shown are the video filter, discrete cosine transform, DCT coefficient quantizer, and run-length amplitude/variable length coder. These blocks are described individually in the sections below.

Video Filter

In the example HDTV data rate calculation shown previously, the pixels were represented as 8-bit values for each of the primary colors – red, green, and blue. It turns out that while this may be good for high performance computer generated graphics, it is wasteful in most video compression applications. Research into the Human Visual System (HVS) has shown that the eye is most sensitive to changes in luminance, and less sensitive to variations in chrominance. Since absolute compression is the name of the game, it makes sense that MPEG should operate on a color space that can effectively take advantage of the eye's different sensitivity to luminance and chrominance information. As such, MPEG uses the YCbCr color space to represent the data values instead of RGB, where Y is the luminance signal, Cb is the blue color difference signal, and Cr is the red color difference signal.

A macroblock can be represented in several different manners when referring to the YCbCr color space. Figure 2 shows 3 formats known as 4:4:4, 4:2:2, and 4:2:0 video. 4:4:4 is full bandwidth YCbCr video, and each macroblock consists of 4 Y blocks, 4 Cb blocks, and 4 Cr blocks. Being full bandwidth, this format contains as much information as the data would if it were in the RGB color space. 4:2:2 contains half as much chrominance information as 4:4:4, and 4:2:0 contains one quarter of the chrominance information. Although MPEG-2 has provisions to handle the higher chrominance formats for professional applications, most consumer level products will use the normal 4:2:0 mode so that is the one concentrated on in this paper.

Because of the efficient manner of luminance and chrominance representation, the 4:2:0 representation allows an immediate data reduction from 12 blocks/macroblock to 6 blocks/macroblock, or 2:1 compared to full bandwidth representations such as 4:4:4 or RGB. To generate this format without generating color aliases or artifacts requires that the chrominance signals be filtered. The pixel co-siting is as given in Figure 3, but this does not specify the actual filtering technique to be utilized. This is up to the system designer, as one of several parameters that may be optimized on a cost vs. performance basis. More details on video filtering may be found in this reference[3].

Discrete Cosine Transform

In general, neighboring pixels within an image tend to be highly correlated. As such, it is desired to use an invertible transform to concentrate randomness into fewer, decorrelated parameters. The Discrete Cosine Transform (DCT) has been shown to be near optimal for a large class of images in energy concentration and decorrelating. The DCT decomposes the signal into underlying spatial frequencies, which then allow further processing techniques to reduce the precision of the DCT coefficients consistent with the Human Visual System (HVS) model.

The DCT/IDCT transform operations are described with Equations 1 & 2 respectively[4]:

$$F(\mu, \nu) = \frac{1}{4} C(\mu)C(\nu) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x, y) \cos\left[\frac{(2x + 1)\mu\pi}{16}\right] \cos\left[\frac{(2y + 1)\nu\pi}{16}\right]$$

$$C(\mu) = \frac{1}{\sqrt{2}} \; for \; \mu = 0$$

$$C(\mu) = 1 \; for \; \mu = 1, 2, \ldots, 7$$

Equation 1: Forward Discrete Cosine Transform

$$f(x, y) = \frac{1}{4} \sum_{\mu=0}^{7} \sum_{\nu=0}^{7} C(\mu)C(\nu)F(\mu, \nu) \cos\left[\frac{(2x + 1)\mu\pi}{16}\right] \cos\left[\frac{(2y + 1)\nu\pi}{16}\right]$$

Equation 2: Inverse Discrete Cosine Transform

In Fourier analysis, a signal is decomposed into weighted sums of orthogonal sines and cosines that when added together reproduce the original signal. The 2-dimensional DCT operation for an 8x8 pixel block generates an 8x8 block of coefficients that represent a "weighting" value for each of the 64 orthogonal basis patterns that are added together to produce the original image. Figure 4 shows a grayscale plot of these DCT basis patterns, and Figure 5 shows how the vertical and horizontal frequencies are mapped into the 8x8 block pattern.

Note again that the above equations are based on data blocks of an 8x8 size. It is certainly possible to compute the DCT for other block sizes, for example 4x4 or 16x16 pixels, but the 8x8 size has become the standard as it represents an ideal compromise between adequate data decorrelation and reasonable computability. Even so, these formidable-looking equations would each normally require 1024 multiplies and 896 additions if solved directly, but fortunately, as with the case of the Fast Fourier Transform, various fast algorithms exist that make the calculations considerably faster.

Besides decorrelation of signal data, the other important property of the DCT is its efficient energy compaction. This can be shown qualitatively by looking at a simple 1-dimensional example. Figure 6 shows an *n*-point increasing ramp function, where *n* in this case equals 4. If the Discrete Fourier Transform (DFT) of this signal were to be taken, then the implied periodicity of the signal is shown as in the top portion of the figure. Quite obviously, an adequate representation of this signal with sines and cosines will require substantial high frequency components. The bottom portion of the figure shows how the DCT operation overcomes this problem, by using reflective symmetry before being periodically repeated. In this manner, the sharp time domain discontinuities are eliminated, allowing the energy to be concentrated more towards the lower end of the frequency spectrum. This example also illustrates an interesting fact, that the DCT of the *n*-point signal may be calculated by performing a 2*n*-point DFT[5].

To further demonstrate the effective energy concentration property of the DCT operation, a series of figures are given showing a deletion of a number of DCT coefficients. Figure 7 shows an 8-bit monochrome image, where an 8x8 DCT operation has been performed on all the blocks of the image, all of the coefficients are retained, then an 8x8 IDCT is performed to reconstruct the image. Figure 8 is the same image with only the 10 DCT coefficients in the upper left-hand corner retained. The remaining 54 higher frequency DCT coefficients have all been set to zero. When the IDCT operation is applied and the image reconstructed, it is shown that the image still retains a fairly high degree of quality compared to the original image that was reconstructed using all 64 DCT coefficients. Figure 9 eliminates another diagonal row of DCT coefficients such that only 6 are kept and used in the IDCT operation. Again, some degradation is apparent, but overall the picture quality is still fair. Figure 10 continues by eliminating another row, resulting in only 3 coefficients saved. At this point, fairly significant blockiness is observed, especially around sharp edges within the image. Figure 11 illustrates the extreme case where only the DC coefficient (extreme upper left-hand corner) is kept. Although dramatic blockiness is apparent, the image is still surprisingly recognizable when it is realized that only 1 out of the original 64 coefficients have been maintained.

Figures 12-14 show the above process in a slightly different light. These three figures clearly show the amount of energy that is missing when the higher frequency coefficients are deleted. It is also apparent that this energy is concentrated in areas of the image that are associated with edges, or high spatial frequencies. Because of this, it is desired that the total number and the degree of DCT coefficient deletion be controlled on a macroblock basis. This control is accomplished with a process called quantization.

DCT Coefficient Quantization

As was shown previously in Figure 5, the lower frequency DCT coefficients toward the upper left-hand corner of the coefficient matrix correspond to smoother spatial contours, while the DC coefficient corresponds to a solid luminance or color value for the entire block. Also, the higher frequency DCT coefficients toward the lower right-hand corner of the coefficient matrix correspond to finer spatial patterns, or even noise within the image. Since it is well known that the HVS is less sensitive to errors in high frequency coefficients than it is for lower frequencies, it is desired that the higher frequencies be more coarsely quantized in their representation.

The process of DCT coefficient quantization is described as follows. Each 12-bit coefficient is divided by a corresponding quantization matrix value that is supplied from an intra quantization matrix. The default matrix is given in Figure 15, and if the encoder decides it is warranted, it may substitute a new quantization matrix at a picture level and download it to the decoder via the bitstream. Each value in this matrix is pre-scaled by multiplying by a single value, known as the quantizer scale code. This value may range in value from 1-112, and is modifiable on a macroblock basis, making it useful as a fine-tuning parameter for the bit-rate control, since it would not be economical to send an entirely new matrix on a macroblock basis. The goal of this operation is to force as many of the DCT coefficients to zero, or near zero, as possible within the boundaries of the prescribed bit-rate and video quality parameters.

Run-Length Amplitude/Variable Length Coding

An example of a typical quantized DCT coefficient matrix is given in Figure 16. As desired, most of the energy is concentrated within the lower frequency portion of the matrix, and most of the higher frequency coefficients have been quantized to zero. Considerable savings can be had by representing the fairly large number of zero coefficients in a more effective manner, and that is the purpose of run-length amplitude coding of the quantized coefficients. But before that process is performed, more efficiency can be gained by reordering the DCT coefficients.

Since most of the non-zero DCT coefficients will typically be concentrated in the upper left-hand corner of the matrix, it is apparent that a zigzag scanning pattern will tend to maximize the probability of achieving long runs of consecutive zero coefficients. This zigzag scanning pattern is shown in the upper portion of Figure 17. Note for the sake of completeness that a second, alternate scanning pattern defined in MPEG-2 is shown in the lower portion of the figure. This scanning pattern may be chosen by the encoder on a frame basis, and has been shown to be effective on interlaced video images. This paper will concentrate only on usage of the standard zigzag pattern, however.

Again, the block of quantized DCT coefficients as presented in Figure 16 is referenced. Scanning of the example coefficients in a zigzag pattern results in a sequence of numbers as follows: 8, 4, 4, 2, 2, 2, 1, 1, 1, 1, (12 zeroes), 1, (41 zeroes). This sequence is then represented as a run-length (representing the number of consecutive zeroes) and an amplitude (coefficient value following a run of zeroes). These values are then looked up in a fixed table of variable length codes[6], where the most probable occurrence is given a relatively short code, and the least probable occurrence is given a relatively long code. In this example, this becomes:

Zero Run-Length	**Amplitude**	**MPEG Code Value**
N/A	8 (DC Value)	110 1000
0	4	0000 1100
0	4	0000 1100
0	2	0100 0
0	2	0100 0
0	2	0100 0
0	1	110
0	1	110
0	1	110
0	1	110
12	1	0010 0010 0
EOB	EOB	10

Note that the first run of 12 zeroes has been very efficiently represented with only 9 bits, and the last run of 43 zeroes has been entirely eliminated, represented only with a 2-bit End Of Block (EOB) indicator. It can be seen from the table that the quantized DCT coefficients are now represented by a sequence of 61 binary bits. Considering that the original 8x8 block of 8-bit pixels required 512 bits for full representation, this is a compression of approximately 8.4:1 at this point.

Certain coefficient values that are not particularly likely to occur are coded with escape sequences to prevent the code tables from becoming too long. As an example, consider what would happen if the last isolated coefficient value of 1 was instead a value of 3. There is no code value for a run-length of 12 followed by an amplitude of 3, so it is instead coded with the escape sequence 0000 01, a 6-bit representation of the run-length (12 = 001100), and finally a 12-bit representation of the amplitude (3 = 000000000011). All of the other values in the table remain the same as before. In this case, the total number of bits will grow to 76, and the compression is lowered to approximately 6.7:1.

Video Buffer and Rate Control

Most of the applications that were mentioned in the introduction use a fixed bit-rate for the transmission of the compressed information. For the case of HDTV broadcasts, this fixed rate will be 18 Mb/sec for the video signal. Unfortunately, the individual video images to be coded may contain drastically varying amounts of information, resulting in wildly varying coding efficiencies from picture to picture. This may also occur within a given picture, as portions of the picture may be very smooth, yet other areas may contain large amounts of high frequency information. Because of these variations, it is necessary to buffer the encoded bitstream before it is transmitted. Due to the fact that the buffer must necessarily by limited in size (physical limitations and delay constraints), a feedback system must be used as a *rate control* mechanism to prevent underflow or overflow within the buffer. The buffer and rate controller are necessary for intra frame only coding/decoding systems, but become even more important for non-intra coded systems as the coding efficiency changes relative to the type of frame coding utilized, and there can be drastic differences in the total number of bits that ultimately are used to represent the original I, P, and B frames.

By looking at Figure 1, it can be seen that the only block available for the rate control mechanism to reasonably modify is the DCT coefficient quantizer. Because the quantizer matrix may be changed on a picture basis and the quantizer scale may be changed on a macroblock basis, these parameters are commonly used by encoder rate control algorithms to provide dynamic control over the relative buffer fullness. In this manner, a constant bit-rate may be provided by the output of the encoder buffer, yet underflow or overflow may be prevented without severe quality penalties such as the repeating or dropping of entire video frames. It should be noted that although rate control algorithms are necessary in fixed bit-rate applications, neither the MPEG-1 nor the MPEG-2 standard define particular implementations. Since these algorithms have direct bearing on the ultimate video presentation quality, most of them are encoder vendor proprietary, and the subject of current research. A list of some of the more well-known general algorithms may be found in this reference[3].

Non-Intra Frame Coding Techniques

The previously discussed intra frame coding techniques were limited to processing the video signal on a *spatial* basis, relative only to information within the current video frame. Considerably more compression efficiency can be obtained however, if the inherent *temporal*, or time-based redundancies, are exploited as well. Anyone who has ever taken a reel of the old-style super-8 movie film and held it up to a light can certainly remember seeing that most consecutive frames within a sequence are very similar to the frames both before and after the frame of interest. Temporal processing to exploit this redundancy uses a technique known as *block-based motion compensated prediction, using motion estimation*. A block diagram of the basic encoder with extensions for non-intra frame coding techniques is given in Figure 18. Of course, this encoder can also support intra frame coding as a subset.

P Frames

Starting with an intra, or I frame, the encoder can forward predict a future frame. This is commonly referred to as a P frame, and it may also be predicted from other P frames, although only in a forward time manner. As an example, consider a group of pictures that lasts for 6 frames. In this case, the frame ordering is given as I,P,P,P,P,P,I,P,P,P,P,…

Each P frame in this sequence is predicted from the frame immediately preceding it, whether it is an I frame or a P frame. As a reminder, I frames are coded spatially with no reference to any other frame in the sequence.

B Frames

The encoder also has the option of using forward/backward interpolated prediction. These frames are commonly referred to as *bi-directional* interpolated prediction frames, or B frames for short. As an example of the usage of I, P, and B frames, consider a group of pictures that lasts for 6 frames, and is given as I,B,P,B,P,B,I,B,P,B,P,B,… As in the previous I & P only example, I frames are coded spatially only and the P frames are forward predicted based on previous I and P frames. The B frames however, are coded based on a forward prediction from a previous I or P frame, as well as a backward prediction from a succeeding I or P frame. As such, the example sequence is processed by the encoder such that the first B frame is predicted from the first I frame and first P frame, the second

B frame is predicted from the second and third P frames, and the third B frame is predicted from the third P frame and the first I frame of the next group of pictures. From this example, it can be seen that backward prediction requires that the future frames that are to be used for backward prediction be encoded and transmitted first, out of order. This process is summarized in Figure 19. There is no defined limit to the number of consecutive B frames that may be used in a group of pictures, and of course the optimal number is application dependent. Most broadcast quality applications however, have tended to use 2 consecutive B frames (I,B,B,P,B,B,P,…) as the ideal trade-off between compression efficiency and video quality.

The main advantage of the usage of B frames is coding efficiency. In most cases, B frames will result in less bits being coded overall. Quality can also be improved in the case of moving objects that reveal hidden areas within a video sequence. Backward prediction in this case allows the encoder to make more intelligent decisions on how to encode the video within these areas. Also, since B frames are not used to predict future frames, errors generated will not be propagated further within the sequence.

One disadvantage is that the frame reconstruction memory buffers within the encoder and decoder must be doubled in size to accommodate the 2 anchor frames. This is almost never an issue for the relatively expensive encoder, and in these days of inexpensive DRAM it has become much less of an issue for the decoder as well. Another disadvantage is that there will necessarily be a delay throughout the system as the frames are delivered out of order as was shown in Figure 19. Most one-way systems can tolerate these delays, as they are more objectionable in applications such as video conferencing systems.

Motion Estimation

The temporal prediction technique used in MPEG video is based on motion estimation. The basic premise of motion estimation is that in most cases, consecutive video frames will be similar except for changes induced by objects moving within the frames. In the trivial case of zero motion between frames (and no other differences caused by noise, etc.), it is easy for the encoder to efficiently predict the current frame as a duplicate of the prediction frame. When this is done, the only information necessary to transmit to the decoder becomes the syntactic overhead necessary to reconstruct the picture from the original reference frame. When there is motion in the images, the situation is not as simple.

Figure 20 shows an example of a frame with 2 stick figures and a tree. The second half of this figure is an example of a possible next frame, where panning has resulted in the tree moving down and to the right, and the figures have moved farther to the right because of their own movement outside of the panning. The problem for motion estimation to solve is how to adequately represent the changes, or differences, between these two video frames.

The way that motion estimation goes about solving this problem is that a comprehensive 2-dimensional spatial search is performed for each *luminance* macroblock. Motion estimation is not applied directly to chrominance in MPEG video, as it is assumed that the color motion can be adequately represented with the same motion information as the luminance. It should be noted at this point that MPEG does not define how this search should be performed. This is a detail that the system designer can choose to implement in one of many possible ways. This is similar to the bit-rate control algorithms discussed previously, in the respect that complexity vs. quality issues need to be addressed relative to the individual application. It is well known that a full, exhaustive search over a wide 2-dimensional area yields the best matching results in most cases, but this performance comes at an extreme computational cost to the encoder. As motion estimation usually is the most computationally expensive portion of the video encoder, some lower cost encoders might choose to limit the pixel search range, or use other techniques such as telescopic searches, usually at some cost to the video quality.

Figure 21 shows an example of a particular macroblock from Frame 2 of Figure 20, relative to various macroblocks of Frame 1. As can be seen, the top frame has a bad match with the macroblock to be coded. The middle frame has a fair match, as there is some commonality between the 2 macroblocks. The bottom frame has the best match, with only a slight error between the 2 macroblocks. Because a relatively good match has been found, the encoder assigns *motion vectors* to the macroblock, which indicate how far horizontally and vertically the macroblock must be moved so that a match is made. As such, each forward and backward predicted macroblock may contain 2 motion vectors, so true bidirectionally predicted macroblocks will utilize 4 motion vectors.

Figure 22 shows how a potential predicted Frame 2 can be generated from Frame 1 by using motion estimation. In this figure, the predicted frame is subtracted from the desired frame, leaving a (hopefully) less complicated residual error frame that can then be encoded much more efficiently than before motion estimation. It can be seen that the more accurate the motion is estimated and matched, the more likely it will be that the residual error will approach zero, and the coding efficiency will be highest. Further coding efficiency is accomplished by taking advantage of the fact that motion vectors tend to be highly correlated between macroblocks. Because of this, the horizontal component is compared to the previously valid horizontal motion vector and only the difference is coded. This same difference is calculated for the vertical component before coding. These difference codes are then described with a variable length code for maximum compression efficiency.

Of course not every macroblock search will result in an acceptable match. If the encoder decides that no acceptable match exists (again, the "acceptable" criterion is not MPEG defined, and is up to the system designer) then it has the option of coding that particular macroblock as an intra macroblock, even though it may be in a P or B frame. In this manner, high quality video is maintained at a slight cost to coding efficiency.

Coding of Residual Errors

After a predicted frame is subtracted from its reference and the residual error frame is generated, this information is spatially coded as in I frames, by coding 8x8 blocks with the DCT, DCT coefficient quantization, run-length/amplitude coding, and bitstream buffering with rate control feedback. This process is basically the same with some minor differences, the main ones being in the DCT coefficient quantization. The default quantization matrix for non-intra frames is a flat matrix with a constant value of 16 for each of the 64 locations. This is very different from that of the default intra quantization matrix (Figure 15) which is tailored for more quantization in direct proportion to higher spatial frequency content. As in the intra case, the encoder may choose to override this default, and utilize another matrix of choice during the encoding process, and download it via the encoded bitstream to the decoder on a picture basis. Also, the non-intra quantization step function contains a *dead-zone* around zero that is not present in the intra version. This helps eliminate any lone DCT coefficient quantization values that might reduce the run-length amplitude efficiency. Finally, the motion vectors for the residual block information are calculated as differential values and are coded with a variable length code according to their statistical likelihood of occurrence.

Intra Frame Decoding

To decode a bitstream generated from the encoder of Figure 1, it is necessary to reverse the order of the encoder processing. In this manner, an I frame decoder consists of an input bitstream buffer, a Variable Length Decoder (VLD), an inverse quantizer, an Inverse Discrete Cosine Transform (IDCT), and an output interface to the required environment (computer hard drive, video frame buffer, etc.). This decoder is shown in Figure 23.

The input bitstream buffer consists of memory that operates in the inverse fashion of the buffer in the encoder. For fixed bit-rate applications, the constant rate bitstream is buffered in the memory and read out at a variable rate depending on the coding efficiency of the macroblocks and frames to be decoded.

The VLD is probably the most computationally expensive portion of the decoder because it must operate on a bit-wise basis (VLD decoders need to look at *every* bit, because the boundaries between variable length codes are random and non-aligned) with table look-ups performed at speeds up to the input bit-rate. This is generally the only function in the receiver that is *more* complex to implement than its corresponding function within the encoder, because of the extensive high-speed bit-wise processing necessary.

The inverse quantizer block multiplies the decoded coefficients by the corresponding values of the quantization matrix and the quantization scale factor. Clipping of the resulting coefficients is performed to the region –2048 to +2047, then an IDCT mismatch control is applied to prevent long term error propagation within the sequence.

The IDCT operation is given in Equation 2, and is seen to be similar to the DCT operation of Equation 1. As such, these two operations are very similar in implementation between encoder and decoder.

Non-Intra Decoding

It was shown previously that the non-intra frame encoder built upon the basic building blocks of the intra frame encoder, with the addition of motion estimation and its associated support structures. This is also true of the non-intra frame decoder, as it contains the same core structure as the intra frame decoder with the addition of motion compensation support. Again, support for intra frame decoding is inherent in the structure, so I, P, and B frame decoding is possible. The decoder is shown in Figure 24.

Implementation Issues

It is all very fine to have standards that define video compression techniques, but general acceptance will never come if those standards cannot be reasonably implemented. Of course the expression "reasonably implemented" will be highly application dependent. Take for example the case of a typical digital television set-top box. These devices are extremely cost sensitive, with target prices of approximately 200-300 dollars for a self-contained unit including chassis and power supply. As such, the typical design consists of a custom VLSI chip dedicated to video compression, and supported by external DRAM and a very low price microprocessor. Cost is further reduced in systems such as these by incorporating the audio decoder and possibly the transport demultiplexer into the same VLSI device as the video decoder. Devices are available now that perform these functions while interfacing with 2MB of standard DRAM via a 64 bit data bus (CCIR-601 resolution video decoding). Several manufacturers have plans to further reduce cost by lowering the data bus interface to 32 bits by using faster SDRAM chips for external storage. Also, it was pointed out previously that the basic techniques used in MPEG video are similar to those used in the JPEG still image compression standard. Because of this, it is fairly easy to design a multi-purpose decoder device to incorporate extra functionality such as JPEG, or even H.261 video conferencing support if the end application warrants the inclusions. High-end applications such as HDTV decoding chipsets (another cost sensitive consumer area) should also be supported by several manufacturers by the end of 1998. An interesting summary of typical hardware requirements is given in the following table, taken from the MPEG FAQ (located at http://www.crs4.it/);

video profile	**typical decoder transistor count**	**total dram**	**DRam bus width, speed**
MPEG-1 CPB	0.4-0.75 million	4Mb	16 bits, 80 ns
MPEG-1 601	0.8-1.1 million	16Mb	64 bits, 80 ns
MPEG-2 MP@ML	0.9-1.5 million	16 Mb	64 bits, 80 ns
MPEG-2 MP@HL	2.0-3.0 million	64 Mb	N/A

Another possible need for MPEG video decoding, but somewhat orthogonal to the above, is with web browsing in the PC environment. In this application, the user may desire to decode the video bitstream without having to purchase a separate card containing extra hardware. Here it makes sense to utilize a software decoder running entirely on the host CPU. There are presently many suppliers of PC software to decode and display MPEG video, and their performance is dependent on the source bitstream, video resolution, and CPU speed.

New PC and workstation applications are emerging that severely tax the CPU, and that cannot be successfully implemented by merely using higher clock speed CPU chips. In these cases, CPU designers have taken a careful look at what "multimedia" applications such as MPEG video require, and have started to include instructions and architectural features within the CPU to facilitate these demanding requirements. An in-depth overview of video compression's influence on CPU design is available in this reference[7], and it includes details of Intel's MMX, Sun's Visual Instruction Set, and Hewlett-Packard's MAX-2 subword parallelism architectures, and how they can be used to improve the performance of applications such as MPEG video.

Summary

This tutorial paper is just an introduction to some of the various components of MPEG video compression. Textbooks have been written about individual techniques such as the discrete cosine transform, and this and other components are the subjects of current academic and industry research. The reader is encouraged to search out more in-depth information on these topics, MPEG syntax, applications, etc., which can be found in the references listed below.

Acknowledgments

Thanks to Larry Pearlstein and Sharif Sazzad of Hitachi America, Ltd. DMSL for the usage of the motion estimation figures and DCT coefficient viewing software, respectively.

References

[1]"Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s," *ISO/IEC 11172-2: Video* (November 1991).

[2]"Generic Coding of Moving Pictures and Associated Audio Information: Video," ISO/IEC 13818-2 : Draft International Standard (November 1994).

[3]Barry G. Haskell, Atul Puri, Arun N. Netravali, *Digital Video: An Introduction to MPEG-2*, Chapman and Hall, 1997.

[4]K.R. Rao, P. Yip, *Discrete Cosine Transform – Algorithms, Advantages, Applications*, Academic Press, Inc., 1990.

[5]Majid Rabbani, Paul W. Jones, *Digital Image Compression Techniques*, SPIE Optical Engineering Press, 1991.

[6]Joan L. Mitchell, William B. Pennebaker, Chad E. Fogg, Didier J. LeGall, *MPEG Video Compression Standard*, Chapman and Hall, 1997.

[7]*IEEE Micro Magazine – Media Processing*, IEEE Computer Society, Volume 16 Number 4, August 1996.

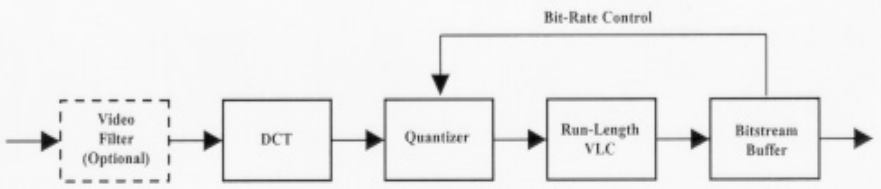

Figure 1: Intra Frame Encoder

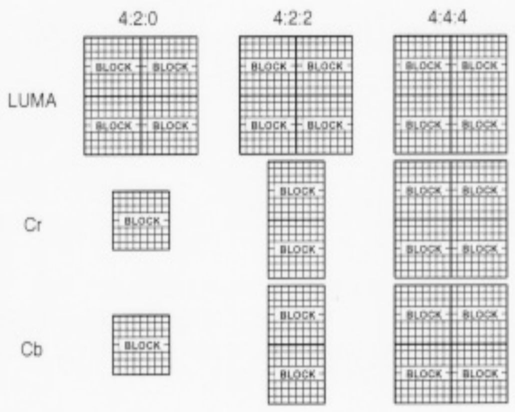

Figure 2: Video Formats

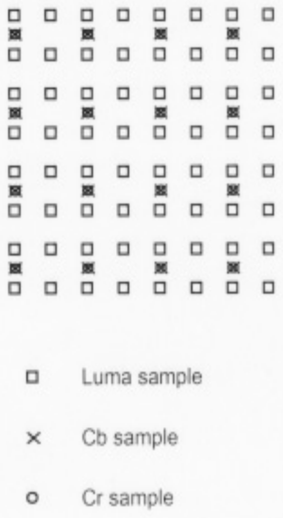

Figure 3: Pixel Format

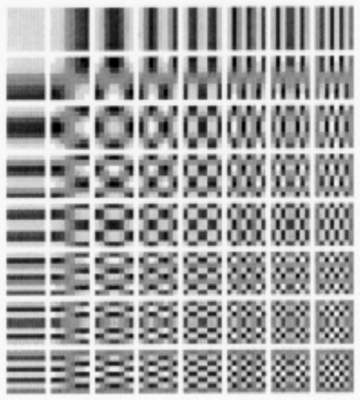

Figure 4: DCT Basis Patterns

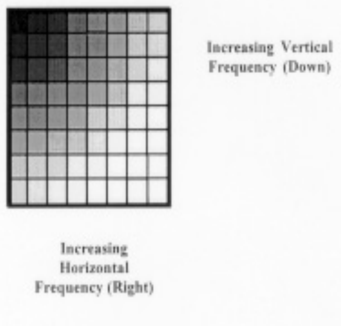
Increasing Vertical Frequency (Down)
Increasing Horizontal Frequency (Right)

Figure 5: DCT Frequency Mapping

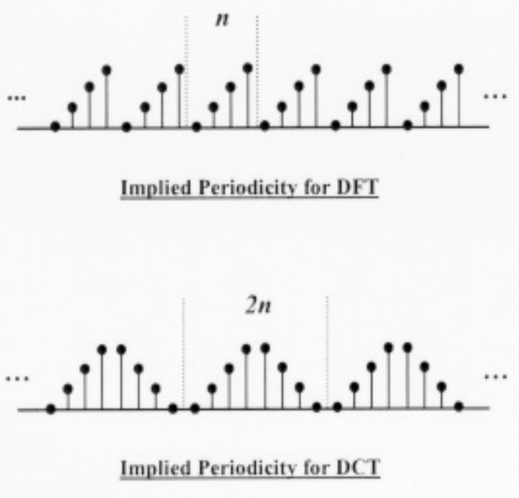

Figure 6: DCT Symmetry

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Figure 15: Default Intra Quant Matrix

8	4	2	1	0	0	0	0
4	2	1	0	0	0	0	0
2	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 16: Example Coefficients

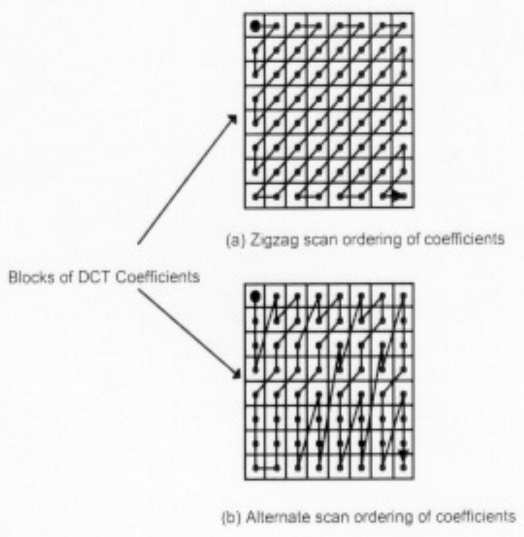
(a) Zigzag scan ordering of coefficients
Blocks of DCT Coefficients
(b) Alternate scan ordering of coefficients

Figure 17: DCT Coefficient Scanning

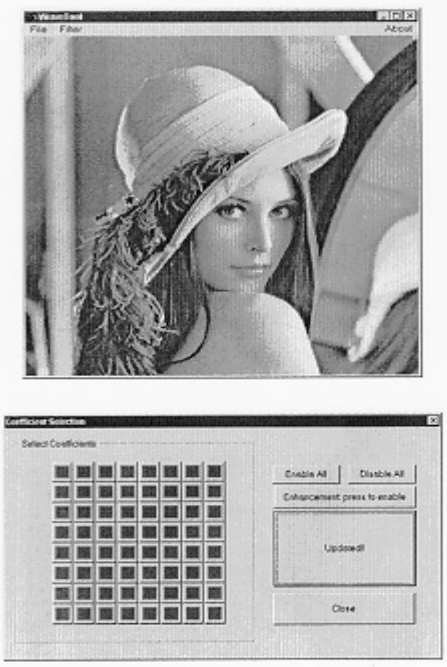

Figure 7

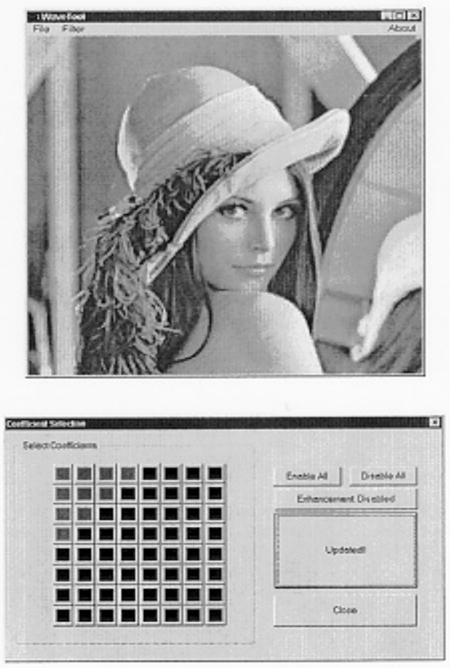

Figure 8

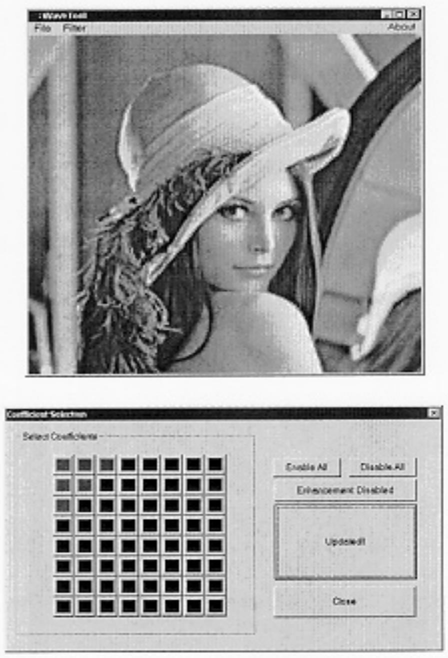

Figure 9

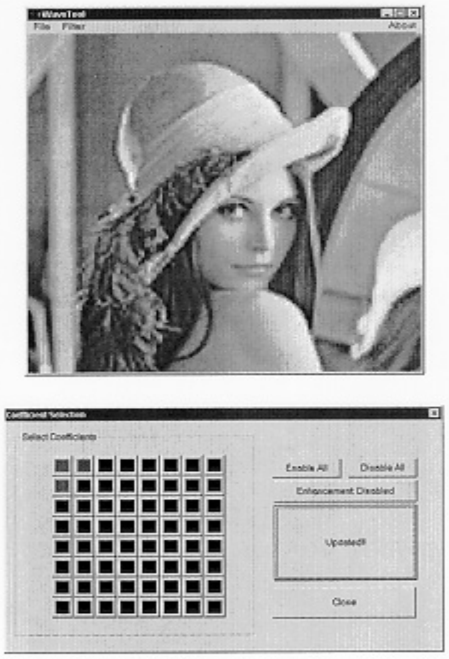

Figure 10

Note: It is hard to see in these scanned images, but the "Select Coefficients" box (the 8x8 grid under each image) contains gray boxes (coefficient on) and black boxes (coefficient off).

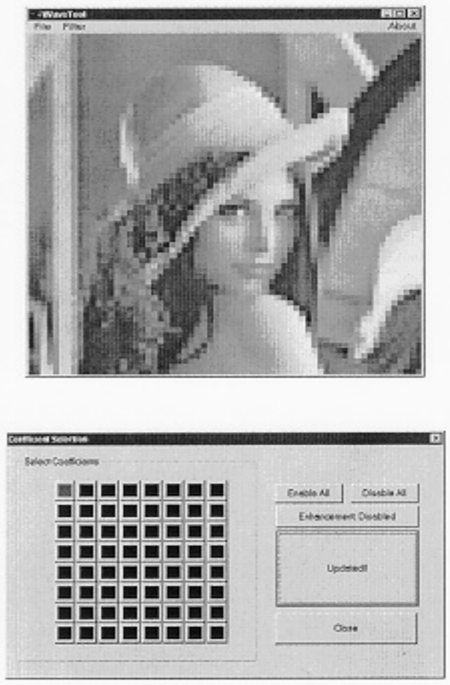

Figure 11

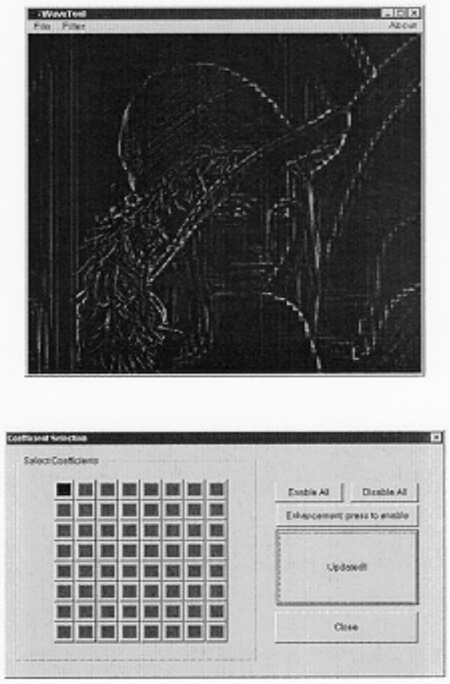

Figure 12

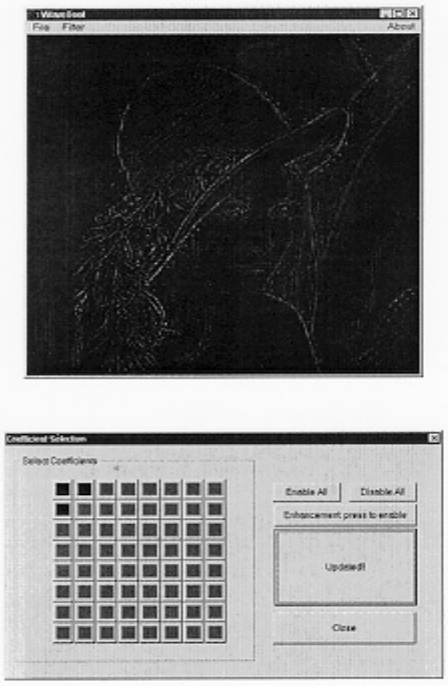

Figure 13

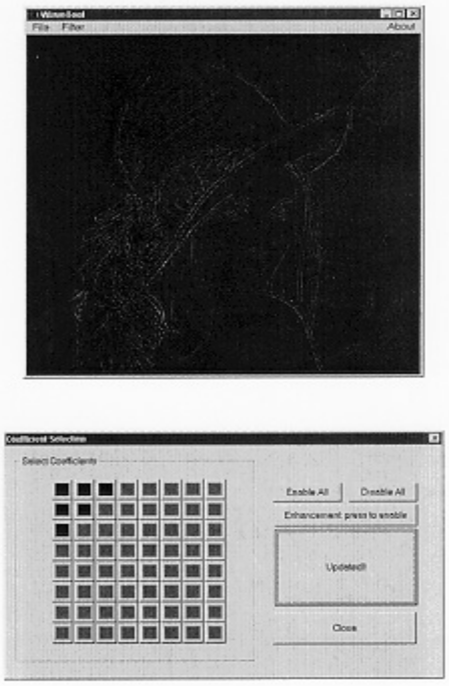

Figure 14

Note: It is hard to see in these scanned images, but the "Select Coefficients" box (the 8x8 grid under each image) contains gray boxes (coefficient on) and black boxes (coefficient off).

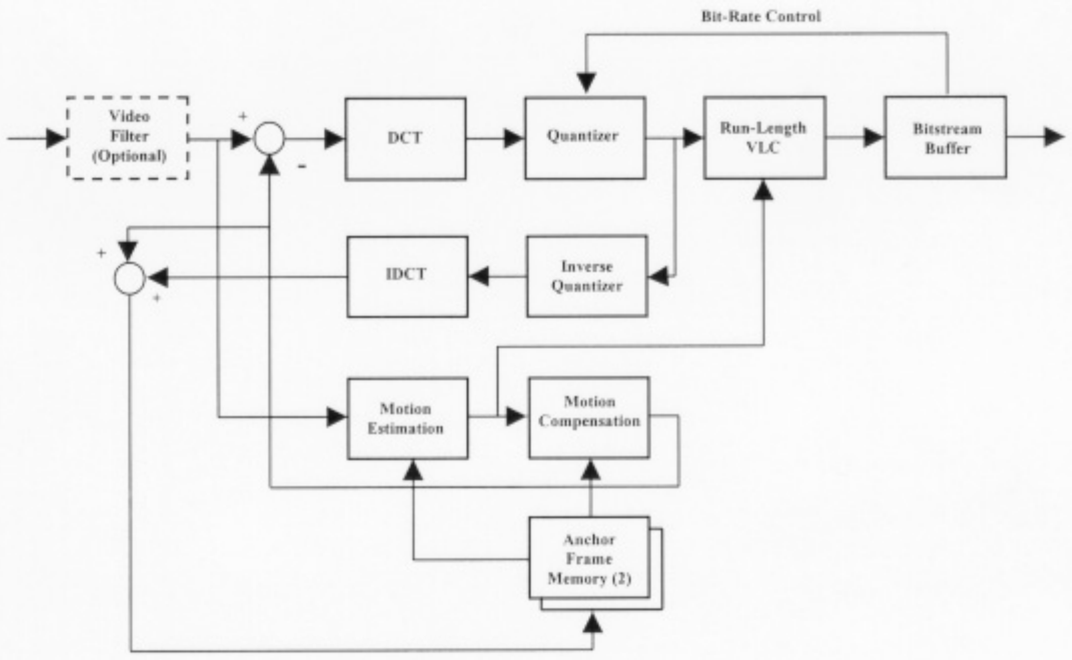

Figure 18: Non-Intra Frame Encoder

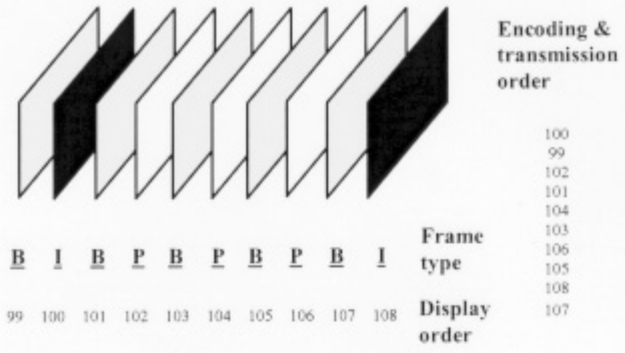

Figure 19: Frame Ordering

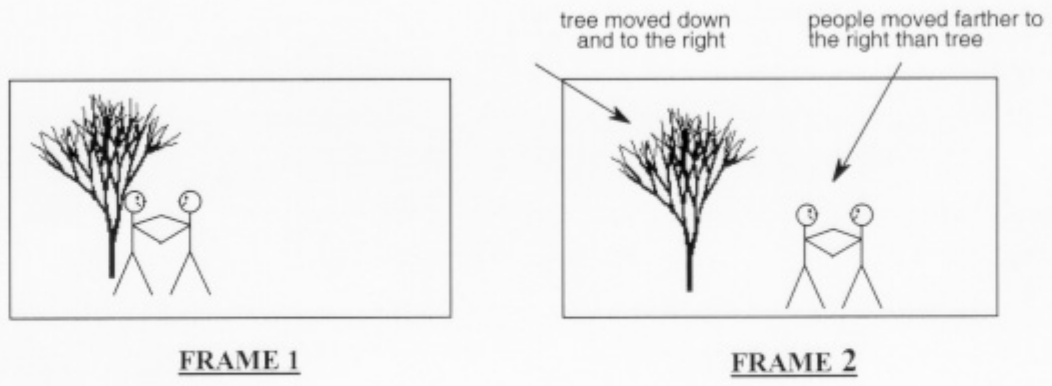
tree moved down and to the right
people moved farther to the right than tree
FRAME 1
FRAME 2

Figure 20

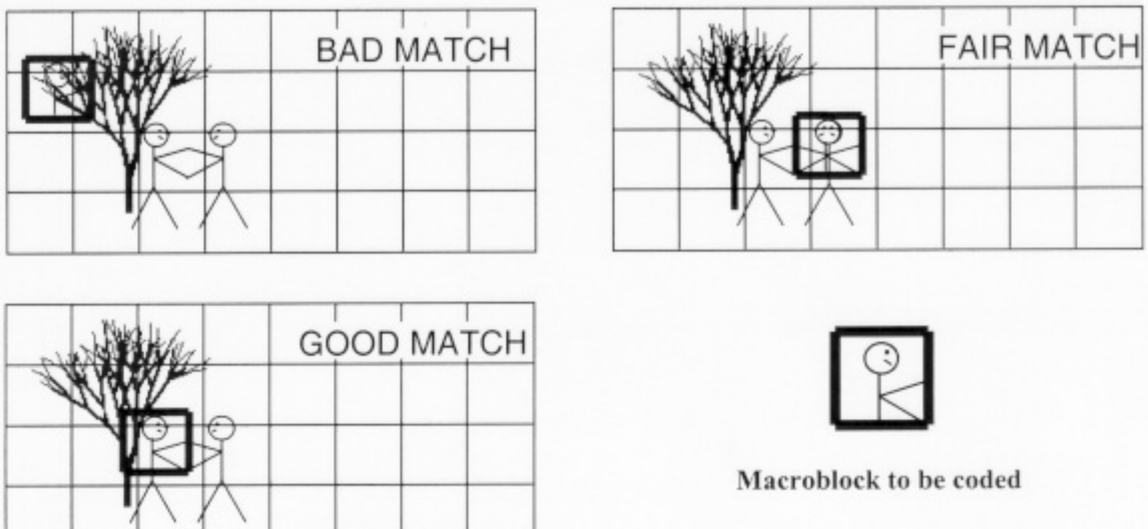

Figure 21

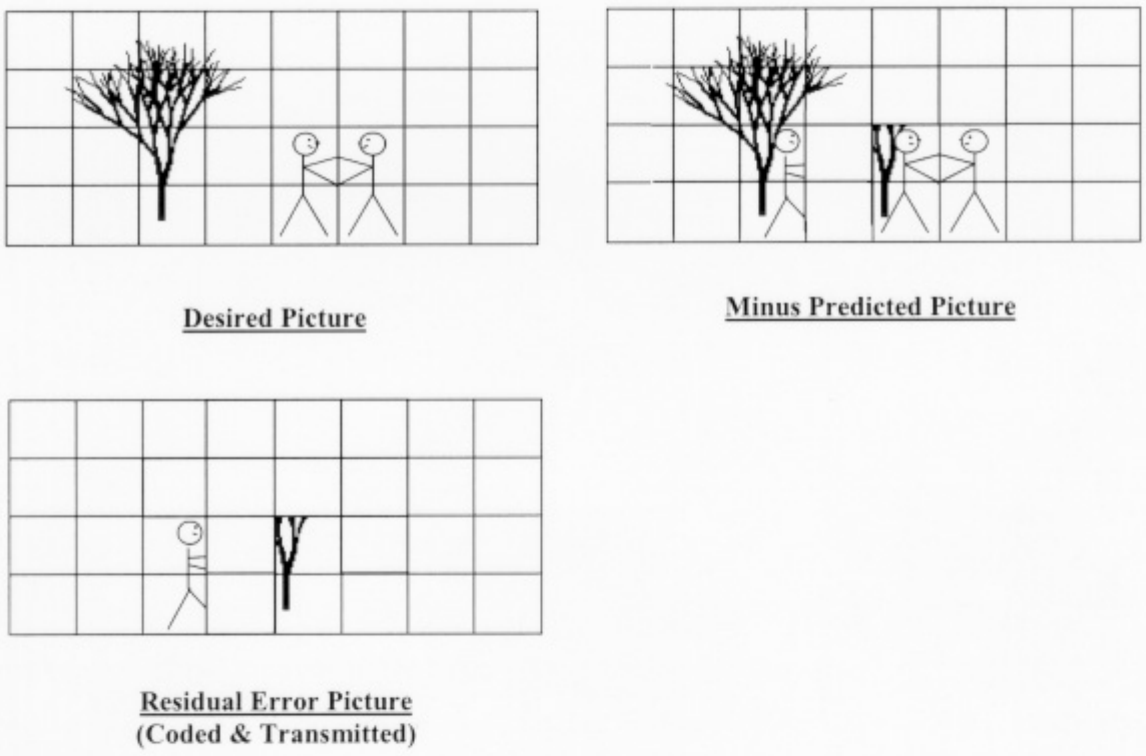

Figure 22

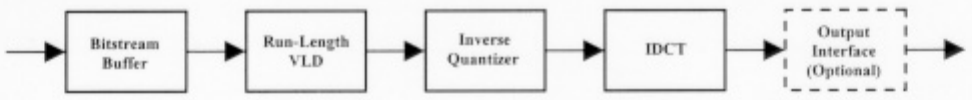

Figure 23: Intra Frame Decoder

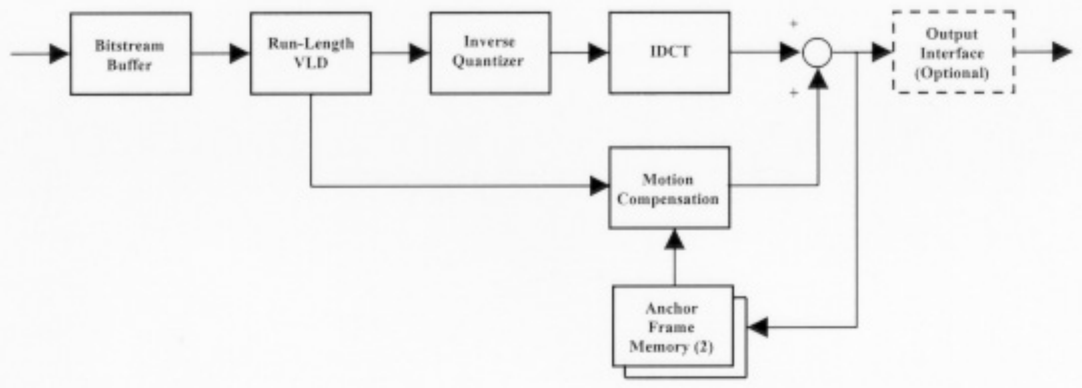

Figure 24: Non-Intra Frame Decoder