Faculty of Computing and Information Technology

Department of Robotics and Digital Technology

Technical Report 93-11

The Theory of CCITT Recommendation H.261, "Video Codec for Audiovisual Services at $p \times 64$ kbit/s" and review of such a codec

Andrew Moore

December 16, 1993

Enquiries:-

Technical Report Coordinator
Robotics and Digital Technology
Monash University
Clayton VIC 3168
Australia

tr.coord@rdt.monash.edu.au +61 3 565 3402

Contents

List of Tables

List of Figures

Abstract

This paper gives information about H.261, including background on the design of the H.261 specification. The implementation covered is an H.261 codec designed to run using the Internet protocols as its transmission system. The discussion of the implementation includes information about and resolution of problems that result from using the Internet protocols as a trasmission system.

Keywords

H.261, codec, Internet, video, videoconference, DCT

Chapter 1

Introduction

Objective

This paper describes the operational theory of the CCITT Recommendation H.261 [3,16] and then proceeds to cover an implementation of a codec satisfying the H.261 standard.

This paper will cover the background of the H.261, its design, information on components of the H.261 specification, finally covering particulars on the final coding in the H.261 standard. In dealing with the implementation [26], the second part of this paper will review an implementation of the H.261 as an Internet protocol based codec. In this section information on the basic make up of a software implementation of H.261 will be discussed in addition to information about and resolution of problems present because of the use of an Internet protocol basis for communications.

Chapter 2

The H.261 Standard

2.1 Background

Video coding has a history that started in the 1940s when C. E. Shannon and others first postulated the probabilistic view of information and its representation, transmission and compression [9,25,7]. The growth in the image compression field has been steady, driven by its contribution to other fields such as multimedia computing. As a result, there have been many different and varied developments in the video coding and compression techniques.

The H.261 recommendation [3] came to be after the CCITT recognized a need to provide a universal standard for video coding and transmission using Integrated Services Digital Network (ISDN) [10]. At this time, the CCITT established a Working Group to recommend a video coding standard for transmission at $m \times 384$ kbit/s ($m = 1,2,\ldots,5$). At a later stage in the study period, a standard for $n \times 64$ kbit/s ($n = 1,2,\ldots,5$) was also considered. With continued developments in video coding and compression techniques it became clear that single standard, $p \times 64$ kbit/s ($p = 1,2,\ldots,30$), could cover the entire ISDN channel capacity.

In December 1990, the CCITT completed and approved **Recommendation H.261, Video Codec for Audiovisual Services at $p \times 64$ kbit/s**.

The main applications intended for this international standard are for videophone and videoconferencing. Therefore, the recommended video coding algorithm has to be able to operate in real time with a minimum of delay. When $p = 1$ or 2, because of the severely limited band width available, only desktop, face-to-face, visual communication (*videophone*) is appropriate. For values of p greater than 5, with the additional available bit rate, more complex images can be transmitted with better quality. This increase in quality makes these higher data rates more suitable for applications such as videoconferencing.

2.2 Video Format

The CCITT decided on the uses of the Common Intermediate Format (CIF) and Quarter-CIF (QCIF) as the video formats for visual telephony. The sizes in addition to the luminance and color coding values are as defined in CCIR Recommendation 601[2]. The sizes of these formats are listed in Table 2.1. The specification of H.261 states that all codecs must be able to operate with QCIF format images. The use of CIF format (size)

Table 2.1: Parameters for the CCITT Video Formats. The values in brackets represent the coded pixels.

	CIF		QCIF	
	Lines/Frame	Pixels/Line	Lines/Frame	Pixels/Line
Luminance (Y)	288	360 (352)	144	180 (176)
Chrominance (C_B)	144	180 (176)	72	90 (88)
Chrominance (C_R)	144	180 (176)	72	90 (88)

frame is optional. The maximum data rate specified for both formats is 30000/1001 ($\approx$ 29.97) frames/second. There is additional provision for 1, 2 or 3 frames to be dropped by the encoder between transmitted frames, in this way performing a spatial subsampling of the image sequence.

$$352 \times 288 = 101376 \quad (2.1)$$

$$101376 \times 8 \times 29.97 = 24.31 \times 10^6 \quad (2.2)$$

$$176 \times 288 = 50688 \quad (2.3)$$

$$50688 \times 8 \times 29.97 = 6.08 \times 10^6 \quad (2.4)$$

Equations 2.2 and 2.4 show that the uncompressed data rates for transmitting CIF and QCIF at 29.97 frames/second are 24.31 and 6.08 Mbit/s respectively. A great deal of bit-rate reduction is required to transport such video signals via ISDN channels ($p \times$ 64kbit/s, p 1,2,. . . ,30). The choice of CIF or QCIF depends on the availability of channel capacity. For $p = 1$ or 2, QCIF is normally used for applications such as desktop videophone. Even if QCIF operated at 10 frames per second, a bit-rate reduction of approximately 92.7 to 1 is needed to transport that signal using a 64 kbit/s channel. Such compression is a major undertaking.

For p greater than 5, CIF may be used since there are more bits available to code a picture. Due to its increased resolution, CIF is appropriate for applications such as full videoconferencing.

The Block is an integral part of the hierarchical encoding structure, a block is made up of 8×8 luminance samples and a pair of 4×4 Chrominance samples (a pair: one for the C_B image and one for the C_R image.) Four blocks make up a Macro Block [3]. The hierarchical structure of the image data is discussed in greater detail later. The composition of a Macro Block is shown in Figure 2.1.

In this way, each Macro Block is composed of four 8×8 luminance (Y) Blocks and two 8×8 chrominance (C_B and C_R) Blocks. A Group of Blocks (GOBs) is composed of 3 × 11 Macro Blocks. A CIF Picture has 12 GOBs while a QCIF has only three GOBs. This elaborate hierarchical block structure is essential for the high-compression video coding algorithm, which will be described in further detail later.

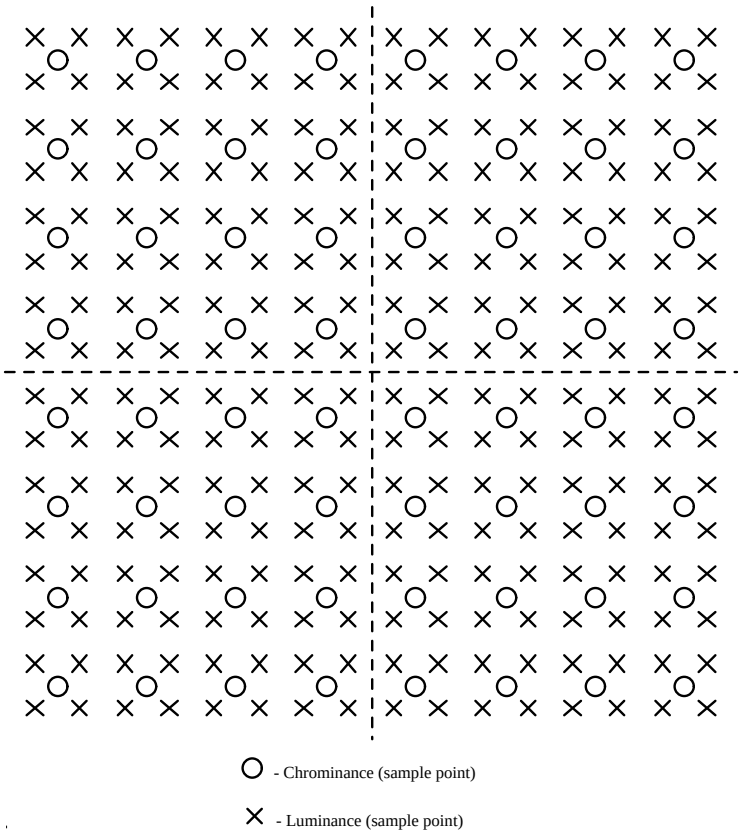


Figure 2.1: Structure of a Macro-Block showing sampling structure of Luminance and Chrominance

.

2.3 Video Coding Algorithm

The basic objective of compression of any sort is to reduce the required bit rate by removing redundant information carried in a particular form. For video images, like other compressed data, there are two sorts of video compression/coding schemes: there is source coding and there is entropy coding.

Source coding deals directly with characteristics of the source material and yields results which are lossy, i.e., picture quality is degraded, while entropy coding relies on the statistical properties of the signals and, in theory, is lossless. H.261 makes use of both techniques.

Source coding can be divided into two further types: intraframe and interframe. Intraframe coding is used for the first picture in a sequence and for images after a scene change, interframe coding is used for sequences of similar images, including those containing moving objects. Intraframe coding removes redundant spatial information within an image, while interframe coding in addition to removing redundant spatial information, also removes temporal redundancy between images.

Due to the nature of Interframe encoding (each *frame* of data is actually a set of differences between that frame and the previous frame), error accumulation is more likely. Apart from any transmission errors effecting image data, there will be an accumulated systematic error incurred from one interframe to the next. This error is due to the fact that there will be error tolerances in the creation of differences between the images, these errors will be between the original intraframe and the following interframes. Such errors between the original frame and the recreated frame will be progressively accumulated because each new interframe (with its own errors) is created from the previous frame (inter or intra) and not as a difference of (for example) the original intraframe reference. As a result, intraframes are regularly sent (every 30 inter frames, for example[1]) to stop this accumulation of errors in the resultant image.

Under certain circumstances, H.261 makes use of both source and entropy coding as well as the inter and intra variations of source coding.

Figure 2.2 gives a diagramatic overview of the H.261 codec. In this diagram the transmission coder and receiving decoder are entropy style systems, whereas the source coder and source decoder are based on *lossy* inter/intraframe style source encoding. In addition to encoding there is the *optional* addition of the BCH[2] $(511, 493)$ forward error correction code. The use of this *optional* error correction code will not be discussed in this paper.

2.3.1 Source Coder/Decoder

A block diagram of the source decoder for H.261 is shown in Figure 2.3. This system is a hybrid of the Discrete Cosine Transform (DCT) and a Differential Pulse Code Modulation (DPCM) schemes incorporating motion estimation. In intraframe coding mode, the DPCM is not operative.

Each 8×8 block in the image is transformed into its DCT coefficients, linearly quantized and then sent to the video multiplex coder. This same picture is also recovered

[1]The H.261 standard [3] defines that an intraframe must be sent for every 132 interframes. Although intraframes can be sent more often.

[2]Bose, Chaudhuri and Hoequenham (code)[18]

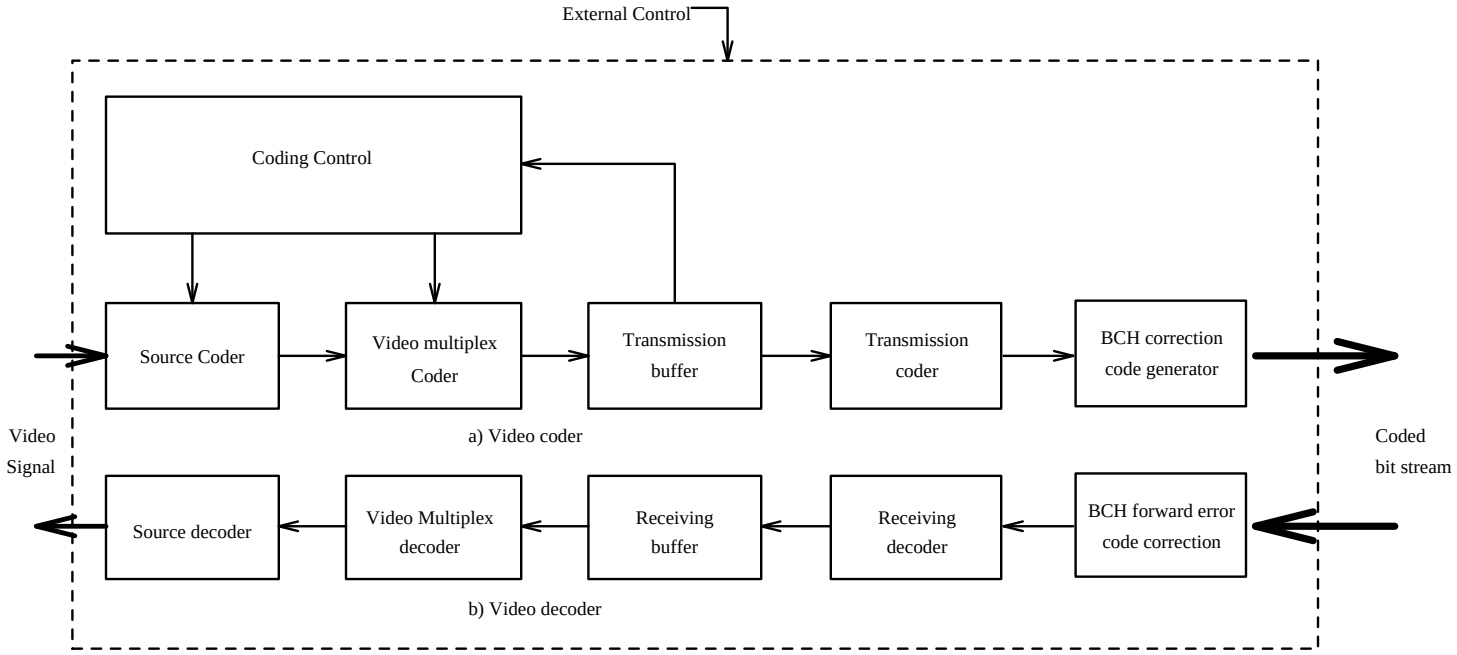


Figure 2.2: Block diagram of the complete video codec

.

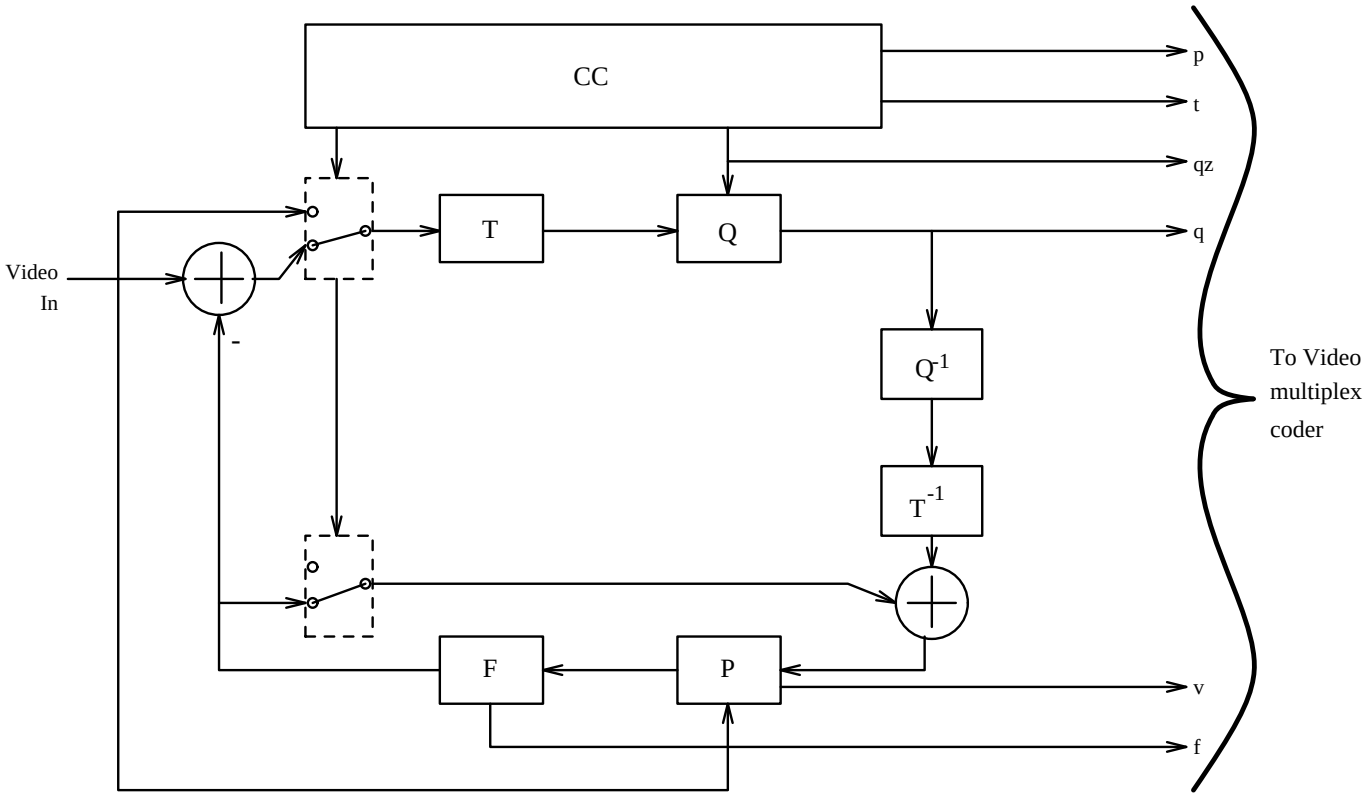
(through an inverse quantizer and inverse DCT transform) and stored in an image memory for use in interframe coding.

The DPCM style of coding is used in the interframe coding mode. The prediction is based on motion estimation by comparing every Macro Block (only luminance blocks are processed) of the current frame with the Macro Blocks in the neighbourhood of the corresponding Macro Block in the previous frame. If the difference between the current and the predicted Macro Blocks is less than a particular threshold, then no data is transformed for that particular Macro Block, instead only a motion vector is encoded. Otherwise, the difference is transformed (by DCT), linearly quantized and then sent to the video multiplex coder along with the motion vector information. A loop filter can be switched on and off to improve image quality be removing high-frequency noise as required.

The step size of the linear quantizer can be adjusted dependent upon the fullness of the transmission buffer of the encoder. When the transmission buffer is close to being full, the quantizer step size can be increased, thereby decreasing the amount of information generated (and to be transmitted) by the encoding. The immediate result (apart from a *consistent* amount of data throughput) is a reduction in the image quality. Alternately, should the transmission buffer be relatively empty, the quantization step can be reduced, thereby increasing the amount of information generated (and to be transmitted) by the encoder.

In a further attempt to increase the efficiency of the coding, variable word-length entropy coding is used in the video multiplex coder which immediately follows the source coder stage. There are five variable word-length coding tables for the quantized DCT coefficients and various other information.

The output of the video multiplex coder is sent to a transmission buffer, which will regulate the flow to a *constant* rate by (as mentioned above) providing feedback for control of the step size of the linear quantizer.



T Transform
Q Quantizer
P Picture memory with motion compensated variable delay
F Loop Filter
CC Coding control
p Flag for INTRA/INTER
t Flag for transmitted or not
qz Quantizer indication
q Quantizing index for transform coefficients
v Motion vector
f Switching on/off of the loop filter

Figure 2.3: Block diagram of the Source coder

2.3.2 Discrete Cosine Transform (DCT)

Transform coding systems based on the Karhunen-Loève (KLT), discrete Fourier Transform (DFT), discrete Cosine Transform (DCT), Walsh-Hadamard Transform (WHT), and a number of other transforms have been constructed and studied extensively. The choice of a particular transform in a given application depends upon the properties of the application (for example, *how much inter-element correlation is present*), upon the amount of reconstruction error that can be tolerated and the computational resources available.

Compression is achieved during the quantization of the transformed coefficients (not during the transformation step).

The mean-square error between subimage $\mathbf{F}$ (of size $n \times n$) and approximation $\hat{\mathbf{F}}$ is given by:

The variable $m(u,v)$ is a particular transform coefficient *masking function* designed to eliminate the basis images that make the smallest contribution to the total post transform images.

$$m(u,v) = \begin{cases} 0 & if\ T(u,v)\ satisfies\ a\ specified\ truncation\ criteria \\ 1 & otherwise \end{cases} \tag{2.5}$$

$\mathbf{H}_{uv}$ is the array of basis images of the transform defined as:

$$\mathbf{H}_{uv} = \begin{bmatrix} h(0,0,u,v) & h(0,1,u,v) & \dots & h(0,n-1,u,v) \\ h(0,0,u,v) & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ h(n-1,0,u,v) & h(n-1,1,u,v) & \dots & h(n-1,n-1,u,v) \end{bmatrix} \tag{2.6}$$

$\mathbf{H}_{uv}$ can be used to compute the series expansion weighting coefficients $T(u,v)$.

$$e_{ms} = E\{\|\mathbf{F} - \hat{\mathbf{F}}\|^2\} \tag{2.7}$$

$$= E\{\|\sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\mathbf{H}_{uv} - \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)m(u,v)\mathbf{H}_{uv}\|^2\} \tag{2.8}$$

$$= E\{\|\sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\mathbf{H}_{uv}[1-m(u,v)]\|^2\} \tag{2.9}$$

$$= \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} \sigma^2_{T(u,v)}[1-m(u,v)] \tag{2.10}$$

where $T(u,v)$ is the 2-D transform of the image f(x,y), $\|\mathbf{F} - \hat{\mathbf{F}}\|$, is the matrix norm of $(\mathbf{F} - \hat{\mathbf{F}})$ and $\sigma^2_{T(u,v)}$ is the variance of the coefficient at transform location (u,v).

It can be shown ([7,5,20,31]) that while DCT has a superior information packing ability to the DFT and WHT, and this is usually the case for most naturally generated images (i.e. *not* a random white noise image,) in fact, the KLT, not the DCT, is the optimal transform in an information packing sense. That is, the KLT minimizes the mean-square error (Equations 2.8,2.10) for any input images and any number of retained coefficients. However, because the KLT is data dependent, obtaining the KLT basis images for each subimage, in general, is a non-trivial computational task. For this reason, the

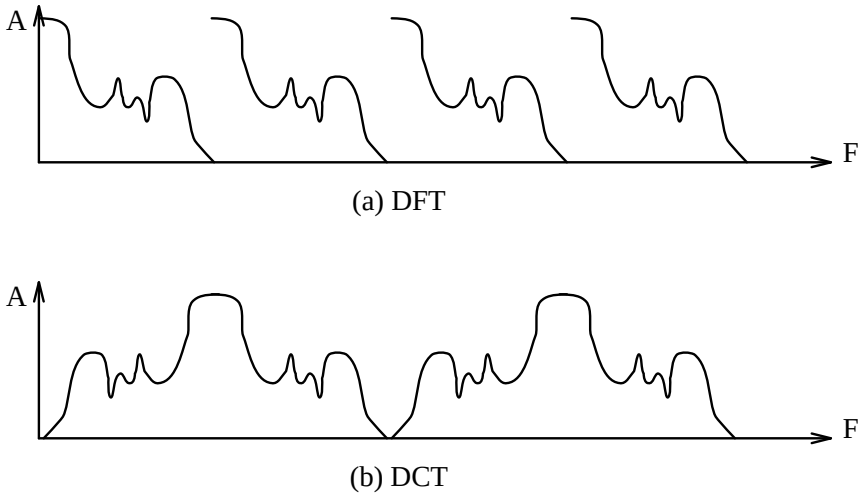


Figure 2.4: Periodic nature of DCT and DFT transforms

KLT is seldom used in practice. Instead of the KLT, a transform such as DCT is used. The DCT uses fixed basis images (input independent) thereby dramatically reducing the computation required. Additionally, the DCT closely approximates the information packing ability of the optimal KLT.

Gonzalez and Woods (1992) [7] reason that in comparison with the DFT, the periodicity of the DCT lends itself to being less prone to *blocking artifacts*[3]. Whereas the DFT will cause Gibbs phenomenon at the block boundaries, thereby giving erroneous values at the discontinuities and boundaries. The DCT, on the other hand, reduces this effect because of its implicit $2n$ point periodicity does not inherently produce boundary discontinuities. The periodic nature is shown in comparison to the DFT in Figure 2.4

As a result of these factors, most practical transform encoding systems are based on the DCT, which provides a good compromise between information packing ability and its computational requirements.

For the particular application at hand, each transmitted block is first processed into separable two-dimensional, 8×8 blocks, these in turn are transformed using the DCT. Equation 2.11 shows the forward transformation for the generation of a discrete cosine

[3]The results of boundaries between subimages becoming overly obvious.

transform of the original 8×8 image block.

$$F(u,v) = 1/4C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)cos[\pi(2x+1)u/16]cos[\pi(2y+1)v/16] \quad (2.11)$$

with $u, v, x, y = 0, 1, 2, \ldots, 7$
where,

$$\begin{aligned} x, y &= \text{spatial coordinates in the pixel domain} \\ u, v &= \text{coordinates in the transform domain} \\ C(u) &= 1/\sqrt{2} \text{ for } u = 0, \text{ otherwise } 1 \\ C(v) &= 1/\sqrt{2} \text{ for } v = 0, \text{ otherwise } 1 \end{aligned}$$

Equation 2.12 shows the inverse transformation for the recovery of an approximation of the original 8×8 image block.

$$f(x,y) = 1/4\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)cos[\pi(2x+1)u/16]cos[\pi(2y+1)v/16] \quad (2.12)$$

with $u, v, x, y = 0, 1, 2, \ldots, 7$
where,

$$\begin{aligned} x, y &= \text{spatial coordinates in the pixel domain} \\ u, v &= \text{coordinates in the transform domain} \\ C(u) &= 1/\sqrt{2} \text{ for } u = 0, \text{ otherwise } 1 \\ C(v) &= 1/\sqrt{2} \text{ for } v = 0, \text{ otherwise } 1 \end{aligned}$$

Note: Within the block being transformed, x=0 and y=0 refer to the pixel nearest the left and top edges of the image.

2.3.3 Differential Pulse Code Modulated/Discrete Cosine Transform Hybrid (DPCM/DCT)

Differential Pulse Code Modulation is only relevant for the interframe case in image processing. Intraframe encoding involves only a single frame (with no reference to the previous frame), the frame is subdivided and each component encoded independently using a DCT and quantization step.

In interframe encoding, the previous frame is regenerated using an inverse of the quantizer and an inverse DCT, this is then stored in a frame buffer for use as follows. In

the interframe mode of Source coding, only the differences between two particular frames need to be sent. (In a sequence of images/frames there is often much redundant temporal information.)

This is done, in the simplest form, by subtracting the new frame from the reconstructed frame in memory. Now these differences are then encoded and quantized to form the data to be transmitted. Often (and in the case of H.261) there is a simplification process whereby if parts of the image (particular 8×8 blocks) have not changed by greater than a particular threshold these components are not transmitted, *in effect* the decoder will use the previous block contents. By not sending redundant, duplicated information, this in turn reduces the amount of transmitted data, thereby giving additional compression.

2.4 Other components in H.261

2.4.1 Motion Compensation

Motion Compensation is an option in the implementation of an encoder. Motion compensation is an additional technique of removing further temporal redundancy transmitted while in the interframe encoding mode of operation. In this mode an encoder allocates a motion vector for each Macro Block (MB.) Both horizontal and vertical components of these motion vectors have integer values not exceeding ± 15. This vector must be representative of the whole Macro Block as it is used for all four luminance blocks in a Macro Block.

A positive value of the horizontal or vertical component of the motion vector signifies that the prediction is formed from pixels in the previous frame which are spatially to the right or below the pixels being produced.

Because of the numerous amount of sampling and thus the inherent amount of computation required, motion estimation is an option in the H.261 coder implementation.

If we use an example, we can show how motion estimation can reduce image data required to be transmitted.

Simple Motion Compensation Figure 2.5 shows how motion compensation detection can reduce the amount of information required to be transmitted. In the figure, (a) shows the original (or reference) frame and (b) shows the next frame. The two images are identical except for a translation (of *two* to the right and *one* down,) thus it should be intuitive that instead of the encoded information contain two copies of similar but redundant information (one for each image), the encoded information should contain the original image and then the following image be marked to say it makes use of motion compensation and in this case, contains only a vector of the motion of this image relative to the previous image.

For this simple example, this would reduce the amount of information from a full second copy of the image, down to only a vector consisting of two parts. This example greatly oversimplifies the additional information required in the coding process, including information to indicate whether motion compensation is used, but in general should show in this example how motion compensation will reduce the amount of coded information required.

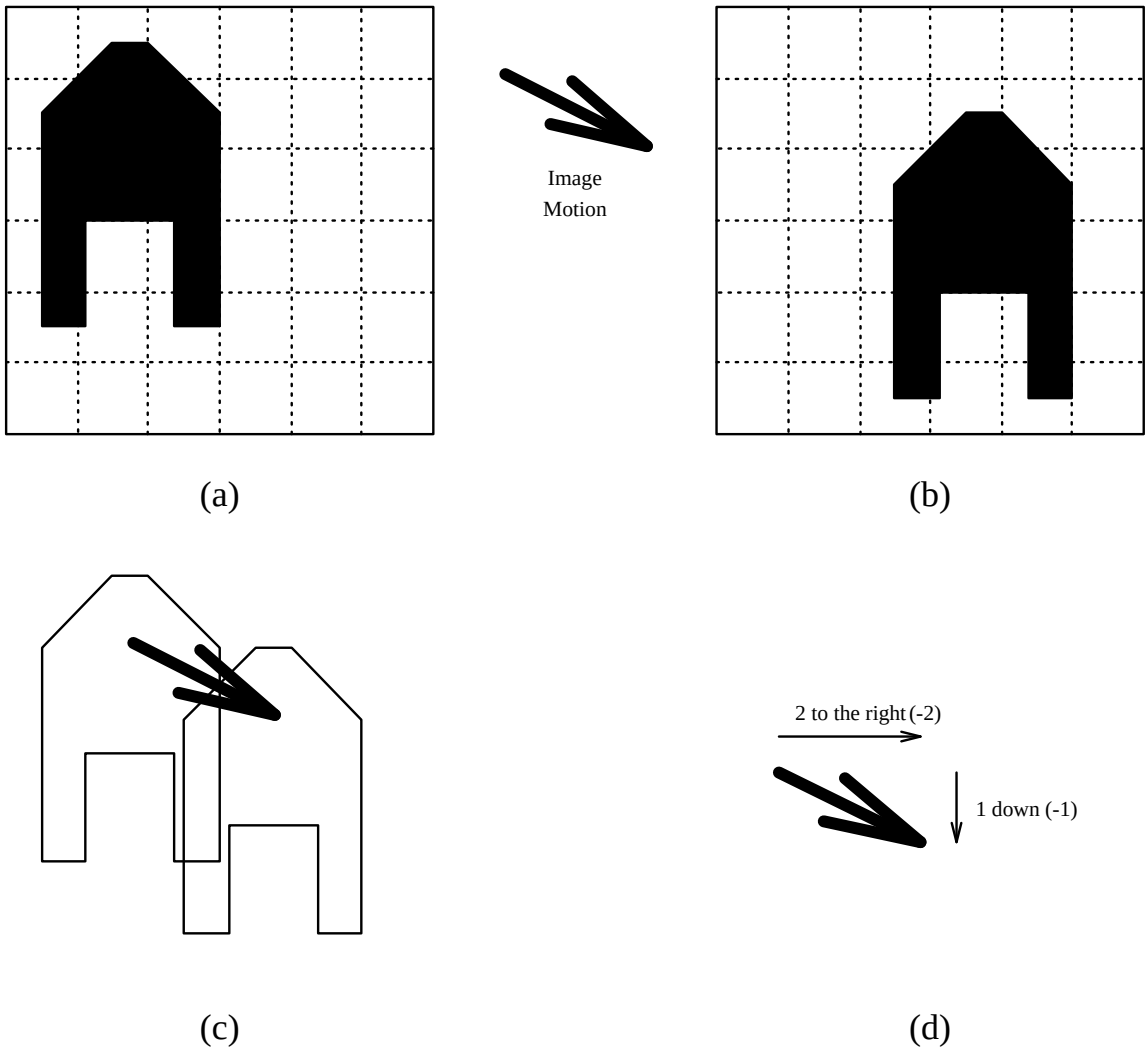


Figure 2.5: A Simple Motion example: (a) shows the original (or reference) frame, (b) shows the next frame, (c) indicates the motion in the image of the object and (d) shows the motion as broken down into its horizontal and vertical components.

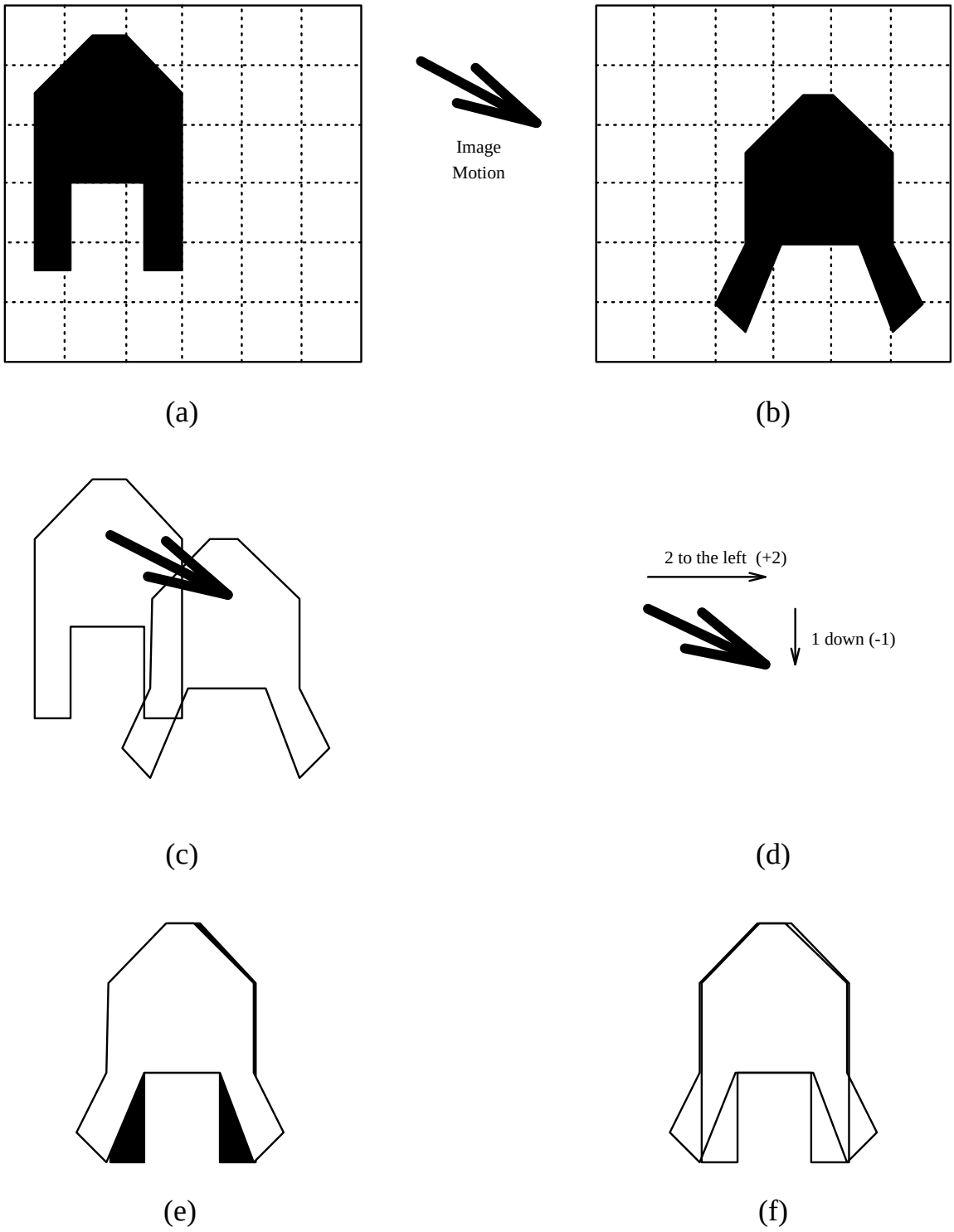


Figure 2.6: A Complex Motion example: (a) shows the original (or reference) frame, (b) shows the next frame, (c) indicates the motion in the image of the object, (d) shows the motion as broken down into its horizontal and vertical components, (e,f) shows the difference between the original and resultant images (when overlaid) (e) shows the *differences* shaded, while (f) shows the overlay in a transparent way.

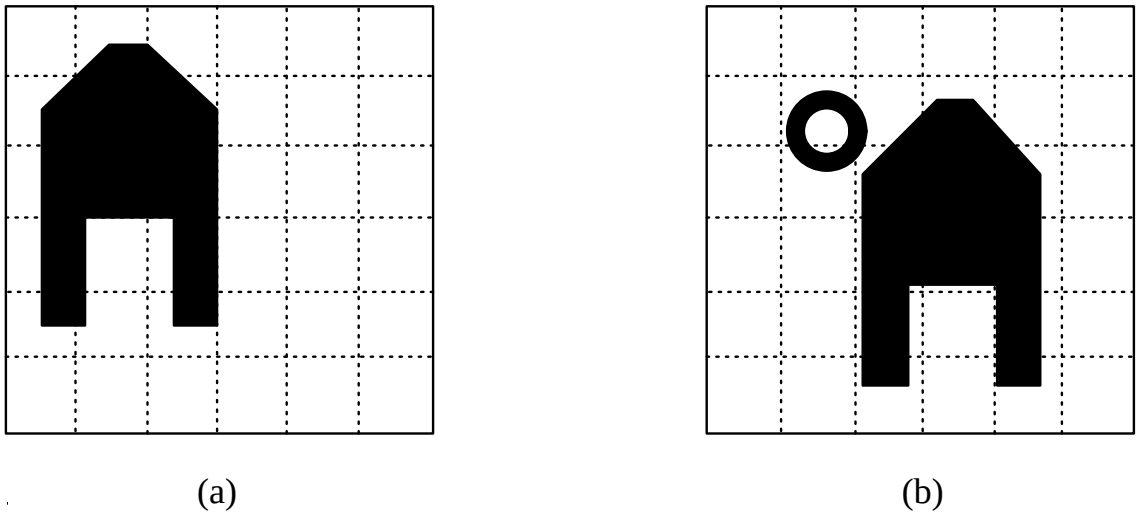


Figure 2.7: A Second Complex Motion example: (a) shows the original (or reference) frame, (b) shows the next frame. In the second frame an object (the *donut*) has been revealed by the motion of the original object.

Complex Motion Compensation Figure 2.6 shows a more complex motion compensation example. In this figure, (a) and (b) show consecutive frames. Figure 2.6, (e) and (f) show the differences between the images. In sub-image (e) these areas are shaded black.

It can be seen that while there is motion, and the images are similar, they are not identical. In this situation, motion compensation can still save on the amount of information that needs to be encoded.

For this example, the encoded information contains the first image and instead of the second image, it encodes a motion vector for the previous image (similar to the previous example) and the differences between the images are encoded as well. This amount of information will still be less than if the two images (a) and (b) were separately encoded.

Another common complex motion compensation situation is shown in Figure 2.7. In this example, the motion of one object has added an object to the scene (the torus shape, or *donut*). Such revealing of an object means that in addition to the encoding of the motion of the image (in this case the A shape) the encoded image will also contain information describing the revealed object.

Some situations where the motion compensation is made complex will occur when the motion of an object reveals other objects, obscures other objects or makes changes to the background (for example, introducing a shadow or filter).

While simplifying the process, this shows that in all images where an amount of motion is present, motion compensation will give some amount of reduction in the amount of encoded information. The difficulties of motion compensation relate to the amount of image processing (comparison, subtraction, etc.) required to (a) determine if motion compensation is worthwhile and (b) generate the coded information images.

Several techniques for performing the motion detection and motion estimation exist [13,22,17]. Motion estimation requires enormous processing to sample and compare the two images between which motion is to be detected. As a result of this, a number of techniques to maximize the efficiency and minimize the time required for this task have been developed, the simplest is to subsample the two images to compare. This

subsampling technique is use in the H261 encoder examined in the second part of this paper.

2.4.2 Filter

The prediction process may be modified through the use of a two dimension spatial filter which operates on pixels in a particular 8×8 block. The filter is used particularly in the case of motion compensation thus the filter is switched on/off for a particular Macro Block, and it acts upon all the blocks inside a particular MB (like the motion estimation itself).

The filter is able to be seperated into one-dimensional horizontal and vertical functions. Both of these are non-recursive with coefficients of $\frac{1}{4}$, $\frac{1}{2}$, $\frac{1}{4}$ except at block edges where one of the taps would fall outside the block. In this case, the 1-D filter is changed to have coefficients of 0, 1, 0. Full arithmetic precision is retained with rounding to 8 bit integer values only at the 2-D filter output. Values whose fractional part is one half are rounded up.

Equation 2.13 shows the two separable components for vertical and horizontal, as well as the 3×3 array of the 2-D filter.

$$\begin{bmatrix} \frac{1}{4} \\ \frac{1}{2} \\ \frac{1}{4} \end{bmatrix} \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix} = \begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix} \tag{2.13}$$

The filter becomes necessary, especially in the case of motion compensation, because for motion compensation there is an assumption of integer amounts of movement by the Macro Block (from 1 to 15 pixel width moves in either up/down or left/right directions.) As a result, when an image moves by less (or more) than an integer amount, the motion compensation routines will *enforce* an integer amount of movement. This has the effect of causing jumping by the image, without smooth movement.

A filter will offset this problem by causing an image to be low-band filtered, as a result images (lines etc) will have a less clearly defined edge and appear to be more smooth in motion. The disadvantage of the filter is that images will appear fuzzier.

A case where the image as placed and the real image do not actually agree is shown in Figure 2.8. In this figure, the motion will place the image in a position where it overlaps adjacent pixels (although it is predominantly placed in one particular set).

The use of the filter is not restricted to use in combination with the motion compensation system. The filter can be used at any time. This becomes useful when image details are finer than the display resolution and, as a result, motion will appear to jump or appear jagged. Using the filter, the image outlines will tend to be blurred, the images will appear softer, as a result motion will not appear as jumpy. The filter does not increase the picture information, it removes some of the information (low-band pass) and this results in smoother, softer images.

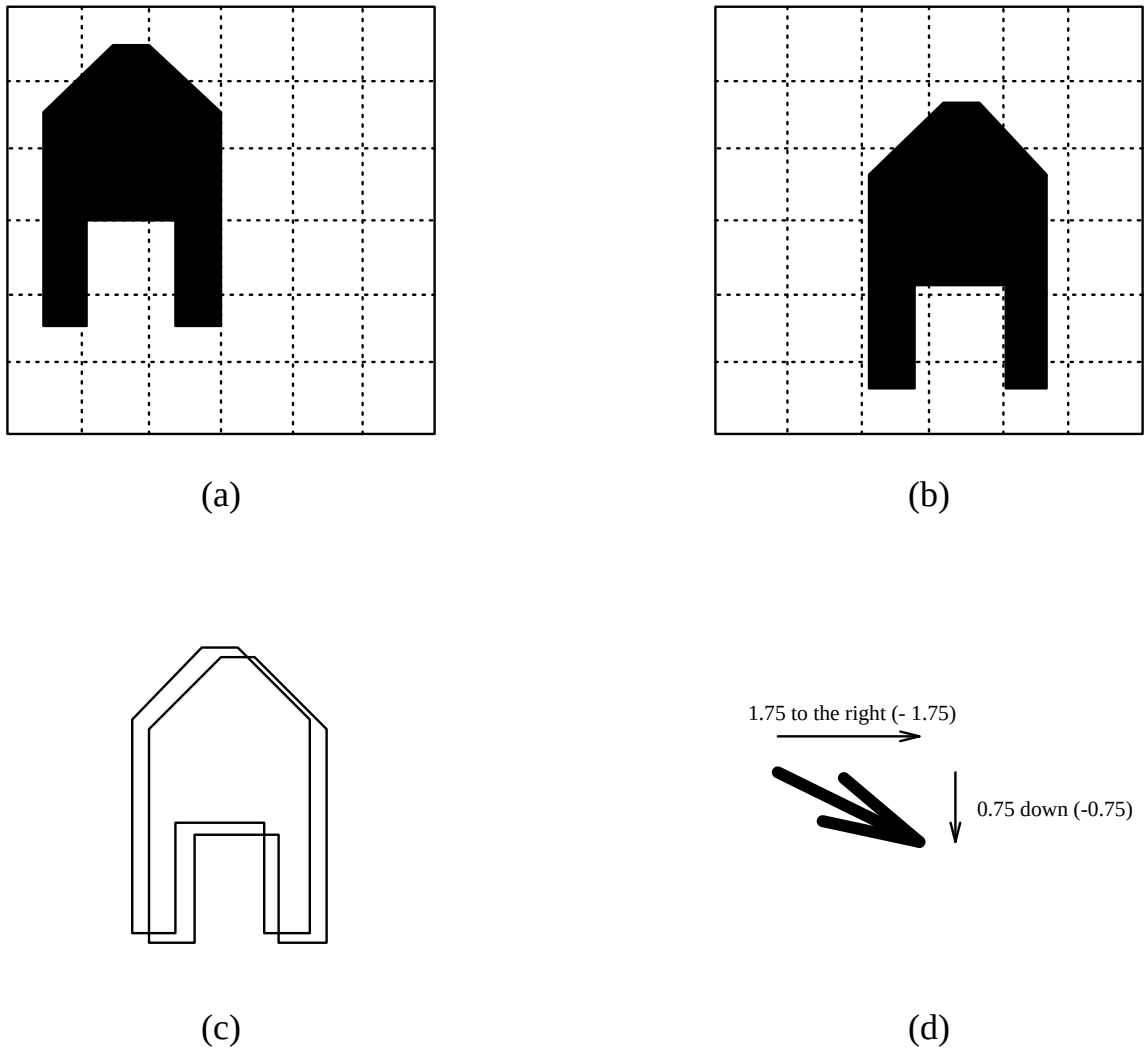


Figure 2.8: Non-integer motion: (a) shows the original (or reference) frame, (b) shows the next frame, (c) shows the non-exact match between the real image movement and the integer interpolation of that movement and (d) shows the motion as broken down into its horizontal and vertical components.

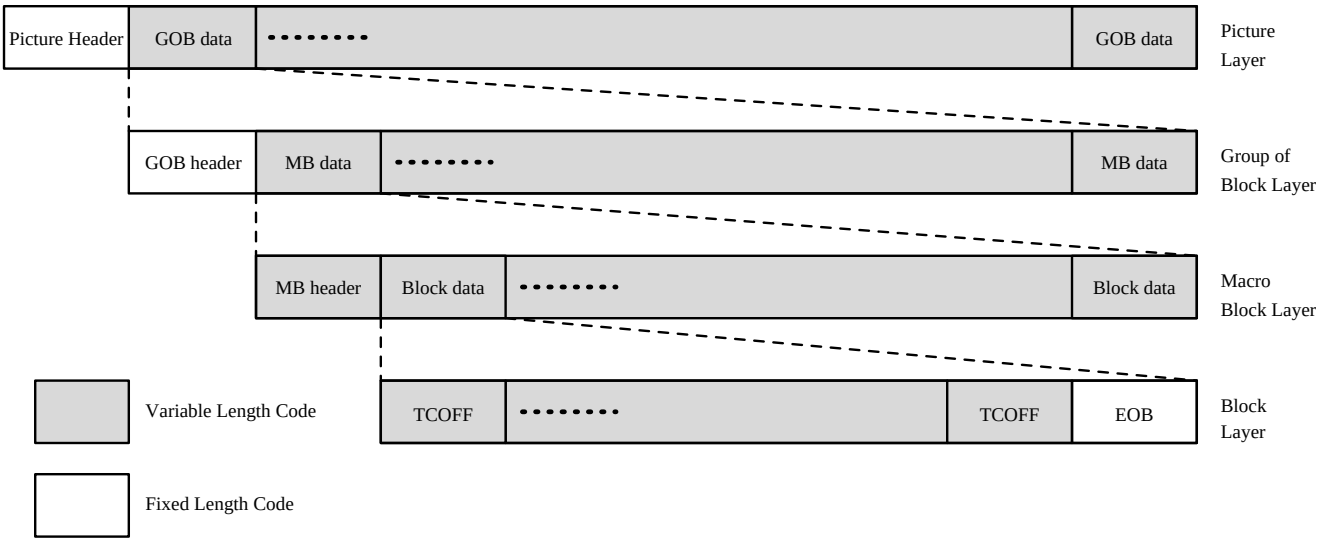


Figure 2.9: Hierarchical nature of the data structures of the coder

2.5 Encoded Image Data Structures

The video coding algorithm describes a method whereby the CIF and QCIF images are divided into a hierarchical block structure consisting of Pictures, Groups of Blocks (GOBs) Macro Blocks (MB) and Blocks. The relationship between each data structure is show in Figure 2.9.

Data for each image consists of a picture header followed by data for the GOBs. The image header includes a 20-bit image start code in addition to other information such as the format of the picture (CIF or QCIF), frame number (temporal reference) etc. Figure 2.10 shows the arrangement of GOBs in an image.

The GOB layer consists of a GOB header, containing a 16-bit GOB start code, the position of this GOB (in the image), quantizer information for the GOB (unless overwritten by an MBs own quantizer set), etc. This is followed by data for the MBs themselves. Figure 2.11 shows the arrangement of Macro Blocks in a GOB.

The MB layer consists of an MB header, containing the variable length code (VLC) for the MB address, the VLC for the MB type (intra/inter, motion estimation on/off, loop filter on/off). This information is in turn followed by the data for Blocks. Figure 2.12 shows the arrangement of Blocks in a Macro block; note the difference between CIF and QCIF formats.

Various sorts of video side information may also be included in an MB header, this becomes dependent on the specifics of the MB itself.

When motion estimate information for a particular block is of the required accuracy, no block data for DCT coefficients needs to be transmitted. It should be noted that not every MB in a GOB needs to be transmitted if it contains no information for that part of the picture.

The Block layer contains DCT coefficients (TCOEFF) of a block followed by a fixed length code, EOB, to indicate the end of a block. The coefficients are coded using two dimensional VLC.

Like all other levels in the hierarchy, not every block in an MB needs to be transmitted if it contains no additional information.

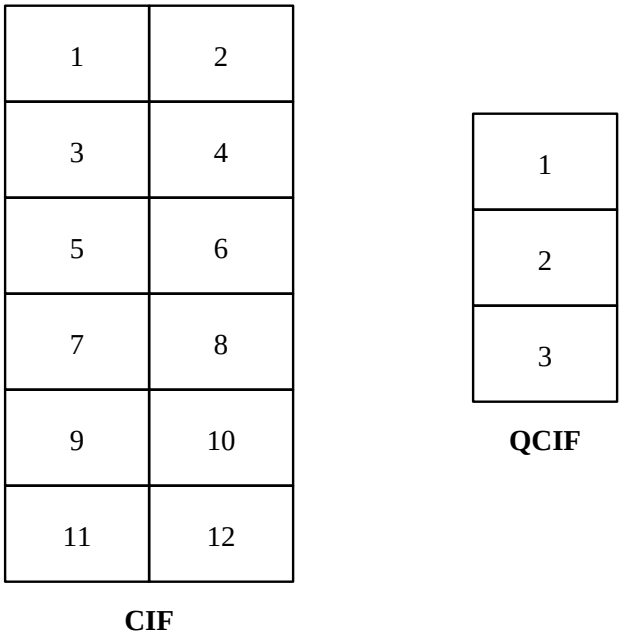


Figure 2.10: Arrangement of GOBs in an image

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28	29	30	31	32	33

Figure 2.11: Arrangement of Macro Blocks in a GOB

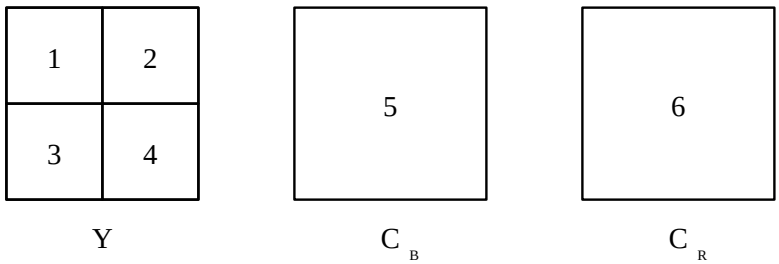


Figure 2.12: Arrangement of blocks in a Macro Block

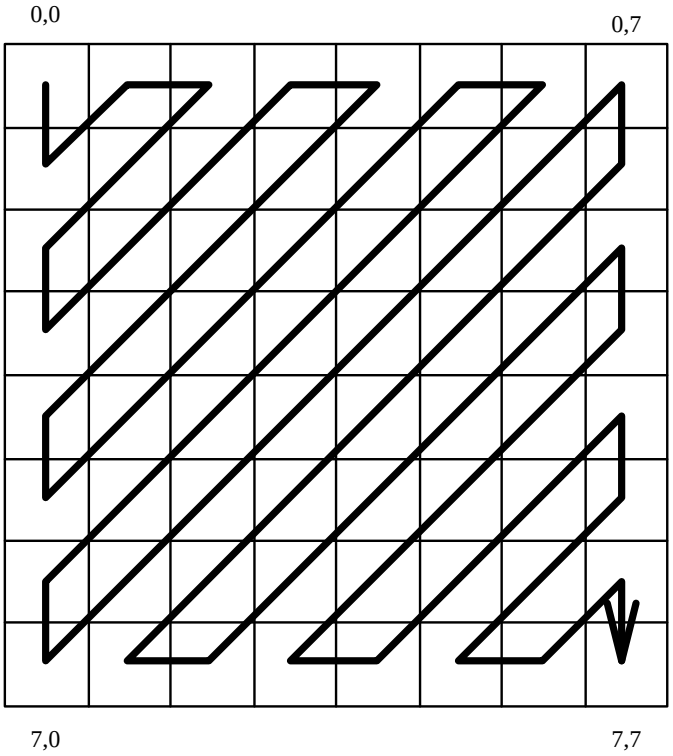


Figure 2.13: Zig-zag sampling of a 2 dimensional 8×8 data block into a 64 element linear array.

2.6 Entropy encoding

The entropy coder consists of two main components:

- zig-zag reorganization of samples
- the Huffman encoding

The zig-zag reorganization of the samples in a block (shown in Figure 2.13) describes the manner in which an 8×8 block of samples is converted into a 64 item linear array. This array represents the results from a two dimensional DCT, where (0,0) is the DC component and (7,7) would be the highest frequency [14].

Now that the block is in a linear array format, the entropy encoder can perform encoding using a modified Huffman coding scheme. This results in a variable bit rate encode for any particular block. Modified Huffman encoding is a practical implementation of the Huffman algorithm. (From Jain (1989) [12]): ...

$$i = ql_1 + j, 0 \leq q \leq Int[\frac{(L-1)}{L_1}], 0 \leq j \leq L_1 - 1 \tag{2.14}$$

The first L_1 symbols are Huffman coded. The remaining symbols are coded by a prefix code representing the quotient q, followed by a terminator code, which is the same as the Huffman code for the remainder $j, 0 \leq j \leq L_1 - 1$. ...

Symbol	Binary Code	*p*	Huffman code (HC)
s_1	0 0 0	0.25	0 0
s_2	0 0 1	0.21	1 0
s_3	0 1 0	0.15	0 1 0
s_4	0 1 1	0.14	0 1 1
s_5	1 0 0	0.0625	1 1 0 0
s_6	1 0 1	0.0625	1 1 0 1
s_7	1 1 0	0.0625	1 1 1 0
s_8	1 1 1	0.0625	1 1 1 1
Average code length	3.0	2.781 (entropy)	2.79
Code efficiency *H/B*	92.7%		99.7%

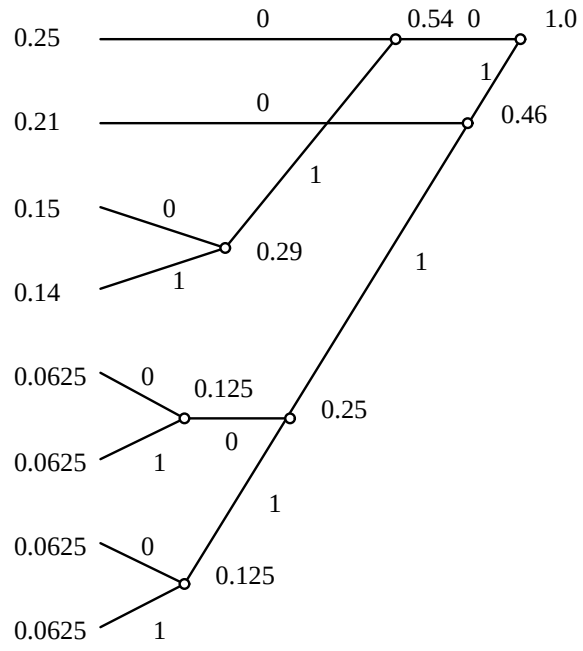


Figure 2.14: Huffman coding example.

This may be better explained by Figure 2.14 (also from Jain (1989)) in which it can be seen the way in which a probability tree is built up from a statistical analysis of the data and bits are allocated to the data on the basis of this probability.

The key to Huffman coding is the use a variable number of bits to represent the data. The more popular the item of data, the less the number of bits used to represent that data.

The H.261 standard takes the use of Huffman one step further by predefining the Huffman compression tables to be used at each compression stage (Block, Macro Block, Group of Blocks, and Image.) Such precalculation has the effect of speeding throughput in this stage of the compression sequence.

Chapter 3

An H.261 implementation

3.1 Overview

The implementation discussed is authored by Thierry Turletti of INRIA in France. It is referred to as h261_encode/h261_decode [26] and has been used as in an Internet Protocol [6,11] based video conferencing system [27].

The H.261 codec has not been written for use over a $p \times 64$ kbit/s data channel as the specification was intended for [3], instead the codec has been written for transfer of images over the packet based networks making up the *Internet*. Naturally this has had several major impacts on the implementation; however, the functionality of the codec implementation still stands.

The h261 software codec is divided into two program components:

h261_encode - being an image encoder satisfying the H.261 recommendations

h261_decode - being an image decoder compatible with **h261_encode** and also satisfying the H.261 recommendations

The software is designed to operating under the UNIX operating system [24,21] making use of the X-Windows system for the purposes of display of the images [23,19].

The encoder has been designed to function with the **VideoPix** video frame grabber for Sun Microsystem's computers. Although, the encoder also works by taking undecoded image files as input. The codec then compresses the image in accordance with the H.261 standard and either transmits the image to a decoder via TCP/IP or stores the h261 encoded output in a file.

The software encoder, at this stage, only supports Black and White images (thereby only dealing with the luminance components of the signals.) This has the considerable advantage of reducing the complexity of the software codec, while still achieving the desired functionality.

3.2 VideoPix imaging hardware

While the codec has been designed specifically with the **VideoPix** frame grabber in mind, the software should be able to be adapted to most other frame grabbers running on UNIX based computers, without many major changes. One direct result of being designed to use

the **VideoPix** hardware is that the software expects the video frame grabber to be slow, the VideoPix hardware is able to return a CIF size image in about 250ms. In testing, it appeared that the major *bottle-neck* impeding the efficient transfer of images is the transfer of the image across the bus (of the host computer) to the main memory, where h261_encode can manipulate the image [28].

The software is set up to take either PAL or NTSC format images as input, at this stage this only effects the size of the image *grabbed*.

3.3 Inter/Intra coding decisions

The first image is processed in intra mode. As there is no image preceding this one, an inter mode coding is not possible; as a result, source coding will only reduce spatial redundancies in the image.

With the inter mode of coding, possible after the coding (and transmission) of an intra frame, the elimination of redundancies in the temporal domain is also possible (e.g., unchanged parts of the picture). Additional temporal redundancies can be removed through the use of motion compensation routines.

3.4 Movement Detection

The movement detection algorithm aims at identifying the blocks in an image which are *sufficiently different* from corresponding blocks in the previous image. These blocks are *marked*. Only *marked* blocks are encoded in the inter mode (the rest are not considered to have changed enough to warrant transmission.) In each block, four successive comparisons are made in order to detect variations. If the sum of the four absolute differences exceeds a threshold, a decision is made to encode the Macro Block.

If the sum of the four absolute differences does not exceed the threshold, a motion vector is encoded for the Macro Block.

In order to test all regions of the block, the pixels tested are not from the same point from image to image. These comparisons are done for a significant number of pixels, i.e. pixels that are sufficiently scattered to cover the whole block. One pixel is chosen per 4×4 block for each 8×8 block (i.e. 4 pixels per block.)

Even with this *sub-sampling* of the image, detection of motion compensation takes approximately 2-3 times as long to compute as simply coding the inter-frame difference. It is interesting to note, as a result, that the choice of threshold becomes a compromise between the time taken to do the encoding and the sensitivity to movement that is required.

If, in interframe coding mode, the encoder finds redundancy at the Macro Block or even Group of Blocks level, these image components are simply not transmitted.

3.5 The DCT and Quantization

The coding algorithm takes as input a set of blocks (i.e. 8×8) pixels, and generates the transform coefficients for these blocks. In inter mode coding, the input blocks are only those blocks marked by the movement detection algorithm.

In intra mode coding, the input blocks are all the blocks of an MB. The H.261 recommendation does not specify at what point to choose inter or intra style encoding (this is left up to the implementer.) In this implementation, a full intra mode coding is used for the first image, for change of scene and each MB that is encoded uses intra mode after 30 inter encoding mode frames. A scene change is said to have occurred when the number of blocks marked by the movement detection algorithm exceeds some threshold.

As per the H.261 recommendation, at the input of the coder, source image blocks are made up of an 8×8 group of samples.

The coding algorithm computes for each block of 8×8 pixels, its DCT and quantization. Then the algorithm calculates the Inverse DCT (IDCT) and reverses the quantization process to recover what will become the *previous image* for the next inter pass.

In this way, the encoding and decoding processes for the codec can be summarised as follows:

For encoding:

- Perform movement detection for each block of the image
- Perform H.261 encoding of all *detected* blocks:
  - Interframe/Intraframe decision IF Interframe THEN:
    - compute the difference between the previous block and the new block
    - perform a DCT of this difference and then quantize the DCT results
    - perform an inverse quantization and inverse DCT then sum the result with the previous block and store this as the new *previous* block

    ELSE IF Intraframe THEN:
    - perform a DCT of the original image and then quantize the DCT results
    - perform an inverse quantization and inverse DCT then store the result as the *previous* block
  - perform H.261 encoding of the image, group of blocks and Macro Blocks using Huffman tables.

For decoding:

- If intraframe coding:
  - Inverse quantization, Inverse DCT of all blocks in the MB then store this as the *previous* image.
- If interframe coding:

  On a Macro Block by Macro Block basis:
  - If the Macro Block is encoded with a motion vector, compute and apply the motion vector to the Macro Block of the previous image.
  - If the Macro Block is not encoded with a motion vector:
    - perform an inverse quantization and an inverse DCT of the encoded blocks in the MB;
    - Sum the results with the previously stored block.
- Now display the image.

Points of Interest

The calculations are only effective for the blocks which have never been sufficiently modified. In consequence, the changes necessary for coding of an image are directly proportional to the quantity of new information which is to be carried. Additionally, the time of the transmission of coded data varies as a function of the movement in the particular sequence of images.

This situation is particularly irritating for the **real-time** scenario (such as a videoconference/videotelephoney, for which the original standard was conceived). As a result, the software must attempt to meet this requirement of independence of the duration of coding from the particular image type carried.

To meet this requirement a number of solutions are implementable:

- reduction of the number of blocks to calculate per image. For this to work, it is necessary to limit the detection of movement in accordance with the type of image. This parameter (the number of blocks per image) has the most direct effect on the duration of encoding required. If such a technique is used, one needs to make a pass of the entire image, (*preprocessing the image*) in order to detect the blocks that have changed most. This solution would not work if many of the blocks in an image had been changed simultaneously, as for example in a scene change.

- changing the quantization steps when a significant change to the image occurs. However, such a solution may not have the desired effect since this stage (quantization) is performed after the calculation of the inter-image difference and of the DCT.

- Another solution consists of only transmitting a number of the coefficients of the DCT in the situation where only some of the samples have changed. As a result, it is necessary that such changes are representative of changes to the whole block, then coefficients can be chosen which encode the low frequencies of the image. For this, it is also necessary to perform a preliminary pass over the image, but it is useless to try to use block detection since the values are the results of previous calculations.

  Figure 3.1 shows how smaller regions of the 16x16 DCT coefficients block is actually transmitted. For the 1×1 case only the first of the coefficients is transmitted, because this is the DC component, the result is a block consisting of only two brightness levels. For the 2×2 example, four DCT coefficients are transmitted, for the 3×3, 9 are transmitted. The full 8×8 (64) DCT coefficients are also transmitted as a reference.

  Using this technique, a study tried several size transforms: 8×8, 4×4, 3×3, 2×2 and 1×1. The considered opinion of the study was that 1×1, 2×2, and 3×3 did not transfer enough information to be of use. Table 3.1 shows results obtained for the test sequence *Miss America* (consisting of 71 frames) [28].

The length of the coding, in addition to the maximum number of bits for an image must be a parameter of coding modifiable on a decoder by decoder basis, based on the state of the communications network. In the case of congestion, the decoder can order the coder to reduce the rate of flow between the coder and it; in this way the best compromise between the length of coding and the rate of compression is sought.

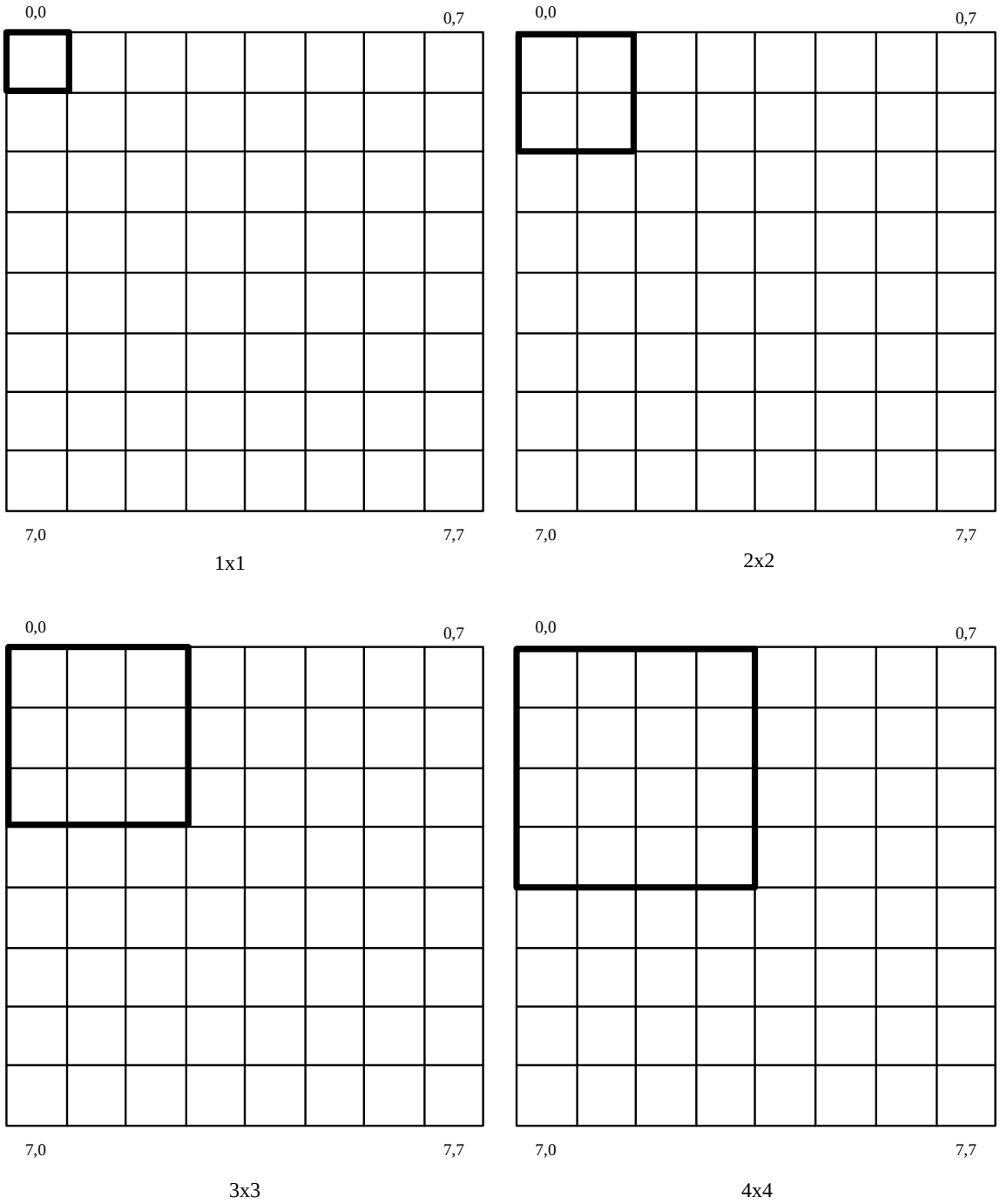


Figure 3.1: This diagram shows how only some of the 64 DCT coefficients are used. 1×1 (1 coefficient), 2×2 (4 coefficients), 3×3 (9 coefficients) and 4×4 (16 coefficients) examples are shown.

Table 3.1: Effects on coding due to number of DCT coefficients used

Number of Coefficients	**1x1**	**2x2**	**3x3**	**4x4**	**8x8**
Mean size of image (in octets)	680	1380	1890	2130	3050
Mean time to encode (in ms)	177	289	320	342	455
Mean rate of transfer (in image/s)	5.6	3.4	3.1	2.9	2.2
Reported S/N (in dB)	30.1	33.7	35.4	36.4	39.4

3.6 Transmission of packet video via TCP/IP

The h261_encode/decode codec is designed to exchange data over a TCP/IP based network; as a result, several extra considerations must be made for the peculiarities of the network being used.

The H.261 recommendation is for the transmission of H.261 over $p \times 64$ kbit/s data links. This is in stark contrast to the TCP/IP network where the actual data rate is dependent on the communications media carrying the video data. The communications carrier or TCP/IP could be as low as 9600 bit/s, 10 Mbit/s in *perfect* ethernet up to 100 Mbit/s in FDDI and even higher rates in the Wide Area Network systems. In this way TCP/IP runs as fast as it can, this speed is dynamic depending upon the particular load imposed on a communications link by other users.

This is the first major difference between the H.261 assumed communications network (n×ISDN channels) and TCP/IP (based on ethernet): ISDN assumes a set rate available (64kbit/s per channel), whereas the rate in TCP/IP might be as low as several kbit/s or as high as several Mbit/s; this is totally dependent on the load and capacity of a particular channel. As a result, the h261_decode/encode software must adjust its network consumption accordingly so as not to become a network bandwidth *pig*, consuming all the network to the detriment of other users.

The second major difference between the ISDN style of communications and TCP/IP based communications is that ISDN communications protocols assume a steady *stream* style connection, that is, all data put in one end of the communications connection is ensured of coming out the other end, unduplicated, unreplicated and in order.

In comparison, the basic Internet Protocol is based upon a *packet network* where data is packetized and sent through the network. Data is not sent in a continual stream, at the most basic level, reliability is not guaranteed, additionally there remains the chance of losing packets all together as well as receiving them out of order or even replicated.

The protocols in the Internet network (in this case the Transmission Control Protocol (TCP)) gives a reliable *pseudo-stream* of data, although data delay is a certainty on a busy network with the work required to make a particular stream of data reliable.

3.6.1 Locating the boundary between images

In order to increase the rate of the compressed data stream, H.261 utilizes modified Huffman encoding. In addition, in inter mode, only the differences between two successive images are coded. A transmitted image has not got a set length, in fact its size is not known until the transmission is complete. This is as a direct result of the non-octet alignment of the data transmitted over the Internet. As a result, the decoder only recognizes information at the end of an image when it receives the the next frames Picture Start Code (PCS).

The PCS is large enough to fill greater than one octet thus giving the (up to seven bits of) remains of the previous image a *push* through the channel to the decoder.

As a result if the incoming image is to be displayed as soon as the last Macro Block is decoded, it is necessary to contain the PCS of the following image in the same byte.

In TCP, the notion of bits does not exist since programs work with a stream of octets. Even padding out data to the size of a byte will not guarantee the moment at which TCP will transmit the data stream.

The result to the end user is at least one frame lag, as well as the possibility that after receiving a set sequence of images the last image will not be displayed (either, until the user quits the program or another image is received).

3.6.2 Controlling the Rate of Flow

There is a need to control the rate of data flow out of the codec to enable time to react in case of congestion on the networks. The rate of flow is a function of the following principal factors:

- The frequency of the image sampling (adjustable by under-sampling)
- The number of allocated bits per image, determined by :
  - the step size of the quantizer (the step size of the quantizer is in inverse proportional to the quality of the image);
  - the setting of the threshold of block change which controls the decision of whether to code or not.

3.6.3 Choice of Quantizer

The choice of the quantizer step used depends upon the coefficients which have to be coded. It is obvious that large coefficient values may not be able to be coded with a quantizer that uses too small a step; a large coefficient corresponding to a quick change of scene. It is for this reason a *haze* is observed at the time of a scene change.

Figure 3.2 shows the data rate for the coding of the *Miss America* sequence for several different quantization values using the Turletti H.261 Codec. This figure is taken from the T. Turletti paper describing the codec[28].

For the first image the network is seen to be congested, if the send() operation *blocks* for an abnormally long amount of time (takes a long amount of time to operate), the step size of the quantizer could be increased, thereby reducing the amount of data to transmit. However, as a result of increasing the step size the image quality will be degraded. Using

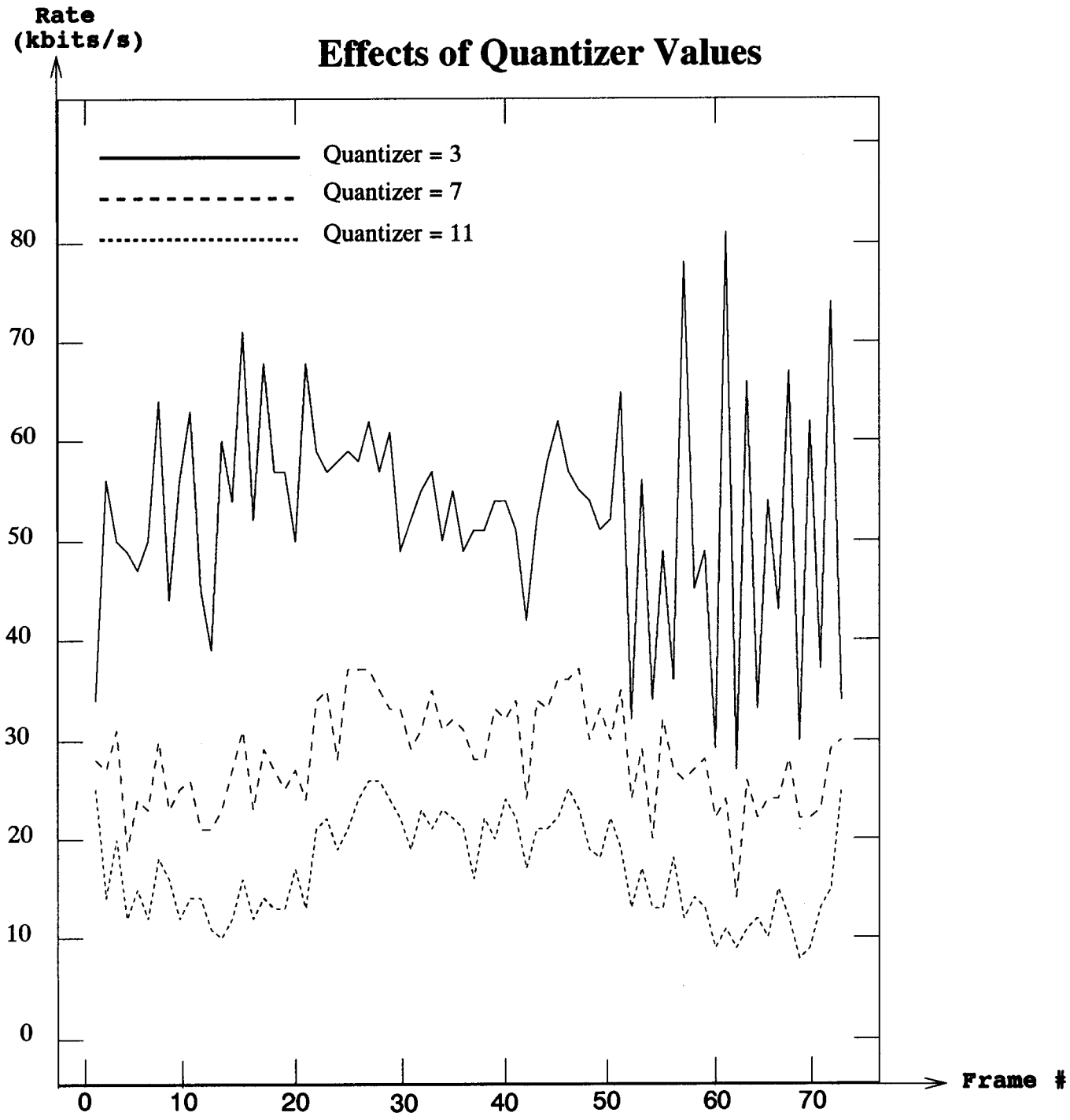


Figure 3.2: The effect of Quantizer step on data rate. Source: T. Turletti (1992) [28]

a rougher DCT (fewer coefficients per block) this will also increase throughput of the images; however, with a corresponding lowering of reconstructed image quality.

As a result, in an effort to achieve the best quality images, the task is to find the best balance between image quality and image rate.

3.7 Summary

From this implementation several items of interest develop:

- Motion compensation takes 2-3 longer than just using inter, thus if motion compensation is to be used, the threshold choice (on whether a block has changed sufficiently to require retransmission) is a compromise between encode time and sensitivity to movement.
- The time of transmission varies as a function of the movement in a particular image sequence.
- In order to achieve the best quality pictures, there is a balance between image quality and image rate.

In the operation of the encoder, the user does not have control of many of the functions of the codec, the user is only able to select between the privileging of *image rate* or *image quality.* In the second selection, the software attempts to select the best quantizer (for the available bandwidth.) For the privileging of rate, two changes can be made:

- if the network is congested, the quantization step is increased thereby decreasing the amount of data transferred;
- if there are a lot of blocks to encode, the use of a rougher DCT, while lowering image quality, will also lower the number of coefficients generated per block, and thus the amount of data transferred.

This coder is an interesting application of the H.261 system and the basis of sending Video over packet switch networks (and for modifications to the encoding system to be required for adequate results), will become more important with the deployment of larger infrastructures of ATM (Asynchronous Transfer Mode) communications which have at their heart packet switch technology.

Chapter 4

Conclusion

The H.261 recommendation forms an excellent introduction to the *world* of video coding and compression. Parts of this recommendation been incorporated into many of the current developments; and as such, H.261 gives a good insight into other developments.

The implementation is fascinating because it has to work in with Internet based networks. As a result, the implementation covers ways in which H.261 can be modified to suite this situation.

Chapter 5

Acknowledgements

I would like to extend my thanks to Dr Wu for giving his students enthusiasm in the subject material and for presenting many informative and dynamic sessions on Video Coding and Compression. I would also like to extend my thanks to Thierry Turletti for his assistance in my attempts to understand the software he has graciously made available, and his making available a number of reference papers used in the preparation of this paper.

Bibliography

[1] Ang P. H., Ruetz P. A., and Auld D., "Video compression makes big gains", *IEEE Spectrum*, vol. ?, pp. 16–19, October 1991.

[2] CCIR , "Recommendation 601-I Digital Methods of Transmitting Television Information", 1986.

[3] CCITT , *Recommendation H.261, Video codec for audiovisual services at p×64 kbit/s*, CCITT, 1990.

[4] Chan S. C. and Ho K. L., "A new two-dimensional fast cosine transform algorithm", *IEEE Transactions on Signal Processing*, vol. 39, pp. 481–485, February 1991.

[5] Clarke R. J., *Transform Coding of Images*, Academic Press, 24–28 Oval Rd., London NW1 7DX, 1985.

[6] Comer D., *Internetworking with TCP/IP Volume 1; Principles, Protocols and Architecture*, volume 2, Prentice-Hall International, Inc., April 1991.

[7] Gonzalez R. C. and Woods R. E., *Digital Image Processing*, Addison-Wesley, 1992.

[8] Haque M. A., "A two-dimensional fast cosine transform", *IEEE Transactions on Acoustic, Speech and Signal Processing*, vol. ASSP-33, pp. 1532–1539, December 1985.

[9] Horgan J., "Claude E. Shannon", *IEEE Spectrum*, vol. , pp. 72–75, April 1992.

[10] IEEE , "Special Issue on ISDN", *IEEE Communications Magazine*, vol. 28, , April 1990.

[11] ISI , "Internet protocol", *Request For Comment : 791*, vol. ?, , September 1981, Writtern in association with DARPA.

[12] Jain A. K., *Fundamentals of Digital Image Procesing*, Prentice-Hall, 1989.

[13] Jain J. R. and Jain A. K., "Displacement Measurement and Its Application in Interframe Image Coding", *IEEE Transactions on Communications*, vol. COM-29, pp. 1799–1808, 1981.

[14] Jayant N., "Signal Compression: Technology Targets and Research Directions", *IEEE Journal on Selected areas in Communications*, vol. 10, pp. 796–818, June 1992.

[15] le Gall D., "Mpeg : A video compression standard for multimedia applications", *Communications of the ACM*, vol. 34, pp. 47–58, April 1991.

[16] Liou M., "Overview of the p*64 kbit/s video coding standard", *Communication of the ACM*, vol. 34, pp. 60–63, April 1991.

[17] Netravali A. N. and Stuller J. O., "Motion compensated transform coding", *Bell Systems Technical Journal*, vol. 88, pp. 1703–1718, September 1979.

[18] Peterson W. W. and Weldon Jr E. J., *Error Correcting Codes*, MIT Press, 1972, Appendices C and D are useful for finding $g(x)$ for the binary BCH codes.

[19] Quercia V. and O'Reilly T., *X Window System Users Guide for X11 R3 and R4*, O'Reilly & Associates, Inc., Sebastapol, CA, May 1990.

[20] Rao K. R. and Yip P., *Discrete Cosine Transform*, Academic Press, Inc, 1990.

[21] Ritchie D. and Thompson K., "The UNIX Time-Sharing System", *Communications of the ACM*, vol. 17, pp. 365–375, July 1974.

[22] Rocca F. and Zanoletti S., "Bandwidth reduction via movement compensation on a model of the random video process", *IEEE Transactions on Communications*, vol. COM-20, pp. 960–965, October 1972.

[23] Scheifler R. and Gettys J., "The X Window System", *ACM Transactions on Graphics*, vol. 5, pp. 79–109, April 1986.

[24] Sobell M. G., *A Practical Guide to the UNIX System V*, Benjamin Cummings, Menlo Park, CA, 1985.

[25] Stix G., "Encoding the "Neatness" of Ones and Zeros", *Scientific American*, vol. , pp. 27–28, September 1991.

[26] Turletti T., "H.261 Codec", 1992, Available using anonymous ftp from avahi.inria.fr:/pub/h261/h261.tar.Z.

[27] Turletti T., "INRIA videoconference system", 1992, Available using anonymous ftp from avahi.inria.fr:/pub/videoconference/ivs.tar.Z.

[28] Turletti T., "Logiciel de codeur-décodeur H.261", September 1992, Paper as yet only published in French, as part of INRIA Project Rodeo.

[29] Wu H., "RDT4630 Video Coding and Compression", July 1992, Syllabus guidelines for Video Coding and Compression (RDT4630), Department of Robitics and Digital Technology, Faculty of Computing and Information Systems, Monash University.

[30] Wu H., "RDT4630 Video Coding and Compression: Practical Work Instruction Notes", August 1992, Submission guidelines for Video Coding and Compression (RDT4630), Department of Robitics and Digital Technology, Faculty of Computing and Information Systems, Monash University.

[31] Wu H., "RDT4630 Video Coding and Compression: Subject Notes", July 1992, Course notes for the subject: Video Coding and Compression (RDT4630), Department of Robitics and Digital Technology, Faculty of Computing and Information Systems, Monash University.