Motion Compensated Interpolation for Subband Coding of Moving Images

by
Mark Daniel Polomski

B.E., Electrical Engineering
SUNY at Stony Brook (1987)

Submitted to the Department of
Electrical Engineering and Computer Science
in Partial Fulfillment of the
Requirements for the
Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

October 1993

Signature of Author: _____
Department of Electrical Engineering and Computer Science
October 4, 1993

Certified by: Oct 4, 93
Bernd Girod
Assistant Professor, Media Laboratory
Thesis Supervisor

Accepted by: _____
Professor F. R. Morgenthaler
Chairman, Committee on Graduate Students

Motion Compensated Interpolation for Subband Coding of Moving Images

by

Mark Daniel Polomski

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 1990 in partial fulfillment of the
requirements for the Degree of
Master of Science in Electrical Engineering

ABSTRACT

With emergence of multimedia, CD-ROMS, and high speed networks there is a need for reducing the data rate for transmission and storage of images without severe loss of quality. An image compression system is presented based on motion compensated interpolation using multiframe matching techniques and subband analysis of keyframes and residual (error) frames, followed by vector quantization and arithmetic coding.

Multiframe matching differs from conventional block matching in that the intermediate frames are used in addition to the keyframes for motion estimation. Multiframe matching significantly improves the efficiency of motion compensated interpolation.

Keyframes and residuals are coded using a quad-split pyramid. The subbands are then coded using variance-normalized, error limited vector quantization. The vector fields, vector quantization codebooks, and look up tables are losslessly coded using arithmetic coding.

In computer simulations, bit rates below 2.5 megabits per second are achieved with high quality reconstruction at the receiver. All simulation computations are carried out using integer precision arithmetic.

Thesis Advisor: Bernd Girod
Assistant Professor, Media Laboratory

Contents

1 Introduction 1

1.1 Thesis Overview 1

2 Background 3

2.1 Low Bit-Rate Image Coding 3

2.2 Human Visual System 5

2.3 Transform Coding 9

2.3.1 Subband Coding 11

2.3.2 DCT Coding 12

2.4 Interframe Coding 13

2.5 Motion Compensated Coding 15

2.5.1 Motion Compensated Prediction 15

2.5.2 Motion Compensated Interpolation 16

2.5.3 Motion Estimation 17

2.6 Vector Quantization 20

2.6.1 Codebook Determination 21

2.6.2 Bit Allocation 22

2.7 Entropy Coding 23
2.7.1 Huffman Coding 23
2.7.2 Arithmetic Coding 24
2.8 Color Coding 26

3 Motion Compensated Interpolation: Theory and Limitations 28
3.1 Mean Square Estimation 28
3.2 Image Models 31

4 Motion Estimation 41
4.1 Block Matching 41
4.2 Multiframe Matching 43
4.3 Hierarchical Multiframe Matching 45
4.4 Prefiltering 50

5 Coding System for High Quality/Low Bit Rate Coding of Motion Sequences 52
5.1 Overview 52
5.2 Preprocessing 57
5.3 Encoder 57
5.3.1 Motion Estimator 58
5.3.2 Keyframe Encoder 58
5.3.3 Motion Compensated Interpolator 58
5.4 Decoder 59
5.4.1 Chrominance Decoding 62
5.4.2 Postprocessing 62

6 Computer Simulations **63**

6.1 Preprocessing 63

6.2 Motion Estimation 64

6.2.1 Segmentation 65

6.2.2 Prefiltering 67

6.2.3 Displacement Search 67

6.3 Keyframe Coding 68

6.4 Calculating Residual Frames 78

6.5 Residual Coding 79

6.6 Arithmetic Coding 86

6.7 Chrominance Coding 87

6.8 Decoding 90

6.9 Postprocessing 90

6.10 Results 90

7 Conclusions **101**

A Computational Complexity and Hardware Implementation **103**

B Filter Coefficients **105**

C Acknowledgments **107**

Bibliography **108**

List of Figures

2.1 Line visibility thresholds as a function of distance from three luminance edges; Contrast ratios $\frac{L_B}{L_D} = 27.5, 8.7, 2.75$. (from Netravali and Haskell [63]) 6

2.2 Contrast sensitivity of the Human Visual System. ($\circ$ = 1 Hz. $\bullet$ = 6 Hz. $\triangle$ = 16 Hz. $\blacktriangle$ = 22 Hz.) (From Netravali and Haskell [63]) 7

2.3 Contrast sensitivity of the Human Visual System. ($\circ$ = 5 cpd $\bullet$ = 4 cpd $\triangle$ = 16 cpd $\blacktriangle$ = 22 cpd) (From Netravali and Haskell [63]) 8

2.4 Modulation frequency needed to keep a flickering light of different frequencies at flicker fusion threshold. (from Kelly [43]) (9300 trolands is approximately the brightness of a CRT display) 10

2.5 Zig-Zag scanning order of DCT coefficients 14

2.6 Fixed interpolation using four-way weighted average 14

2.7 Motion compensated interpolator 16

2.8 Block Matching. Frames n and $n+3$ are keyframes. Data from the keyframes only are used in the pattern matching 18

2.9 An example of building a Huffman code 25

3.1 An isotropic two dimensional autocorrelation function 32

3.2 Measured and modeled mean square error using autocorrelation function $\mathbf{R}(x,y) = \sigma^2\alpha^{-a(x^2+y^2)^{\frac{1}{2}}}$ 34

3.3 Measured vs. modeled mean square error using autocorrelation function $\mathbf{R}(x,y,\tau) = \sigma^2(\tau)\alpha^{-a(x^2+y^2)^{\frac{1}{2}}}$. The model is weighted by the percentage of area of each frame in motion 37

3.4 Measured normalized mean square error using coded and uncoded keyframes in *Alley* sequence 38

4.1 Two dimensional logarithmic search. 42

4.2 Conjugate direction search. 43

4.3 Three step search. 44

4.4 Example showing how a single displacement vector is used for each Three-dimensional volume. 45

4.5 Comparison of Multiframe matching vs. Block matching with frame spacing = 2, with and without pre-filtering 46

4.6 Comparison of Multiframe matching vs. Block matching with frame spacing = 3, with and without pre-filtering 47

4.7 Comparison of Multiframe matching vs. Block matching with frame spacing = 4, with and without pre-filtering 48

4.8 Comparison of Multiframe matching vs. Block matching with frame spacing = 5, with and without pre-filtering 49

4.9 Hierarchical multiframe matching displacement estimator 50

5.1 Motion compensated interpolation system coder 53

5.2 Motion compensated interpolation system decoder 54

5.3 Multiframe Matching. Frames n and $n+3$ are keyframes. Data from the keyframes and intermediate frames are used in the pattern matching. 55

5.4 K-step search pattern 56

5.5 Block diagram of keyframe encoder. 59

5.6 Block diagram of pyramid analysis QMF filters. 60

5.7 Block diagram of pyramid synthesis QMF filters. 61

6.1 Graphical illustration of the percentage of the frame area classified as non–moving in *Alley* sequence. 66

6.2 Displacement vectors for *Alley* sequence frames 1 to 3 69

6.3 Displacement vectors for *Alley* sequence frames 25 to 27 70

6.4 Displacement vectors for *Alley* sequence frames 49 to 51 71

6.5 Displacement vectors for *Alley* sequence frames 73 to 75 72

6.6 Displacement vectors for *Alley* sequence frames 96 to 99 73

6.7 Lowpass filter used both horizontally and vertically in subband pyramid analysis . . 74

6.8 Highpass filter used both horizontally and vertically in subband pyramid analysis . . 75

6.9 A typical keyframe pyramid decomposition from the *Alley* sequence 76

6.10 Interpolated intermediate frame pels from bilinearly interpolated pels in the keyframes 78

6.11 Interpolated recreation of frame 1 of the *Alley* sequence 80

6.12 Residual from frame 1 of the *Alley* sequence 81

6.13 Interpolated recreation of frame 25 of the *Alley* sequence 82

6.14 Residual from frame 25 of the *Alley* sequence 83

6.15 Interpolated recreation of frame 49 of the *Alley* sequence 84

6.16 Residual from frame 49 of the *Alley* sequence 85

6.17 Gaussian decimation filter used in chrominance coding 88

6.18 Gaussian interpolation filter used in chrominance coding 89

6.19 Lowpass filter used both horizontally and vertically in subband pyramid synthesis . 91

6.20 Highpass filter used both horizontally and vertically in subband pyramid synthesis . 92

6.21 Luminance and Chrominance PSNR for *Alley sequence.* 98

List of Tables

3.1 Percentage of area classified non-moving in *Alley* sequence 36

6.1 Parameters used in the multifield matching motion estimation search. 65

6.2 Error limits used in coding the keyframes for *Alley* 77

6.3 Error limits used in coding the residuals for *Alley* 77

6.4 Comparison of the average energy of keyframes and residuals in *Alley*. 79

6.5 Error limits used in chrominance coding. 88

6.6 Channel bit rate for *Alley* sequence frames $0 \rightarrow 20$ 93

6.7 Channel bit rate for *Alley* sequence frames $21 \rightarrow 44$ 94

6.8 Channel bit rate for *Alley* sequence, frames $45 \rightarrow 68$ 95

6.9 Channel bit rate for *Alley* sequence frames $69 \rightarrow 92$ 96

6.10 Channel bit rate for *Alley* sequence frames $93 \rightarrow 116$ 97

6.11 Displacement vector bit rates for *Alley* sequence 97

6.12 Channel bit rate for *Alley* sequence Chrominance 99

6.13 Fraction bandwidth allocation for *Alley* simulation 99

A.1 Worst-case comparison operations for a multiframe search as used in the computer simulations on *Alley* sequence using 4800 vectors per four frame set. 103

A.2 Worst–case comparison operations for block search in a motion compensated predictive system using ±7 pel search area, using 4800 vectors per four frame set 104

B.1 QMF analysis and synthesis filters used in pyramid decomposition 105

B.2 Gaussian Decimation / Interpolation Filters used in chrominance coding 106

Chapter 1

Introduction

1.1 Thesis Overview

The organization of this thesis follows: Chapter 2 presents background material on image compression techniques. Chapter 3 presents theory on the performance of motion compensated interpolation on noise-free keyframes, as well as the effects of coding noise in the keyframes and on the performance of motion compensated interpolation. A discussion of motion estimation techniques follows in Chapter 4.

A coding system for high quality image transmission or storage is presented in Chapter 5. This system incorporates many of the image compression techniques discussed in the previous chapters. Motion compensated interpolation is the heart of the coding system. Keyframes and residuals are coded using a pyramid decomposition which is vector quantized. The output of the vector quantization is adaptively followed by arithmetic coding. The lowest frequency subbands are predictive coded before arithmetic coding.

Chapter 6 presents the results of computer simulation of the coding system and the parameters

used to obtain those results. Conclusions are presented in Chapter 7.

Three appendices follow: Appendix A presents calculations of the computational complexities associated with motion estimation and motion compensated interpolation. Appendix B lists the values of filter coefficients used in the simulations. And finally, Appendix C thanks the many people responsible for helping me in this endeavor.

Chapter 2

Background

2.1 Low Bit-Rate Image Coding

Recently there has been interest in low bit rate image coding for both transmission and storage of images [3, 52, 68]. Image data reduction enables motion pictures to stored and manipulated by personal computers and transmitted over cost effective links, such as T1 (1.5Mbit/sec) telephone lines or standard computer networks. Such compression may enable movies on demand or multimedia workstations where image audio and other data types are easily manipulated on a computer screen[94].

The basic goal of low bit-rate coding is to reduce the amount of data necessary for transmission or storage while maintaining acceptable image quality. The problem is twofold: First, by exploiting psychovisual properties of the human visual system (HVS) *irrelevant* information is removed. Secondly, the statistical correlations between the data are exploited to remove *redundant* information. However, limitations of the channel bandwidth or storage device may require further reduction in data resulting in noticeable loss of quality.

*Inter*frame techniques exploit the redundancies between frames in a sequence of images for data reduction. *Intra*frame coding uses the redundancies within a single frame for data reduction.

Some standards for image coding are currently emerging. The Joint Photographic Experts Group (JPEG) standard for coding still images [97] is based on the Discrete Cosine Transform (DCT) with the coefficients coded using Huffman coding or arithmetic coding.

Px64, a teleconferencing standard for real time coding sequences of moving images, uses motion compensated prediction for interframe coding, with the prediction residual error coded using DCT's, which are quantized and Huffman coded. As the intended use is teleconferencing, this standard supports relatively low resolution at very low bit rates. Px64 operates in channels with multiples of 64 Kbits per second. A similar coding standard, CCITT Nx384, operates in multiples of 384 Kbits per second. This coding is similar to px64, as both use DCT and motion compensation. The higher bit rates allow higher quality than the teleconferencing standard, but this is still a sequential access system; that is, all frames from the starting frame must be decoded to reach any one frame.

The Motion Pictures Experts Group (MPEG) is developing a standard for high quality, low bit-rate coding of image sequences. The current proposals use motion compensated prediction or interpolation followed by DCT coding of the residuals to obtain compression. One of the requirements for MPEG is a low seek latency to allow random access, making it suitable for interactive use.

Some traits that are desirable for an interactive motion picture coding system are:

- Scalability. It is desirable to be able to trade off spatial or temporal resolution versus bit-rate, or spatial resolution versus temporal resolution. This may enable a user to view multiple sequences at a reduced resolution. A network movie server would then be able to reduce spatiotemporal resolution if the network use becomes too high. Temporal Scalability allows outputs at variable frame rates.

- Random Access. In data storage and retrieval systems random access with a reasonable latency

is desired. Purely predictive systems require the recreation of all previous frames before the desired frame may be displayed. For this reason, fast forward is difficult, requiring decoding to be performed at rates higher than the basic frame rate. Also, reverse replay requires large numbers of frame stores in predictive schemes.

2.2 Human Visual System

The final receiver of an image coding system is the human eye. For this reason, an understanding of the HVS is important in image compression [65, 73]. It is this understanding which enables us to eliminate irrelevant information while maintaining acceptable picture quality.

One important aspect of the HVS is masking (reduction in the visibility of stimuli) due to a complex non-uniform background. Spatial masking occurs when there is a large change in luminance (eg. an edge) causing reduced visibility on both sides of the edge. This masking effect is more pronounced for edges of high contrast [63]. Figure 2.1 shows line visibility as a function of distance from edges.

This masking is taken advantage of in Differential Pulse Code Modulation (DPCM) quantizers, which use unequal quantization steps. Large transitions can be coarsely quantized due to this elevation of the visibility threshold.

Spatial contrast sensitivity varies with spatial and temporal frequency, but is also a function of contrast and viewing distance. The dependence on viewing distance can be eliminated if the frequency is expressed in cycles per degree subtended. Figure 2.2 shows spatial contrast sensitivity vs. spatial frequency for several values of temporal frequency, and Figure 2.3 shows temporal frequency contrast sensitivity vs. temporal frequency for several spatial frequencies.

Some important points in the context of image coding are:

- Contrast Sensitivity falls off at higher frequencies; the peak sensitivity is around 4 cycles/degree.

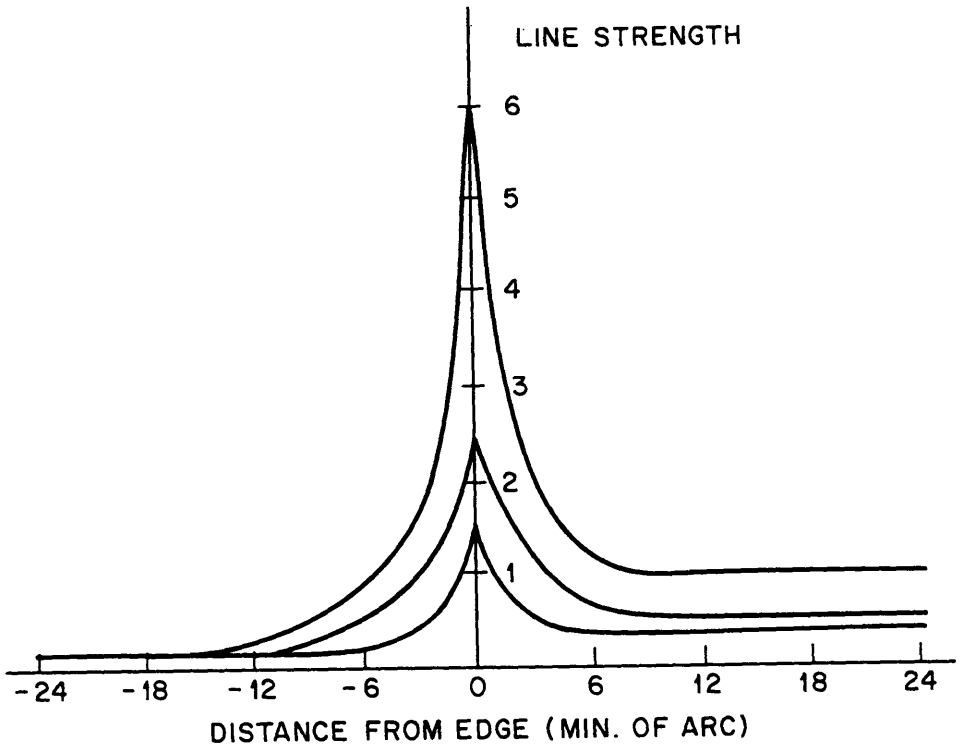

Figure 2.1: Line visibility thresholds as a function of distance from three luminance edges; Contrast ratios $\frac{L_B}{L_D} = 27.5, 8.7, 2.75$. (from Netravali and Haskell [63])

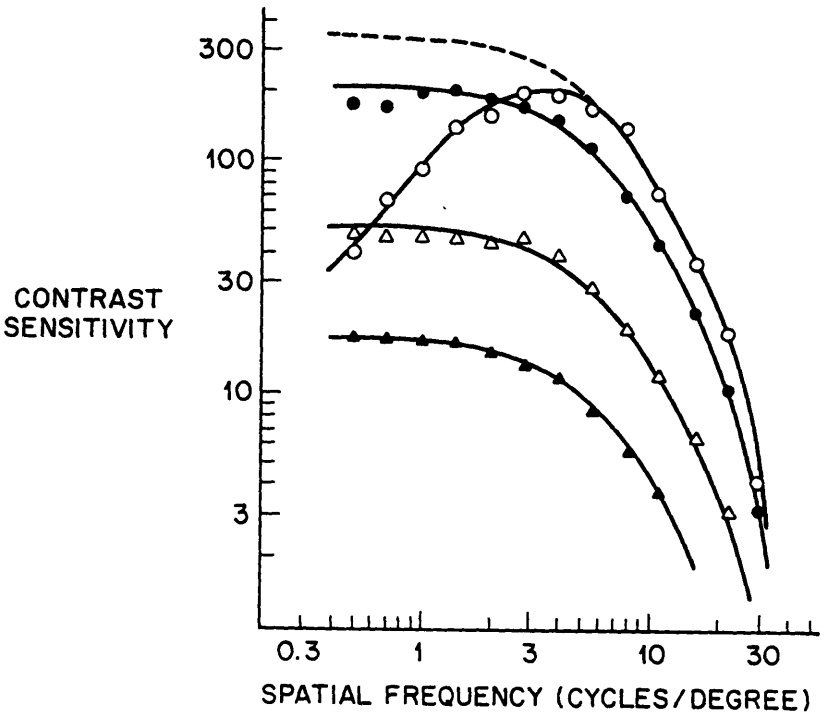

Figure 2.2: Contrast sensitivity of the Human Visual System. (o = 1 Hz. • = 6 Hz. △ = 16 Hz. ▲ = 22 Hz.) (From Netravali and Haskell [63])

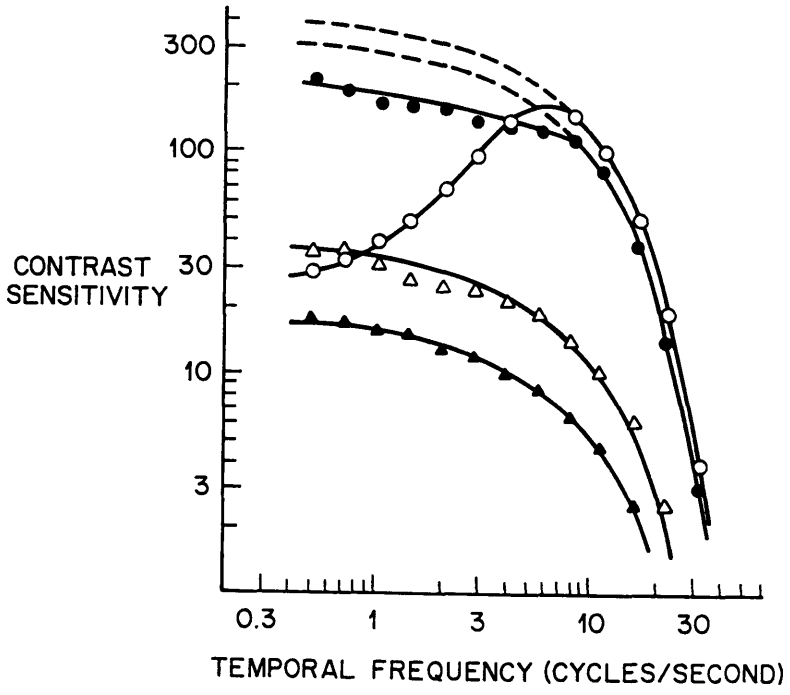

Figure 2.3: Contrast sensitivity of the Human Visual System. (o = 5 cpd • = 4 cpd △ = 16 cpd ▲ = 22 cpd) (From Netravali and Haskell [63])

- Horizontal and vertical orientations have the greatest contrast sensitivity.

Interframe coding can take advantage of temporal masking and perception. Masking effect will depend on whether the eye is tracking the movement or not. Humans have higher acuity to tracked objects, while randomly moving objects which are untracked have significant reduction in perceived resolution. It is important that tracked moving areas, especially those with edges, do not have appreciable noise added by an image coding process [28].

In the case of drastic movement, which the eye cannot track, the perceived spatial resolution is lowered. Scene changes can be coded at reduced resolution, provided that resolution is gradually restored. Unfortunately it is impossible to know apriori whether or not viewers of an image will be tracking moving areas, thus making this property difficult to use in a practical system.

At lower temporal frequencies the HVS is very sensitive to flickering. The sensitivity to flickering also depends on the luminance level and the ambient luminance. At the brightness level of a typical Cathode Ray Tube (CRT) display, the sensitivity to flicker is at a maximum around 15 Hz. (Figure 2.4). Above 70 Hz. flickering is essentially imperceptible. At lower frequencies the sensitivity falls off considerably. The flicker threshold is an important consideration in compression systems where the coding method varies in time, eg. keyframes inserted in a predictive system, interframe interpolation, and systems with reduced chrominance refresh rate.

2.3 Transform Coding

Transform coding uses filtering techniques to achieve data compaction [58, 98] by transforming data from one domain to another, eg. space to frequency, or by dividing the data into different frequency regions. Transform coding does not achieve data compression in itself, but merely converts the data to a more convenient form for bit allocation. The transformed output data is then reduced using methods such as s runlength coding, Differential Pulse Code Modulation (DPCM), quantization,

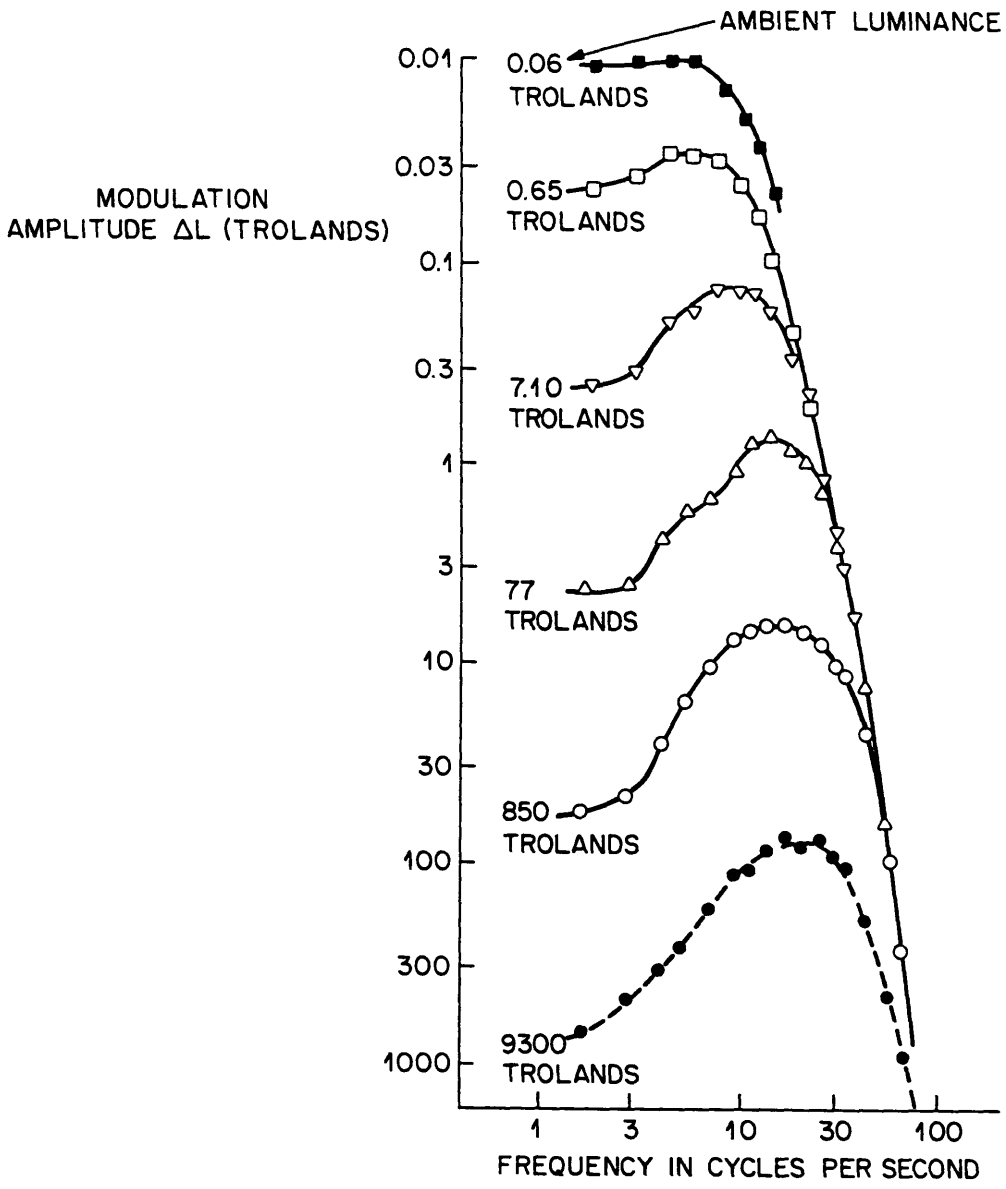

Figure 2.4: Modulation frequency needed to keep a flickering light of different frequencies at flicker fusion threshold. (from Kelly [43]) (9300 trolands is approximately the brightness of a CRT display)

and entropy coding.

Block transforms such as the Discrete Cosine Transform (DCT), Discrete Fourier Transform (DFT), and Hadamard and Karhunen-Loeve transforms operates on a single block of data at a time. Overlapped transforms, such as subband decompositions, use data from neighboring samples to separate the source into different frequency bands.

2.3.1 Subband Coding

Subband coding is a method in use for data compression of images [14, 15, 21, 42, 87, 90, 92, 101, 102]. In subband coding an input sequence is split up into spatial or spatiotemporal frequency subbands via filtering, often using quadrature mirror filters (QMF's). By exploiting the different probability distributions within the subbands [17, 100] and the importance of each subband relative to the acuity of the HVS, the available bandwidth can then be efficiently allocated by a coder. For example, the HVS is more sensitive to the low spatial frequency band than the highest spatial frequency band [16], and the HVS spatial frequency response to diagonals is lower than the response to horizontals and verticals [82]. Thus, fewer bits per picture element (pel) can be spent on certain of the frequency bands. Other bands which contain more important information relative to the response of the HVS should be carefully coded.

It is interesting to note that similar conclusions about the relative importance of the various frequency bands can be obtained by energy considerations. The highest frequency bands contain little average energy per sample. Coding each band to equal distortion criterion will allocate bits roughly in proportion to the frequency band's importance to the HVS.

In low bit-rate image transmission systems, these subbands are then coded using a redundancy reduction technique, such as vector quantization (VQ) [14, 15, 80, 92, 101].

2.3.2 DCT Coding

The DCT, a linear orthonormal transform, is the most common block transform used in image coding [61]. DCT coding has been used on source images [2, 40, 95], residuals [4, 22, 47, 55], and subbands. The input image is tiled into non-overlapping NxN blocks and each block is separately transformed. The DCT transform method can approach the efficiencies of Karhunen-Loeve transform [63] at a much lower level of computations.

The transform may be performed by matrix multiplication

$$C = T S T' \tag{2.1}$$

where C is the transformed output (coefficients) T is the DCT basis matrix, and S is the source block (matrix). The basis matrix is given by

$$\begin{aligned} t_{i,j} &= \sqrt{\frac{2-\delta_{i-1}}{N}} \cos\left[\frac{\pi}{N}\left(j-\frac{1}{2}\right)(i-1)\right] \\ i,j &= 1 \ldots N \\ \delta_p &= \begin{cases} 1 & \text{if } p = 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.2}$$

The inverse (IDCT) is found by

$$C = T' S T \tag{2.3}$$

Fast discrete cosine (FCT) algorithms have been developed [53] which use on the order of $N \log N$ operations versus the N^2 for direct computation. Typical block sizes used are 8x8, 4x4 and 16x16.

The DCT in itself does not give any data compression, but it compacts the energy into a few frequency bands which can then be more efficiently quantized. Most often the coefficients are scanned

in a zig-zag fashion (Figure 2.5) before quantization. The highest frequency coefficients very often are zero or quantized to zero, making the zig-zag scan pattern efficient for run length coding of the quantized coefficients. It is interesting to note that DCTs can be considered to be subband decompositions acting on a single block of data partitioning the data into equal sized bands.

One drawback of block transforms is that discontinuities can occur at block boundaries. If the higher frequency samples are too coarsely quantized this "blockiness" will be apparent. A recent method to overcome this blocking effect is the Lapped Orthogonal Transform (LOT) [54]. The image is divided into slightly overlapping blocks before the transform is applied. This results in fewer visible edge artifacts at equivalent data rates to the DCT.

2.4 Interframe Coding

Interpolative coding can be divided into fixed and adaptive methods. In fixed interpolation, a predetermined set of pels is used to recreate the missing pels. For example, a four way weighted average of alternate fields (Figure 2.6) is one possible method of fixed interpolation. More sophisticated methods such as cubic splines or higher order polynomials could be used, but it has been reported [61, 63] that not much is gained in error reduction at the cost of greater complexity by using more complicated schemes. One drawback in fixed interpolation coding is the spatial blurring [30] (reduced spatial and temporal resolution) that occurs when moving objects are present in the image sequence due to the low-pass filtering effect of the fixed interpolation filter.

Adaptive interpolation uses varying rules to select which pels in the keyframe are used as the source for the interpolated pels in an attempt to minimize the error in the interpolated pels. One such adaptive method that has shown potential is motion compensated interpolation (MCI) [13, 30, 44, 48, 55, 70]. In this method motion vectors representing the translational motion of objects determine which pels are to be used in the interpolation. Spatial resolution (sharpness of the

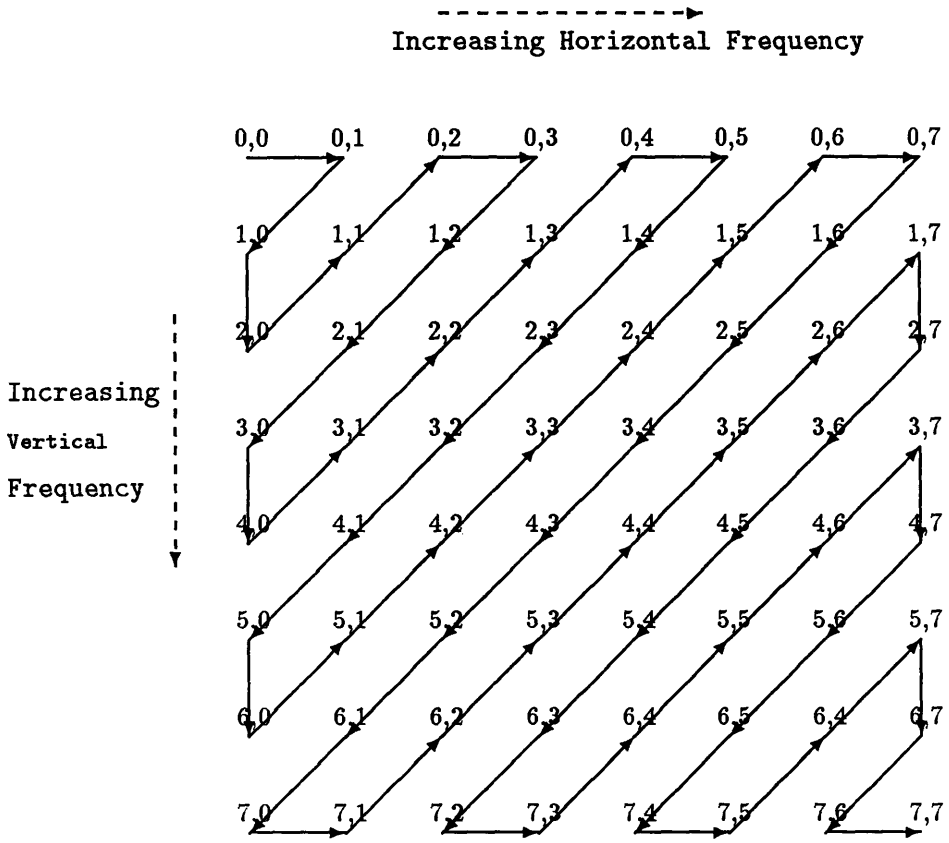

Figure 2.5: Zig-Zag scanning order of DCT coefficients

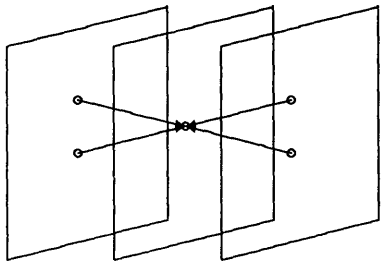

Figure 2.6: Fixed interpolation using four-way weighted average

interpolated image) can be improved using MCI vs. non-adaptive interpolation.

2.5 Motion Compensated Coding

The class of motion adaptive techniques is another method that has been applied to image coding [55, 61, 62, 79]. Both motion compensated interpolation (MCI) [13, 30, 44, 46, 48, 55] and motion compensated prediction (MCP) [25, 38, 45, 64, 69, 89] have been used for image compression. In an interpolative coding scheme, a subset of the picture elements (keyframes) are transmitted to the receiver [63]. The pels that were not selected for transmission are reproduced at the receiver using displacement information determined from the keyframe images at the receiver, or sent via side information from the transmitter. In an extrapolative or predictive coding scheme, pels in the immediate future are extrapolated from past pels [25, 29, 56, 75, 81, 89]. Margoudakis reports in a comparison of MCI and MCP that the motion compensated interpolation scheme was more stable and subjectively preferred [55].

2.5.1 Motion Compensated Prediction

Motion compensated prediction uses the previous frame or frames to predict future ones. This technique is most commonly used for sequential linear playback applications, such as teleconferencing. Since each frame uses information from all of the previous frames, this technique is quite efficient. MCP is better suited for sequential access than random access applications, although some coding systems use MCP with keyframes inserted at regular intervals to allow some random access. A predictive system is essentially a sequential access system that would require the recreation of all preceding frames up to the desired frame making access time prohibitively large.

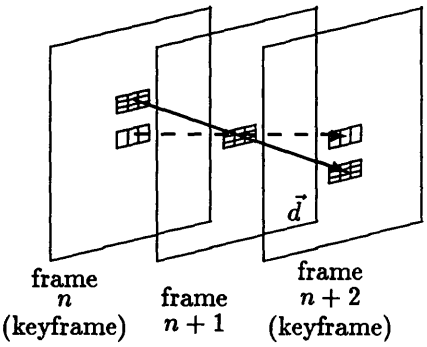

Figure 2.7: Motion compensated interpolator

2.5.2 Motion Compensated Interpolation

Random access is an important feature for many image storage and transmission systems [41]. Using MCI rather than MCP allows recreation of selected frames after retrieving only two key frames and their associated vector displacement fields. In addition, MCP is suitable for bi-directional replay.

Figure 2.7 shows an example of motion compensated and fixed interpolation. The keyframes, frames n and $n+2$, are transmitted; frame $n+1$ is to be reproduced by interpolation. Usually one displacement vector is transmitted for a block of pels. This is expanded to provide a displacement vector for each pel in the image. The pel at (x_1, y_1) in frame $n+1$ is interpolated using pels $(x_1 - dx/2, y_1 - dy/2)$ in frame n and $(x_1 + dx/2, y_1 + dy/2)$ in frame $n+2$, as indicated by the solid arrow in Figure 2.7. If simple linear interpolation were used, the pel at (x_1, y_1) in frame $n+1$ is interpolated using pels (x_1, y_1) in frame n and (x_1, y_1) in frame $n+2$, as indicated by the dotted arrow in Figure 2.7. This has the effect of introducing motion blur since pels from the background and the moving object are averaged together.

It is important to note that the transmitted keyframes available at the receiver contain noise from the coding process. However, the reconstruction is completely known at the transmitter, so an error signal (residual) can be sent to achieve the best reconstruction for a given bit rate.

For a motion compensated interpolation coder to be successful, the most important factor is an accurate estimate of the displacement [25, 30, 48, 63]. In addition, in the case of complex translational motion, rotational motion, or uncovering background, additional side information (an error signal) may need to be transmitted in order to maintain acceptable quality [38, 63].

2.5.3 Motion Estimation

Several methods for motion estimation are in general use: "pel recursive" [63], pattern matching [30, 38, 39], transform domain techniques[29, 32, 25], and gradient matching [49]. Another method that has been used to determine optical flow lines is spatiotemporal filtering[36].

Most of these methods assume the following [39, 61, 63, 71]:

1. Illumination is uniform both spatially and temporally.

2. Objects are moving in a plane parallel to the camera lens.

3. Uncovering of background and occlusion of one object by another is neglected.

Using these assumptions, the monochromatic intensities $Y(\vec{z},t)$ and $Y(\vec{z},t-\tau)$ are related by

$$Y(\vec{z},t) = Y(\vec{z}-\vec{d},t-\tau) \tag{2.4}$$

where $\vec{z}$ is the two dimensional vector of the spatial position within the frame, $\vec{d}$ is the two dimensional translation vector, and τ is the time difference between the frames. Thus, in real image sequences an estimate of the intermediate frame $\hat{Y}(\vec{z},\tau)$ can be made from two keyframes by interpolating:

$$\hat{Y}(\vec{z},t) = Y(\vec{z}-\vec{d_0},t-\tau)W_0 + Y(\vec{z}=\vec{d_1},t+\tau)W_1 \tag{2.5}$$

where $\vec{d_0}$ is the displacement vector from the first keyframe to the intermediate frame, and $\vec{d_1}$ is the

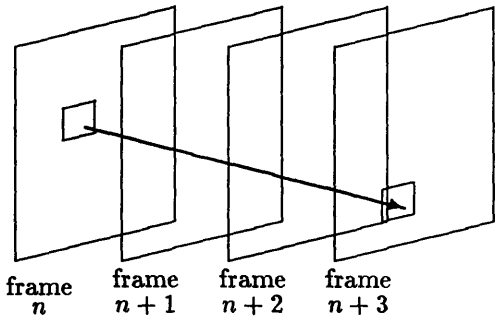

Figure 2.8: Block Matching. Frames n and $n+3$ are keyframes. Data from the keyframes only are used in the pattern matching

displacement vector from the intermediate frame to the second keyframe. W_0 and W_1 are weights such that $W_0+W_1=1$. Thus the problem is to estimate $\vec{d}$ from the intensities of consecutive frames.

The pel recursive algorithm is based on recursively minimizing the motion estimation error based on a steepest decent algorithm on the two dimensional intensity gradient [63, 69]. The accuracy of the motion estimate relies on many iterations of the algorithm which can be computationally expensive.

In pattern or block matching algorithms, the assumption is made that there is constant displacement within small two dimensional blocks [39, 103]. The image sequence is modeled as a wide-sense stationary stochastic process – the best estimate $\hat{\vec{d}}$ is an attempt to find the maximum of the cross correlation of pels in the second frame with a reference pel in the first frame. The displacement $\vec{d}$ is estimated over samples in a search area, using matching techniques so as to minimize some measure of the prediction error

$$E(\vec{d}) = \sum_{\vec{z}\in\mathbf{B}} DFD\left(Y(\vec{z},t), Y(\vec{z}+\vec{d},t+\tau)\right) \tag{2.6}$$

where $\mathbf{B}$ is the block of interest within the image, which is being shifted over the search area in the next frame. Figure 2.8 shows an example of block matching.

The difference DFD is some measure of the intensity difference between the pels, such as squared

difference or absolute difference. The precise definition of DFD has little effect on the accuracy of the estimate or length of search [63]. Generally an absolute error is preferred over the computationally more expensive squared error function. The accuracy of $\vec{d}$ is limited to a distance of 1 pel. This could be improved by interpolating between neighboring pels in the source images [27, 29, 25]. Other methods to improve accuracy include segmentation into moving and non-moving areas before the displacement is performed [71] (which also reduces the search processing time) and correction of invalid vectors [44] after the displacement estimation is completed.

The choice of block size **B** has an effect on the accuracy of the estimator [9]. The block size should be chosen to contain the minimum feature size, such as a moving hand or limb, in the source image in order to track such objects. Small block sizes can be used for finer gradations of the estimate, but the possibility of false positive correspondence increases inversely with window size. For example, searching over an image of a textured surface such as a sweater, there are many possible close matches. Accuracy can increase with increased window size up to some point, after which the accuracy degrades as window size increases due to the inability to track small objects . However, computational complexity increases quadratically with window size [93] unless the samples in the block are subsampled during the matching search. Also, too large a window can introduce false motion as an artifact [49], which is noticeable as a corona of moving background around moving objects.

Another factor in the accuracy of the estimation is the extent of the pattern matching search. Too small of a search area may not extend far enough to accurately detect large displacements. On the other hand, too large of a search area may converge an incorrect match.

The displacement vector fields obtained from the motion estimation are transmitted at one per keyframe. The interpolating filter needs a displacement vector for each individual pel; thus the set of displacement vectors must be expanded into an array equal to the dimensions of the original image.

To avoid block artifacts, a bilinear interpolation can be performed on the displacement vectors to smooth the expanded vector array. From this expanded array of vectors $\vec{d'} = (dx, dy)$, the pel at frame $n + i$ is interpolated using frames n and $n + N$. In the general case of interpolating $N - 1$ intermediate frames from two keyframes (frames n and $n + N$), the interpolation filter weighted more heavily on the closer keyframe:

$$Y_{n+k}(x_i, y_j) = Y_n\left(x_i - dx_i\frac{k}{N}, y_i - dy_i\frac{k}{N}\right)\frac{N-k}{N} + Y_{n+N}\left(x_i + dx_i\frac{N-k}{N}, y_i + dy_i\frac{N-k}{N}\right)\frac{k}{N}. \tag{2.7}$$

Care must be taken to avoid over smoothing the vector field; too much smoothing can result in artifacts which often look as if drops of water had blended neighboring samples together.

2.6 Vector Quantization

Vector Quantization (VQ) is a statistical coding method which maps multiple samples in the source alphabet to a single codeword. Even if coded in a lossless manner, multiple samples will always have an entropy less than or equal to the single sample entropy [10, 23, 85]. This mapping must be known at the receiver; generally a code book is transmitted. If the number of samples mapped to each codeword is too large, the cost in channel bitrate of transmitting the code book can negate the savings of jointly coding multiple source letters.

The technique of vector quantization originally was developed for speech coding [51, 72] , but now is commonly used in image processing [1, 5, 6, 7, 8, 11, 14, 15, 17, 18, 31, 33, 34, 52, 56, 61, 74, 86, 88, 91, 92, 96, 101, 104]. It is especially well suited for pyramid coding as the separate subbands can be coded to meet bit allocation or error criteria tailored to the energy in that band or the band's

relative importance to the HVS.

2.6.1 Codebook Determination

The vector quantization process divides a source space into non-overlapping regions and maps all samples in that region to a single codeword ie.,., a representative vector is calculated for each region. These regions are multidimensional volumes within the source domain. Each sample in the source block corresponds to a dimension or axis in the domain space. The term *vector* is used as each sample (block) corresponds to a vector in this multidimensional domain space.

Most often in image coding applications the source image is divided into rectangular blocks of pels before vector coding, although other partitions have been used. These blocks may also be grouped with other related blocks. For instance, blocks of pels in RGB space may be grouped together. Westerink [101] has vector coded blocks from several subbands together.

In order to quantize these blocks, some method of partitioning the source domain must be used. The first scheme and probably the most common is the Linde, Buzo and Gray (LBG) method [51]. The algorithm starts by making an initial guess for the code book. Each input vector is assigned to the code which is closest in Euclidean distance from it. An error metric (distortion), most often the sum of mean-squared-errors, for this code book is calculated. The vectors in the codebook are then replaced by the centroid of all the source vectors which map to it. The distortion is recalculated and the process is repeated until the distortion fails to decrease. The LBG algorithm does not guarantee that the minimum reached will be a global minimum, and it is computationally intensive.

More efficient algorithms use tree based techniques to split and search the vector space. These take on the order of $n \log n$ operations. K-dimensional trees (k-d trees) have been successfully used in vector quantization [18, 80, 88].

Once the vector code book has been determined, the source image must then be rendered. A

code vector is assigned to each block in the source. If the LBG algorithm is used, each vector is assigned by a minimum distortion criterion. If a k-d tree was used, the tree is traversed making scalar decisions based on the data along the split axis for that node. The traversal continues until a leaf is reached to determine the codeword.

One of the most important features of the k-d tree partition is that the code vectors are sorted in a manner that is convenient for this rendering process. To render using vectors from the LBG method, the distortion of each block against the vectors in the entire code book must be calculated.

2.6.2 Bit Allocation

One important reason for using pyramid coding is the flexibility it affords in bit allocation. The information content in real scenes varies considerably over time. It is useful to allocate the channel use as the information varies. The quality can be set at some constant level letting the bit rate vary accordingly. In addition, if buffering is available a period of low information can allow later more complex scenes to be coded at a higher effective bit rate, while maintaining relatively constant channel bandwidth. In the case of subband coding, bit allocation can vary between bands as different spatial bands have nonstationary energy distributions.

Early work in VQ used a fixed codebook size to bound the channel use. This has the drawback that quality varies with content; more complex scenes that contain more information will be reproduced at a lower quality level than others. More recent work [19, 80] has used error limits to bound the size of the codebook. With this method the domain continues to be split until the desired distortion level is reached. The distortion metric may be mean-squared error, maximum peak error, peak signal to noise ratio , mean-squared signal to noise ratio or some other metric.

2.7 Entropy Coding

In this thesis the term *Entropy Coding* is used for any method that *losslessly* reduces the number of bits necessary to represent a data set, eg. attempts to reach an average number of bits per word that approaches the entropy, H of the source:

$$H = -\sum Q_n \log_2 Q_n \tag{2.8}$$

where Q_n is the probability of the source letter n occurring.

2.7.1 Huffman Coding

Huffman coding [37] has long been used to reduce the data rate needed to transmit a source. Huffman coding is a fixed length to variable length coding technique. The Huffman technique allocates the fewest bits per sample to the most common source letters and more bits per sample to the less common source letters. The output code words are concatenated to form a prefix or suffix code that can be uniquely decoded [10, 23]. Each symbol's length approximates the entropy of that symbol.

$$L_n = \lceil -\log_2 (Q_n) \rceil \tag{2.9}$$

where $\lceil \cdot \rceil$ is the next larger integer greater than or equal to $\cdot$, and Q_n is the probability of source letter n occurring. Thus, the average number of bits per symbol L over a long stream of data is

$$L \geq H = -\sum_{i=1}^{N} Q_n \log_2 (Q_n) . \tag{2.10}$$

Where H is the entropy of the source as defined in Equation 2.8.

Actually L is bounded on both sides [23]

$$H + 1 \geq L \geq H. \tag{2.11}$$

Forming the code book is the most computationally intensive part of Huffman coding. The code book is formed by building a tree. The N letters in the source alphabet are the leaves of the tree, each of which is assigned a probability Q_n [24]. Figure 2.9 shows an example of Huffman prefix code assignment using a tree. The probability distribution Q may be obtained from the sample statistics or may be based on some apriori model of the source. The two least probable leaves are joined into a node. This node assumes the probability of the sum of the probabilities of the two leaves. Each branch is assigned a zero or a one according to whether it is the left or right branch. The node is now treated as a leaf in the next grouping of the two least probable leaves. This grouping continues until the root node (with probability one) is reached. The code word for each source letter is found by following from leaf to root reading off the binary digits. This codebook can now be used as a look-up table to encode the source. The variable length codewords are then concatenated, before being sent through the communication channel.

Decoding is done by scanning through the transmitted bits branching left or right through the tree as each bit arrives. A code word is known to be complete when a leaf is reached, the decoded letter is emitted and a new tree walk started.

Note that the number of bits for each symbol in a Huffman code must be an integer. This is the major limitation of this coding method.

2.7.2 Arithmetic Coding

Arithmetic coding is another method of fixed length to variable length coding that has recently become more popular [50, 57, 60, 76, 77, 78, 84]. It is referred to as *arithmetic* coding because it

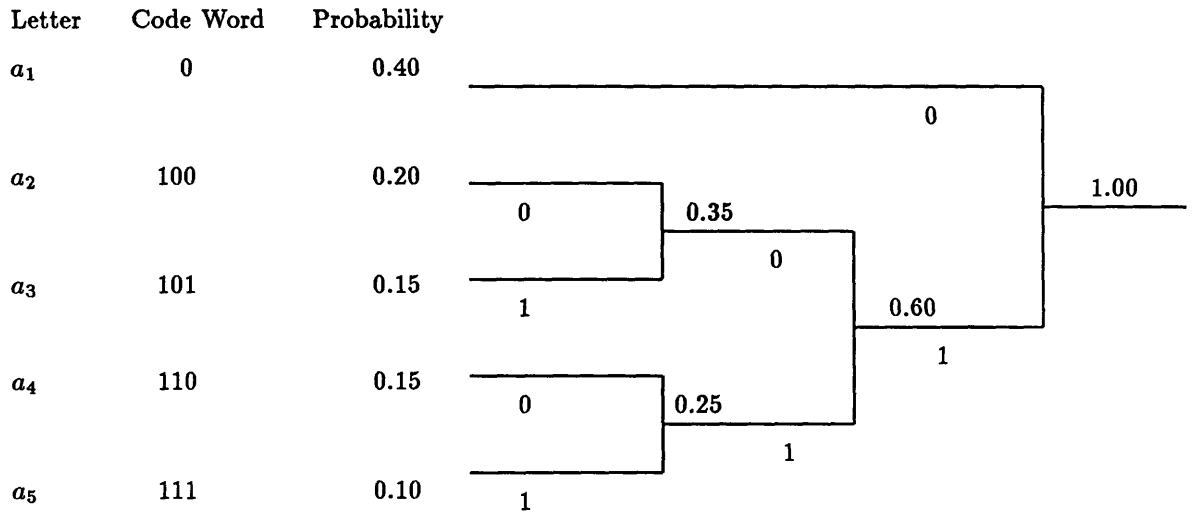

Figure 2.9: An example of building a Huffman code

compiles a code string which is the arithmetic combination of the probabilities of the individual symbols. The major advantage of arithmetic coding over Huffman coding is that the number of bits per symbol is not constrained to be an integer, so it can more closely approach the theoretical entropy limit. The coding process involves the addition of binary fractions rather than concatenating integer length code words.

Arithmetic coding performs a mapping of a source sequence $a_0, a_1, a_2, \ldots$ to a point x on the interval $[0, 1)$ which represents the cumulative probabilities of the symbols $a_0 \ldots a_n$. The value of x in the interval $[0, 1)$ can be expanded in negative powers of two as

$$x = \sum_{i=0}^{\infty} x_i \, 2^{-(i+1)} \tag{2.12}$$

where each of the x_i is a binary digit. This mapping of the source string to the unit interval is done in such a way that x is uniformly distributed in the interval $[0, 1)$, thus each of these x_i are statisticly independent and each has equal probability of being 0 or 1.

Each member of the source alphabet is assigned a subinterval in $[0, 1)$ according to the probability of that letter occurring. As each letter in the string is emitted by the source, the unit interval is iteratively divided into subintervals.

This subdivision continues until, due to the finite precision of the code register, no further subintervals can be made. The code word is then emitted and the process continues. The output code word associated with each string is the left end point of the interval of interest.

2.8 Color Coding

Color systems are typically characterized by three linearly independent primaries. These can be transformed through matrix multiplications to other color spaces. Through this change of basis a color space more suitable for coding can be used. In this manner the energy can be compacted into channels that are relatively more important to the HVS and thus, can be allocated different portions of the channel bandwidth.

A transform in which chrominance is orthogonal to luminance matches the HVS well. The common broadcast standard NTSC color space YIQ, where Y is luminance and IQ are chrominance [83], is one such color space. This is related to the primaries RGB by

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.596 & -0.274 & -0.322 \\ 0.211 & -0.522 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.13}$$

and the inverse

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0 & 0.956 & 0.623 \\ 1.0 & -0.272 & -0.648 \\ 1.0 & -1.105 & 0.705 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \tag{2.14}$$

Another commonly used color space is YUV

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} .299 & .587 & .114 \\ .701 & -.587 & .500 \\ -.299 & -.587 & .886 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.15}$$

and YUV to RGB

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.00 & 1.00 & 0.00 \\ 1.00 & -.509 & -.194 \\ 1.00 & 0.00 & 1.00 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \tag{2.16}$$

Note that the inverse YUV (equation 2.16) needs only two multiplications and four additions for each three sample conversion versus the inverse YIQ which needs six multiplications and six additions. This could be a factor in a broadcast type system with few transmitters and many receivers.

The temporal response of the HVS to chrominance is less than the response to luminance. Some coding schemes [14, 80] have updated chrominance at half the rate of luminance with little perceptible loss of quality.

The chrominance components are closely related to luminance. This correlation has been used in reducing the total amount of data for the color image system [12]. The three color components may also be treated as a single vector and vector quantized as such [104].

Chapter 3

Motion Compensated Interpolation: Theory and Limitations

3.1 Mean Square Estimation

In this chapter the theory and limitations of Motion Compensated Interpolation will be developed.

We wish to estimate the value of an intermediate frame, $\mathbf{s}(x, y, t+\tau)$ $1 < \tau < N-1$, where τ is the frame number, from samples at $\mathbf{s}(x, y, t)$ and $\mathbf{s}(x, y, t+N)$ (the keyframes)

$$\hat{\mathbf{s}}(x, y, t+\tau) = \sum_{j=0}^{npels} \sum_{k=0}^{nlines} \left[a_{j,k,0}\, \mathbf{s}(x+j, y+k, t) + a_{j,k,N}\, \mathbf{s}(x+j, y+k, t+N) \right]. \qquad (3.1)$$

In this case, we are assuming pure integer displacement of each pel, so all the $a_{j,k,n}$ are zero,

except coefficients at $t = 0, j = -\widehat{dx}\,\frac{\tau}{N}, k = -\widehat{dy}\,\frac{\tau}{N}$ $a_{j,k,0} = \frac{(N-\tau)}{N}$ and at $t = N, j = \widehat{dx}\,\frac{(N-\tau)}{N}, k = \widehat{dy}\,\frac{(N-\tau)}{N}$ $a_{j,k,N} = \frac{\tau}{N}$, corresponding to the displaced pels in the previous and next keyframes. Keeping only the non-zero terms in (3.1) gives:

$$\hat{\mathrm{s}}\,(x, y, t+\tau) = \frac{(N-\tau)}{N}\mathrm{s}\left(x - \widehat{dx}\,\frac{\tau}{N}, y - \widehat{dy}\,\frac{\tau}{N}, t\right) + \frac{\tau}{N}\mathrm{s}\left(x + \widehat{dx}\,\frac{(N-\tau)}{N}, y + \widehat{dy}\,\frac{(N-\tau)}{N}, t+N\right). \tag{3.2}$$

Define the interpolation error $\varepsilon_N(\tau)$ as the difference between the true signal $\mathrm{s}(x, y, t+\tau)$ and the interpolated estimate $\hat{\mathrm{s}}(x, y, t+\tau)$

$$\begin{aligned}
\varepsilon_N(\tau) &= \mathrm{s}\,(x, y, t+\tau) - \hat{\mathrm{s}}\,(x, y, t+\tau) \\
&= \mathrm{s}\,(x, y, t+\tau) - \\
&\quad \left[\frac{(N-\tau)}{N}\mathrm{s}\left(x - \widehat{dx}\,\frac{\tau}{N}, y - \widehat{dy}\,\frac{\tau}{N}, t\right) + \frac{\tau}{N}\mathrm{s}\left(x + \widehat{dx}\,\frac{(N-\tau)}{N}, y + \widehat{dy}\,\frac{(N-\tau)}{N}, t+N\right)\right].
\end{aligned} \tag{3.3}$$

From Equation (3.3) it follows that the mean square value, MS, of the interpolation error is

$$\mathbf{MS} = \mathrm{E}\left[(\varepsilon_N(\tau))^2\right] \tag{3.4}$$

$$= \mathrm{E}\left[(\mathrm{s}(x, y, t+\tau) - \hat{\mathrm{s}}(x, y, t+\tau))^2\right] \tag{3.5}$$

$$= \mathrm{E}\left[\mathrm{s}^2(x, y, t+\tau)\right] + \mathrm{E}\left[\hat{\mathrm{s}}^2(x, y, t+\tau)\right] - 2\,\mathrm{E}\left[\mathrm{s}(x, t, t+\tau)\hat{\mathrm{s}}(x, y, t+\tau)\right] \tag{3.6}$$

Where E[·] is the expected value.

Expanding the three terms in Equation (3.6) by using (3.2) gives:

$$\mathrm{E}\left[\mathrm{s}^2(x, y, t+\tau)\right] = \mathbf{R}(0, 0, 0) \tag{3.7}$$

$$
\begin{aligned}
\mathrm{E}\left[\hat{\mathbf{s}}^2(x,y,t+\tau)\right] &= \mathrm{E}\left[\left(\frac{N-\tau}{N}\right)^2 \mathbf{s}^2\left(x-\widehat{dx}\frac{\tau}{N}, y-\widehat{dy}\frac{\tau}{N}, t\right)\right. \\
&\quad + \left(\frac{\tau}{N}\right)^2 \mathbf{s}^2\left(x+\widehat{dx}\left(\frac{N-\tau}{N}\right), y+\widehat{dy}\left(\frac{N-\tau}{N}\right), t+N\right) \\
&\quad + 2\left(\frac{N-\tau}{n}\right)\left(\frac{\tau}{N}\right)\mathbf{s}\left(x-\widehat{dx}\frac{\tau}{N}, y-\widehat{dy}\frac{\tau}{N}, t\right) \\
&\quad \left.\mathbf{s}\left(x+\widehat{dx}\left(\frac{N-\tau}{N}\right), y+\widehat{dy}\left(\frac{N-\tau}{N}\right), t+N\right)\right] \\
&= \left(\frac{N-\tau}{N}\right)^2 \mathbf{R}(0,0,0) + \left(\frac{\tau}{N}\right)^2 \mathbf{R}(0,0,0) \\
&\quad + 2\left(\frac{N-\tau}{N}\right)\frac{\tau}{N}\mathbf{R}\left(\widehat{dx}\left(\frac{N-\tau}{N}\right)+\widehat{dx}\frac{\tau}{N}, \widehat{dy}\left(\frac{N-\tau}{N}\right)+\widehat{dy}\frac{\tau}{N}, N\right) \\
&= \left\{\left(\frac{N-\tau}{N}\right)^2 + \left(\frac{\tau}{N}\right)^2\right\}\mathbf{R}(0,0,0) \\
&\quad + 2\left(\frac{N-\tau}{N}\right)\frac{\tau}{N}\mathbf{R}\left(\widehat{dx}, \widehat{dy}, N\right) \qquad (3.8)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{E}\left[\mathbf{s}(x,y,t+\tau)\hat{\mathbf{s}}(x,y,t+\tau)\right] &= \mathrm{E}\left[\mathbf{s}(x,y,t+\tau)\left\{\left(\frac{N-\tau}{N}\right)\mathbf{s}\left(x-\widehat{dx}\frac{\tau}{N}, y-\widehat{dy}\frac{\tau}{N}, t\right)\right.\right. \\
&\quad \left.\left. + \left(\frac{\tau}{N}\right)\mathbf{s}\left(x+\widehat{dx}\left(\frac{N-\tau}{N}\right), y+\widehat{dy}\left(\frac{N-\tau}{N}\right), t+N\right)\right\}\right] \\
&= \left(\frac{N-\tau}{N}\right)\mathbf{R}\left(\widehat{dx}\frac{\tau}{N}, \widehat{dy}\frac{\tau}{N}, \tau\right) \\
&\quad + \frac{\tau}{N}\mathbf{R}\left(\widehat{dx}\left(\frac{N-\tau}{N}\right), \widehat{dy}\left(\frac{N-\tau}{N}\right), N-\tau\right) \qquad (3.9)
\end{aligned}
$$

$\mathbf{R}(\cdot)$ is the autocorrelation function:

$$
\mathbf{R}(u,v,\tau) = \mathrm{E}\left[\mathbf{s}(x,y,t)\mathbf{s}(x+u, y+v, t+\tau)\right]. \qquad (3.10)
$$

Substituting Equations (3.7),(3.8), and (3.9) into Equation (3.6) gives the following result for the mean squared error:

$$
\begin{aligned}
\mathrm{MS} &= \mathbf{R}(0,0,0) + \left[\left(\frac{N-\tau}{N}\right)^2 + \left(\frac{\tau}{N}\right)^2\right]\mathbf{R}(0,0,0) \\
&\quad + 2\left(\frac{N-\tau}{N}\right)\left(\frac{\tau}{N}\right)\mathbf{R}\left(\widehat{dx}, \widehat{dy}, N\right) - 2\left(\frac{N-\tau}{N}\right)\mathbf{R}\left(\widehat{dx}\left(\frac{\tau}{N}\right), \widehat{dy}\left(\frac{\tau}{N}\right), \tau\right)
\end{aligned}
$$

$$
\begin{aligned}
& - \left(\frac{\tau}{N}\right) \mathbf{R}\left(\widehat{dx}\left(\frac{N-\tau}{N}\right), \widehat{dy}\left(\frac{N-\tau}{N}\right), N-\tau\right) \\
= \; & \frac{2}{N^2}\left(N^2+\tau^2-N\tau\right)\mathbf{R}\left(0,0,0\right) + 2\left(\frac{N-\tau}{N}\right)\left(\frac{\tau}{N}\right)\mathbf{R}\left(\widehat{dx}, \widehat{dy}, N\right) \\
& - 2\left(\frac{N-\tau}{N}\right)\mathbf{R}\left(\widehat{dx}\left(\frac{\tau}{N}\right), \widehat{dy}\left(\frac{\tau}{N}\right), \tau\right) \\
& - \left(\frac{\tau}{N}\right) \mathbf{R}\left(\widehat{dx}\left(\frac{N-\tau}{N}\right), \widehat{dy}\left(\frac{N-\tau}{N}\right), N-\tau\right)
\end{aligned}
\tag{3.11}
$$

3.2 Image Models

The image correlation function can be modeled as an isotropic function [66], such that the autocorrelation depends on the Euclidean distance

$$
\mathbf{R}(x,y) = \sigma^2 \alpha^{-a\ d(x,y)}. \tag{3.12}
$$

The function $d(x,y)$ is some distance measure such as

$$
d(x,y) = (x^2+y^2)^{\frac{1}{2}}. \tag{3.13}
$$

The constants α and a together determine the rate that the autocorrelation decreases as you move away from the origin. This simply means that the correlation is decreasing as you move in any direction away from the correct point. Figure 3.1 shows a plot of an isotropic autocorrelation function $e^{0.9(x^2+y^2)^{\frac{1}{2}}}$.

In the case of pure translational motion, the intensity is related by

$$
\mathbf{s}(x,y,t+\tau) = \mathbf{s}(x+\tau\,\delta_x, y+\tau\,\delta_y, t) \tag{3.14}
$$

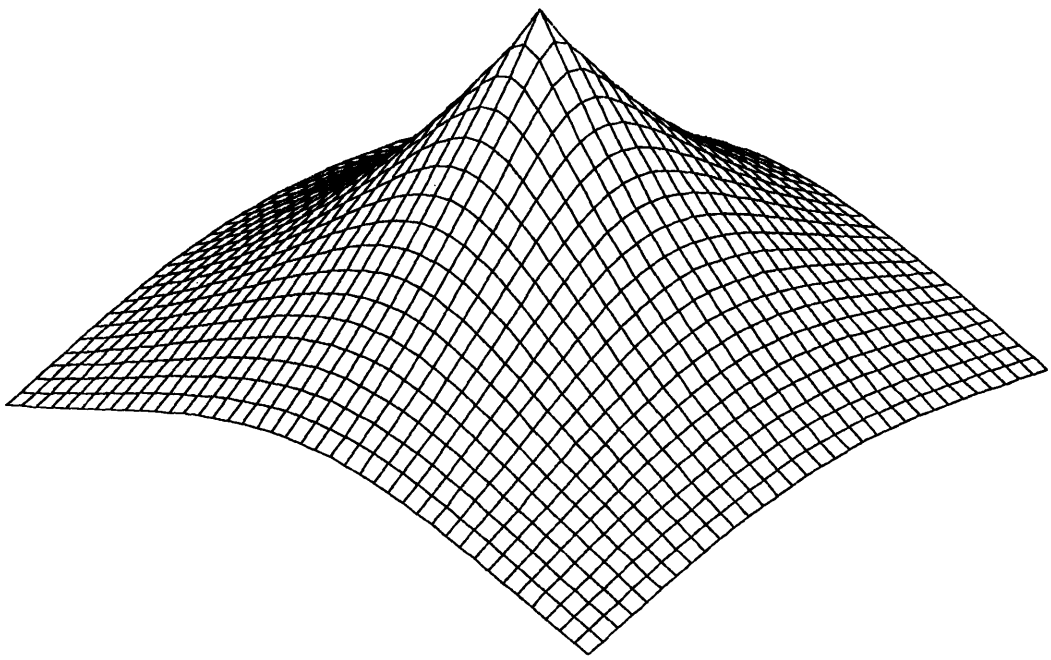

Figure 3.1: An isotropic two dimensional autocorrelation function

or in terms of the two keyframes at t and $t+N$

$$\mathrm{s}\,(x,y,t+\tau) = \frac{(N-\tau)}{N}\mathrm{s}\left(x - dx\,\frac{\tau}{N}, y - dy\,\frac{\tau}{N}, t\right) + \frac{\tau}{N}\mathrm{s}\left(x + dx\,\frac{(N-\tau)}{N}, y + dy\,\frac{(N-\tau)}{N}, t+N\right) \tag{3.15}$$

where the true displacement between t and $t+N$ is $dx = N\,\delta_x, dy = N\,\delta_y$. The estimated displacement differs from the true displacement by

$$\widehat{dx} = dx + \epsilon_x \tag{3.16}$$

$$\widehat{dy} = dy + \epsilon_y. \tag{3.17}$$

So, using the autocorrelation function given in 3.12 and 3.13,

$$\mathbf{R}(x,y,\tau) = \sigma^2(\tau)\alpha^{-a\left(\epsilon_x{}^2+\epsilon_y{}^2\right)^{\frac{1}{2}}}. \tag{3.18}$$

From (3.11) and (3.18), the mean square error will be

$$\begin{aligned}\mathrm{MS} = {} & \frac{2}{N^2}\left(N^2+\tau^2-N\tau\right)\sigma^2(0) + 2\left(\frac{N-\tau}{N}\right)\left(\frac{\tau}{N}\right)\sigma^2(N)\alpha^{-a\left(\epsilon_x^2+\epsilon_y^2\right)^{\frac{1}{2}}} \\ & - 2\left(\frac{N-\tau}{N}\right)\sigma^2(\tau)\alpha^{-a\,\frac{\tau}{N}\left(\epsilon_x^2+\epsilon_y^2\right)^{\frac{1}{2}}} \\ & - \left(\frac{\tau}{N}\right)\sigma^2(N-\tau)\alpha^{-a\,\frac{N-\tau}{N}\left(\epsilon_x^2+\epsilon_y^2\right)^{\frac{1}{2}}}.\end{aligned} \tag{3.19}$$

Typical values in real images are $0.90 < \alpha^{-a} < 1$ [63]. If an accurate motion estimation was made using integer pel accuracy, the difference from the true displacement in (3.16) and (3.17) is from round off errors. Then ϵ_x and ϵ_y are bounded less than $\frac{1}{2}$. Using the values $\sigma^2 = 1$, $\alpha^{-a} = 0.90$, and $\epsilon_x = \epsilon_y = 0.50$, and the variance $\sigma^2(\tau)$ from the *Alley* sequence, Equation (3.19) is plotted

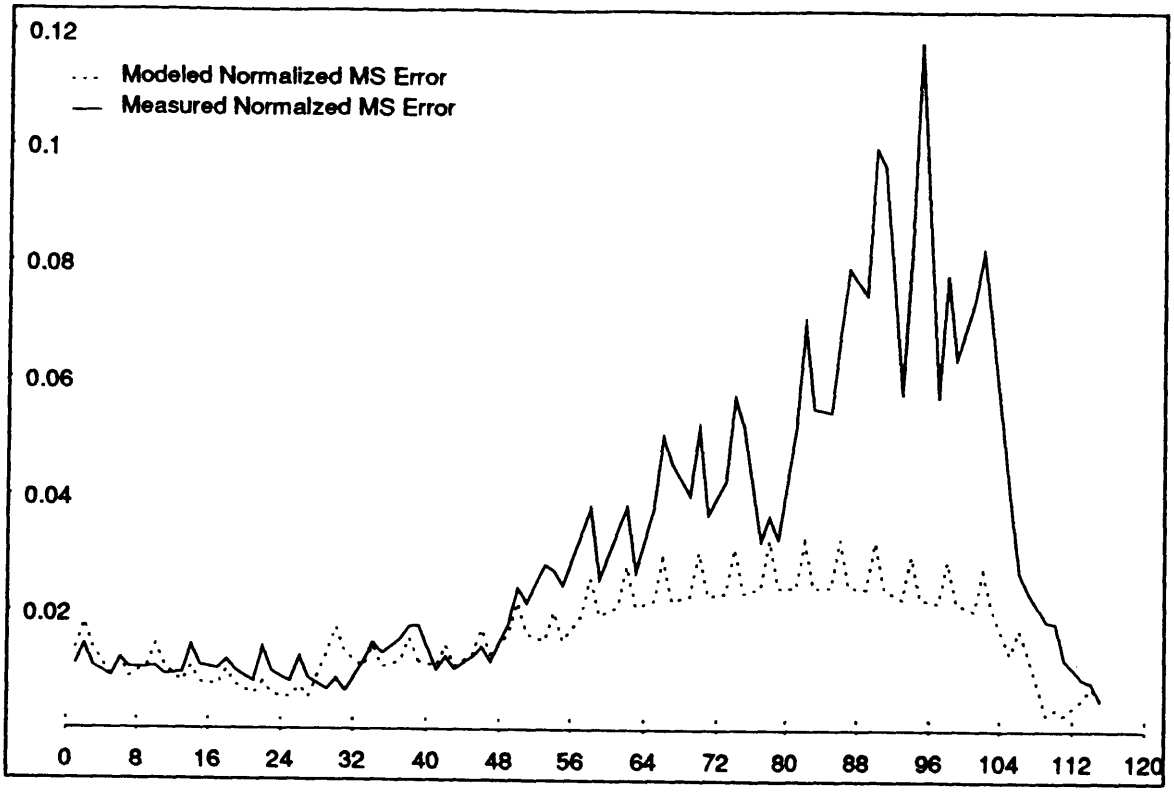

Figure 3.2: Measured and modeled mean square error using autocorrelation function $\mathbf{R}(x,y) = \sigma^2\alpha^{-a(x^2+y^2)^{\frac{1}{2}}}$.

in Figure 3.2 along with the measured normalized mean square interpolation error for the sequence using keyframe spacing of four frames. Note that the measured values shown in the graph are the interpolation mean square errors obtained from interpolation using uncoded keyframes, normalized by the maximum variance of the keyframes.

It is seen that the curve for the model follows the general shape of the measured curves. Note that the interpolation error within a set of intermediate frames interpolated with the same vectors is highest in the center frame, which is the farthest frame from the two keyframes. The maximum interpolation error also occurs when the greatest fraction of the image is in motion.

To improve the model of (3.19), the assumption is made that non-moving areas have zero error, thus the expected mean square error is

$$\widetilde{\mathrm{MS}} = mv\,\mathrm{MS} \tag{3.20}$$

where mv is the fraction of area in the image which is moving. The percentage of each frame set which was classified as non-moving is given in Table 3.1 (See section 6.2.1 for a discussion of how the non-moving area is classified).

For comparison with measured values, the modeled mean square error is weighted with the fraction of moving area, mv, in the images:

$$\begin{aligned}\widetilde{\mathrm{MS}} \;=\; mv \Bigg[& \frac{2}{N^2}\left(N^2+\tau^2-N\tau\right)\sigma^2(0) + 2\left(\frac{N-\tau}{N}\right)\left(\frac{\tau}{N}\right)\sigma^2(N)\alpha^{-a\left(\epsilon_x^2+\epsilon_y^2\right)^{\frac{1}{2}}} \\ & -2\left(\frac{N-\tau}{N}\right)\sigma^2(\tau)\alpha^{-a\,\frac{\tau}{N}\left(\epsilon_x^2+\epsilon_y^2\right)^{\frac{1}{2}}} \\ & -\left(\frac{\tau}{N}\right)\sigma^2(N-\tau)\alpha^{-a\,\frac{N-\tau}{N}\left(\epsilon_x^2+\epsilon_y^2\right)^{\frac{1}{2}}}\Bigg].\end{aligned} \tag{3.21}$$

Using the normalized sample variance value from the keyframes of the sequence *Alley* for $\sigma^2(\tau)$, $\alpha^{-a} = 0.90$, and $\epsilon_x = \epsilon_y = 0.50$, Equation 3.21 is plotted in figure 3.3 along with the measured normalized mean square interpolation error for the sequence frames and the modeled error of Equation (3.20). This model more closely follows the experimental observations. This implies that the amount of motion in a scene has a significant effect on the interpolation error. It is important to note also that the in the real sequence, the motion is not simple translational displacement of rigid objects. In particular, the model fails in the frames 85 to 105, where one figure crosses in front, occluding the other. The interpolation error becomes large here, when the displacement estimator is unable to determine the flow path of the covering and uncovering areas.

Frames	Percent Non-Moving
$1 \rightarrow 3$	49.729168
$5 \rightarrow 7$	67.125000
$9 \rightarrow 11$	60.229168
$13 \rightarrow 15$	70.312500
$17 \rightarrow 19$	71.833336
$21 \rightarrow 23$	77.666664
$25 \rightarrow 27$	79.500000
$29 \rightarrow 31$	52.020832
$33 \rightarrow 35$	60.645832
$37 \rightarrow 39$	56.979168
$41 \rightarrow 43$	60.187500
$45 \rightarrow 47$	52.687500
$49 \rightarrow 51$	40.041668
$53 \rightarrow 55$	44.395832
$57 \rightarrow 59$	28.708334
$61 \rightarrow 63$	22.833334
$65 \rightarrow 67$	19.166666
$69 \rightarrow 71$	16.062500
$73 \rightarrow 75$	14.000000
$77 \rightarrow 79$	10.354167
$81 \rightarrow 83$	10.708333
$85 \rightarrow 87$	9.916667
$89 \rightarrow 91$	11.041667
$93 \rightarrow 95$	17.708334
$97 \rightarrow 99$	20.104166
$101 \rightarrow 103$	24.895834
$105 \rightarrow 107$	52.333332
$109 \rightarrow 111$	89.104164
$113 \rightarrow 115$	79.104164

Table 3.1: Percentage of area classified non-moving in *Alley* sequence

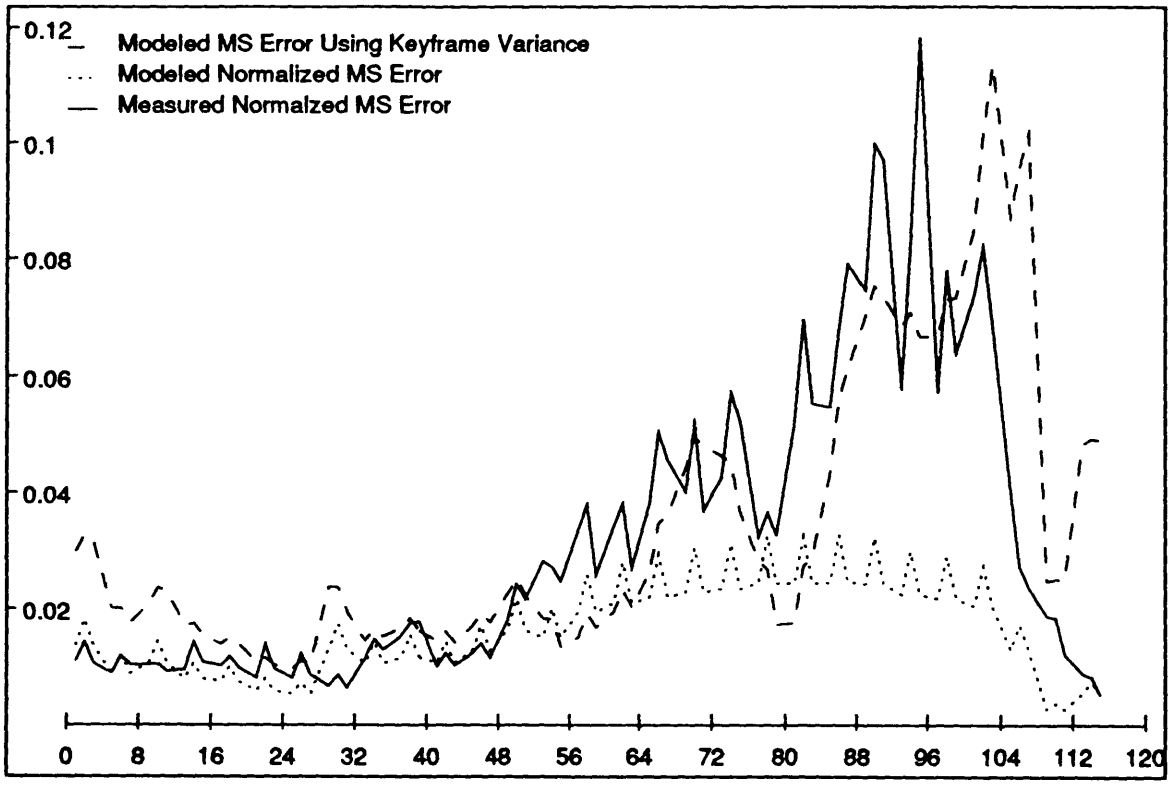

Figure 3.3: Measured vs. modeled mean square error using autocorrelation function $\mathbf{R}(x, y, \tau) = \sigma^2(\tau)\alpha^{-a(x^2+y^2)^{\frac{1}{2}}}$.

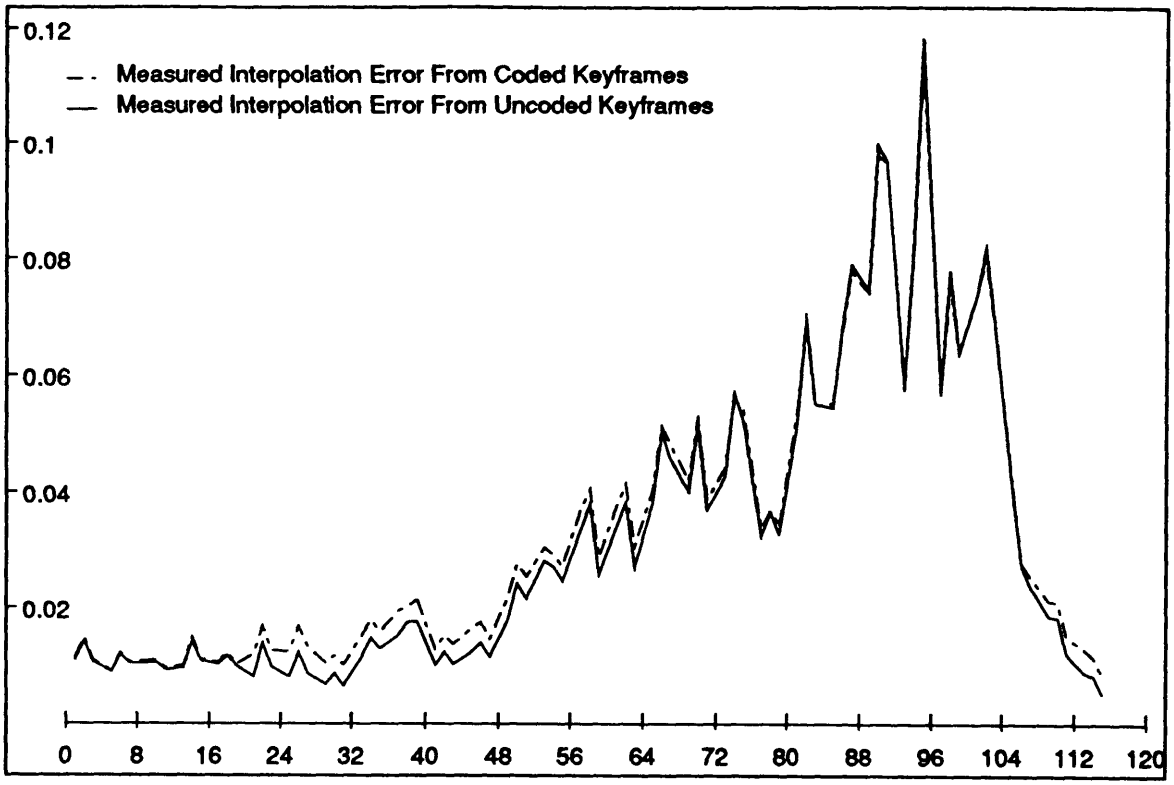

Figure 3.4: Measured normalized mean square error using coded and uncoded keyframes in *Alley* sequence

This page is intentionally left blank.

This page is intentionally left blank.

Chapter 4

Motion Estimation

4.1 Block Matching

Conventional block matching algorithms use the data from blocks within two frames to estimate linear displacement by finding the minimum of some error metric, such as the sum of absolute differences or mean square error.

This technique works well for interframe prediction techniques, but performs poorly for a motion compensated interpolation coding system. The two keyframes used for a motion estimate may describe the translational motion from keyframe to keyframe, but doe not accurately describe the translation from one keyframe to the next intermediate frame or from the second keyframe back to the intermediate frames. For this reason an improved motion estimation method for interpolation is used called *multiframe matching* (MFM).

Various block matching search techniques have been investigated [61]:

1. Two dimensional logarithmic search.

2. Modified conjugate direction search.

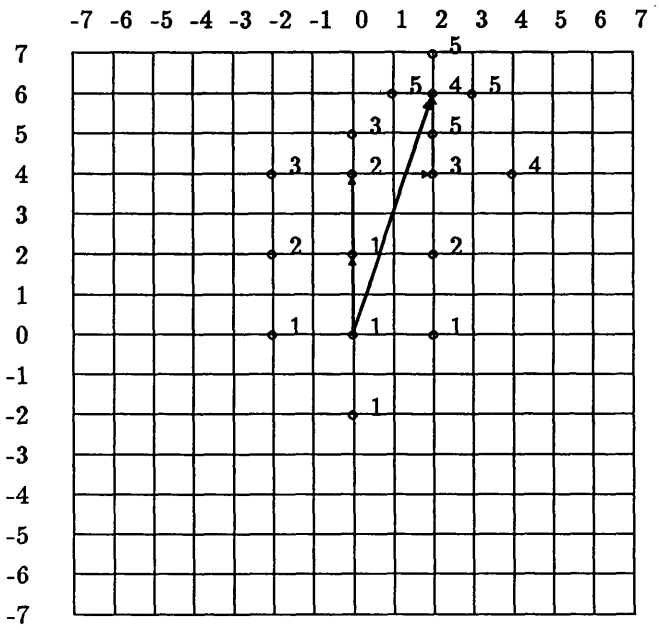

Figure 4.1: Two dimensional logarithmic search. From Musmann [61]

3. Three step search.

All of these techniques make the assumption that $E(\vec{d})$ increases monotonically as the search window shifts away from the direction of minimum difference.

In the two dimensional logarithmic search, the prediction error $E(x, y)$, is calculated at four displacements around the starting point (Figure 4.1). The displacement with the minimum distortion is selected as the next starting point, to follow the path of least distortion. For each successive search step, the distance between search windows is reduced by $\frac{1}{2}$, causing the search area to be reduced logarithmicly. The steps are repeated until the minimum distance resolution desired is reached.

The modified conjugate direction search uses two steps to search for the direction of minimum difference. A search is first performed in the x direction shifting one pel at a time to find the point of minimum difference. The search then proceeds in the y direction (Figure 4.2).

The three step search uses eight search positions spaced around the starting point (Figure 4.3).

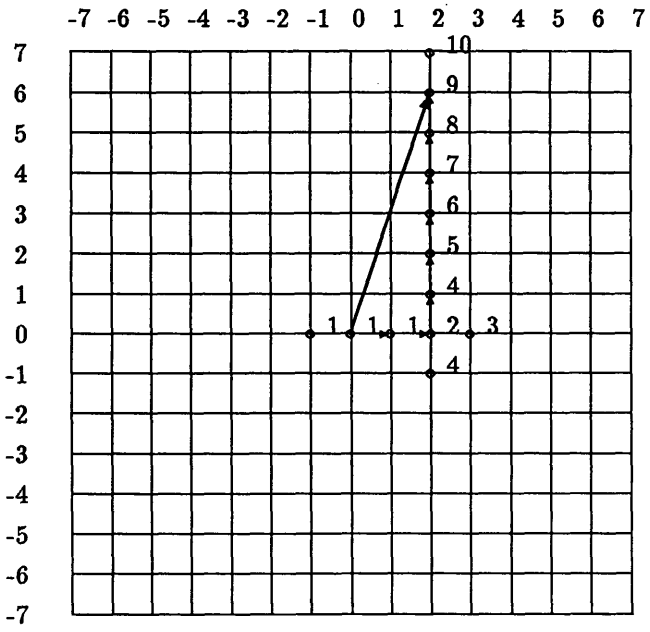

Figure 4.2: Conjugate direction search. From Musmann [61]

The point which had the minimum difference is then used as the starting point for the next search. Each successive search step uses more closely spaced window positions to give the distance resolution desired.

4.2 Multiframe Matching

Multiframe matching is an extension of block matching. It is applied to the image sequence in a temporal block fashion. The sequence is divided into sets of frames each of length $nframes$. The first and last frames in these sets are keyframes. The second keyframe of each preceding frame becomes the first keyframe in the current set of images.

This set of images, comprised of the two keyframes and the intermediate frames, is divided into tessellated three dimensional blocks, each of which is assigned a displacement vector (see figure 4.4).

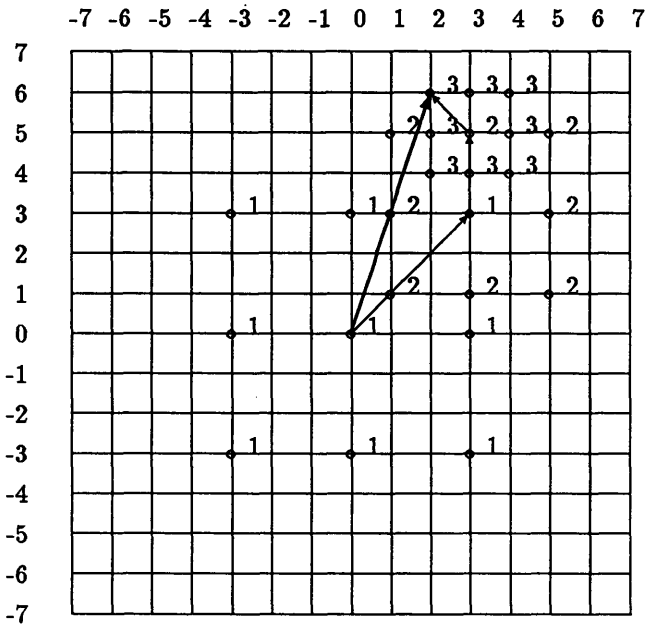

Figure 4.3: Three step search. From Musmann [61]

This displacement vector is determined by the minimum sum of absolute differences between the keyframes and the intermediate frames - weighted to reflect the distance to each keyframe. This method effectively gives the best average displacement within a three dimensional block, to minimize the sum of the interpolation error for the intermediate frames in that three dimensional block.

MFM performs better than block matching for recreating intermediate frames through interpolation. Figures 4.5 to 4.8 show the RMS error of interpolated frames using MFM and block matching. Note that further the distance between keyframes, the greater the performance improvement of MFM over block matching. Keyframe spacing of two is the case when MFM and block matching are equivalent.

The displacement estimation was performed using block size of 8x8 for both the block matching and MFM. The search area was $\pm$ 32 pels in both cases. This interpolation was performed using the original keyframes, not coded keyframes containing noise. Also, these data are from images which

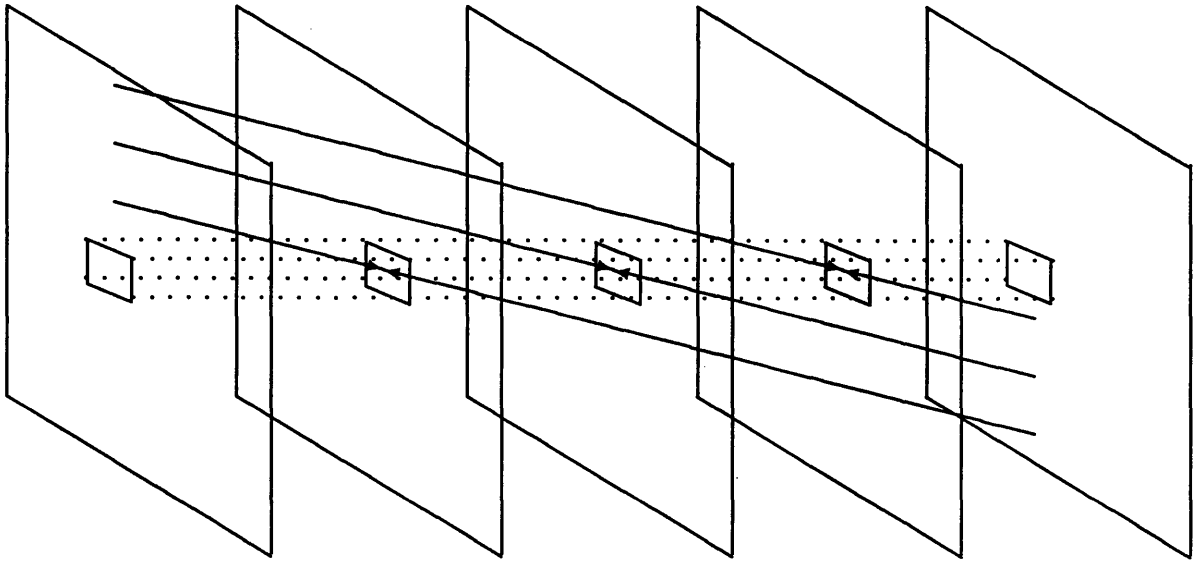

Figure 4.4: Example showing how a single displacement vector is used for each Three-dimensional volume.

were cropped to contain mostly moving areas.

4.3 Hierarchical Multiframe Matching

The method of multiframe matching is extended to an hierarchical search method. One of the errors encountered in pattern matching is false matches. For example, the movement of an arm over a textured sweater may cause false matches (and the resultant grossly incorrect displacement estimate) due to the regular pattern of the texture. The problem here is that the average feature size is smaller than the window size. If the search window contains an edge, this gives a more unique pattern to match resulting in more accurate displacement estimation.

Hence, it would appear that larger window sizes would give better results. This is true to some extent, but as the window size grows, localized motion is not detected, causing motion blurring in those areas of the interpolated frame.

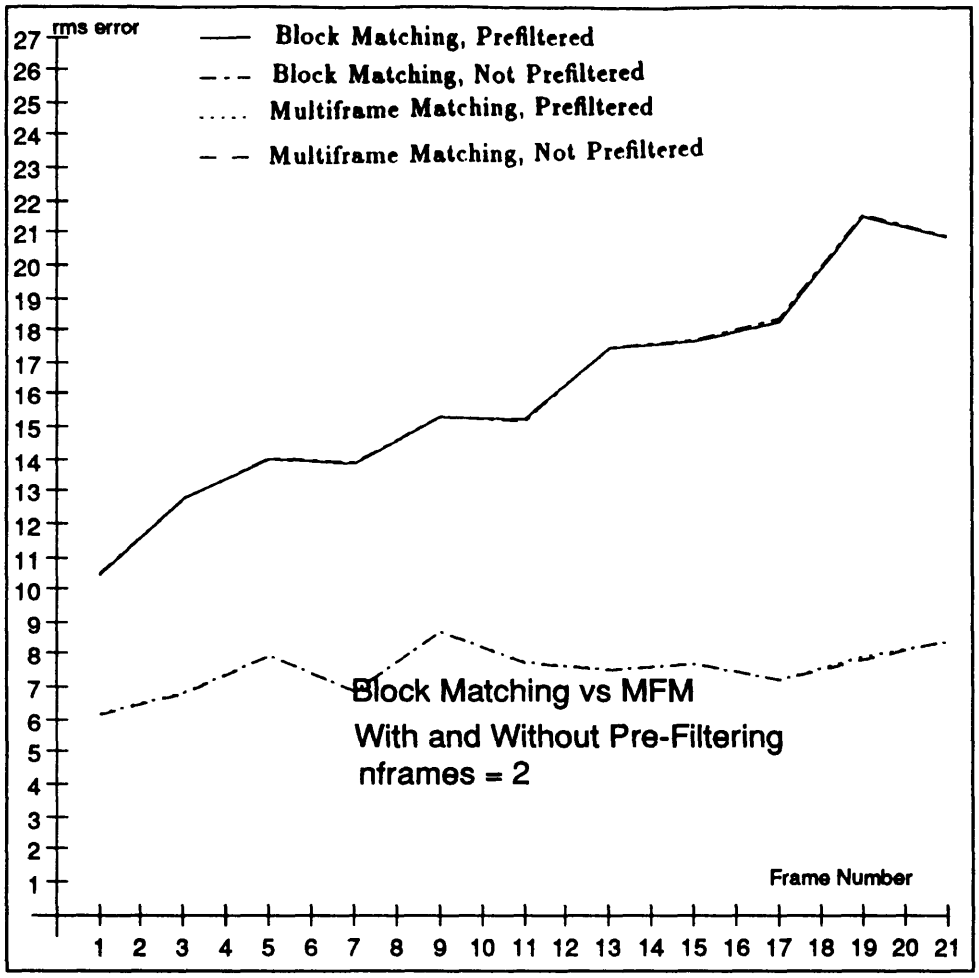

Figure 4.5: Comparison of Multiframe matching vs. Block matching with frame spacing = 2, with and without pre-filtering

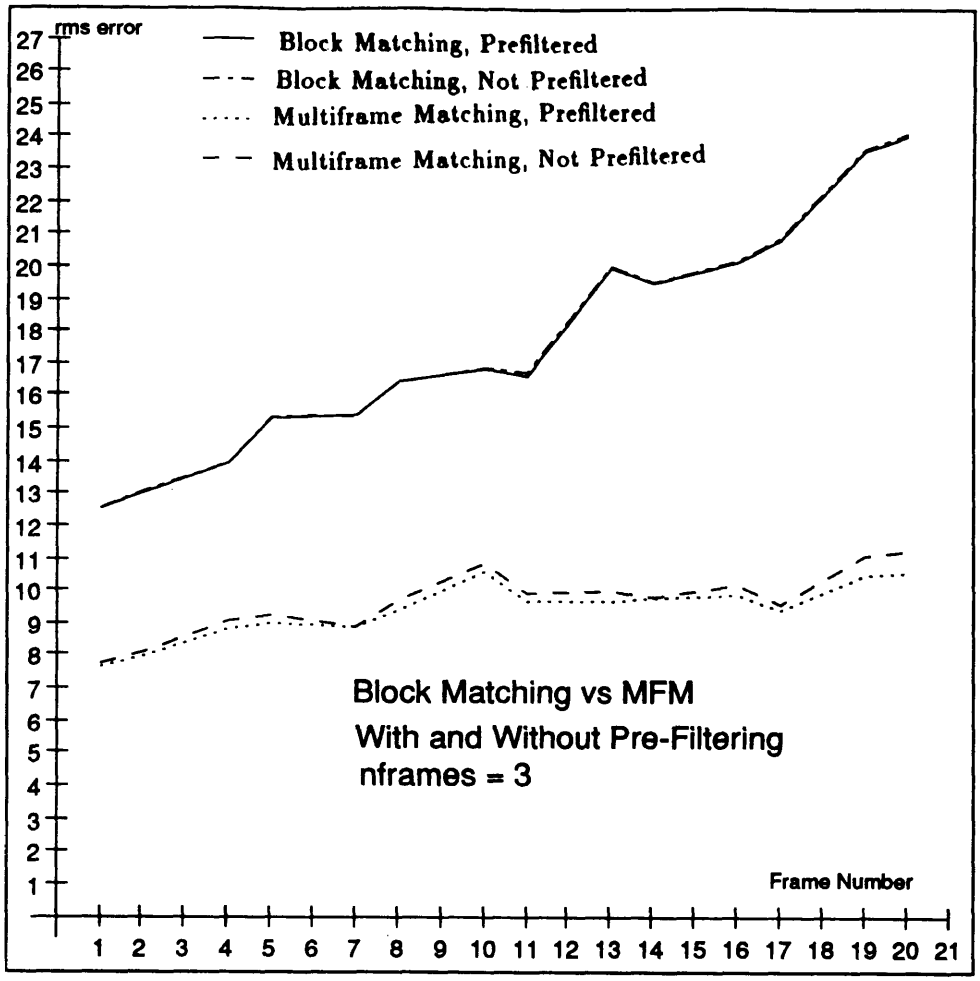

Figure 4.6: Comparison of Multiframe matching vs. Block matching with frame spacing = 3, with and without pre-filtering

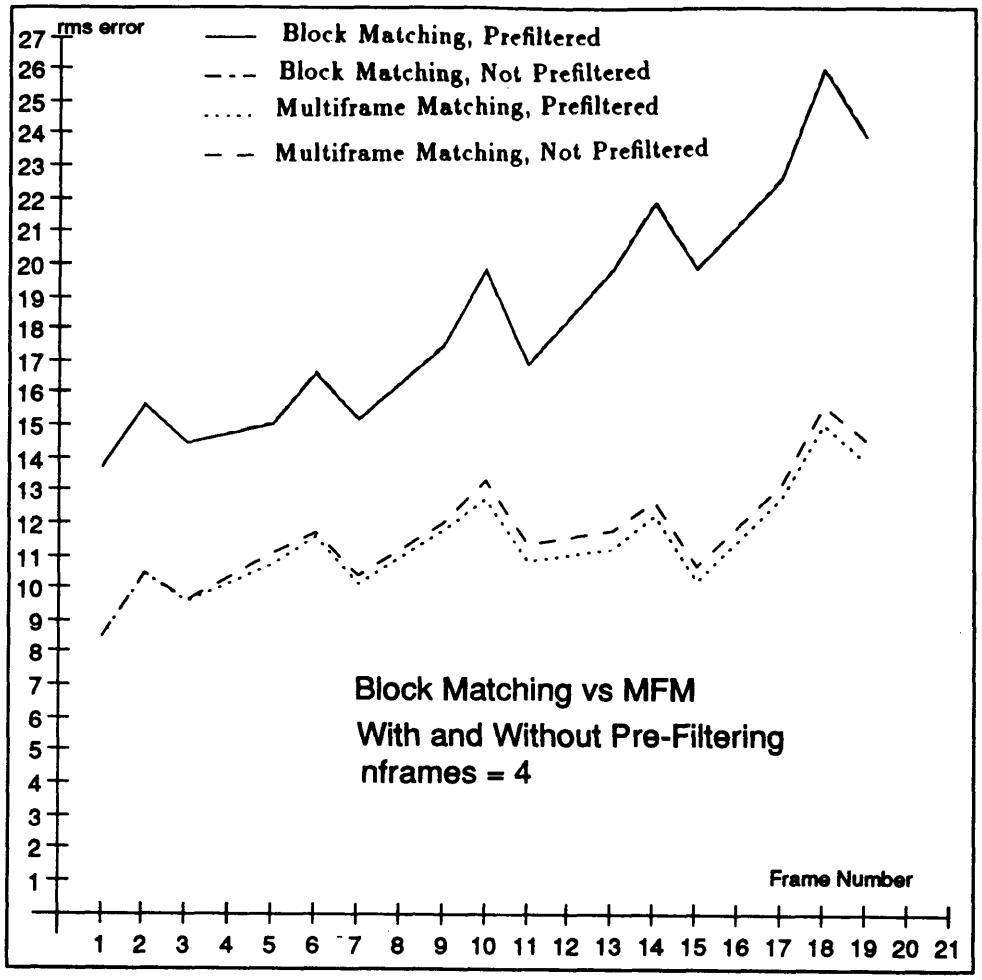

Figure 4.7: Comparison of Multiframe matching vs. Block matching with frame spacing = 4, with and without pre-filtering

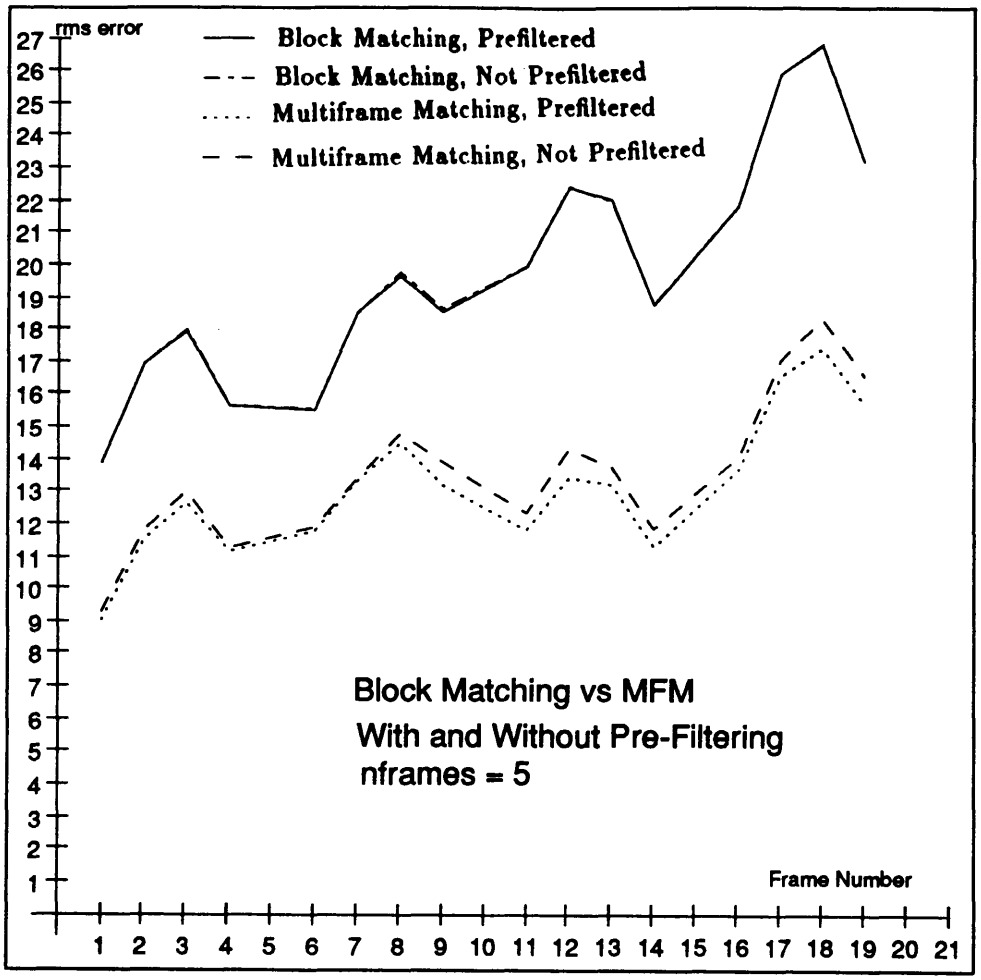

Figure 4.8: Comparison of Multiframe matching vs. Block matching with frame spacing = 5, with and without pre-filtering

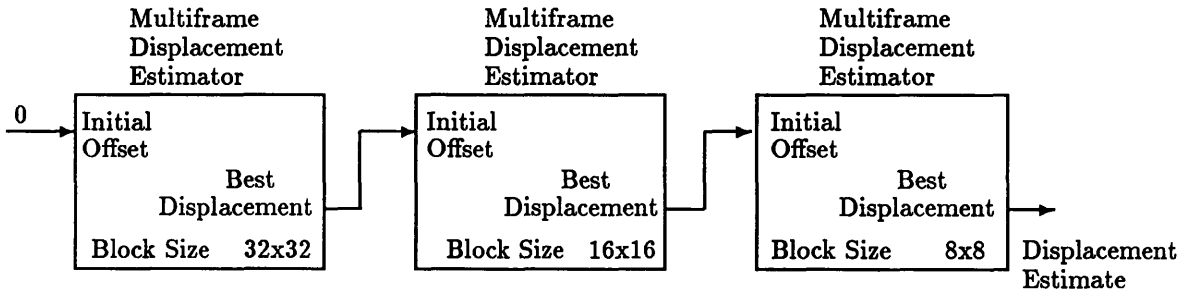

Figure 4.9: Hierarchical multiframe matching displacement estimator

An hierarchical approach is used to overcome these problems. An initial motion estimate is calculated using a large window size (say 32x32) with one vector for each 3-D volume (typically 8x8x$nframes$). This initial search is done over a large search area (covering $\pm$ 60 pels) with a course spacing (4 pels horizontally and vertically).

These displacement estimates are then used as an initial starting point for a finer search using smaller window sizes (see Figure 4.9). This is continued until the desired granularity is reached. By using small window sizes over a small search area, a fine, yet accurate search can be obtained.

4.4 Prefiltering

Prefiltering using a lowpass filter improves the performance of the motion estimation or prediction [9, 20, 27, 29, 39]. The spatial lowpass filtering has several effects.

- It can reduce the chance of convergence onto a local minimum by smoothing the distortion function.
- It reduces the errors due to noise in the image. This is a "Wiener Filter" effect [29].
- The image can be subsampled within the search window without the risk of aliasing due to sampling below the Nyquist rate.

Simple filters have been shown to be sufficient for prefiltering prior to a displacement estimate[9]. An equally weighted sum of all the samples over a small region of support (a box filter) can be used. Although the frequency response of the box filter does not closely approximate that of an ideal lowpass filter [67], it is adequate for the purpose of region matching.

Computationally, the expense of a boxfilter is very low, approximately one addition and one subtraction per sample in each direction, horizontal and vertical, and one multiplication for normalization. The efficient implementation is carried out as follows: For a boxfilter of length N, the sum S of the first $\frac{N}{2}$ samples x is calculated. This sum is doubled to effectively mirror the samples at the edges. Each output point y_i is then equal to $S + x_{i+\frac{N}{2}} - x_{i-\frac{N}{2}}$. Then the next input data $\frac{N}{2}$ samples ahead is added to the sum and the previous data $\frac{N}{2}$ samples away is subtracted. This gives the sum of the N samples around the output point. This process is repeated in the vertical direction, and each point is weighted by $\frac{1}{N^2}$. In the case where N is a power of two, the weighting can be implemented by shifting. Note also that the computation is nearly independent of the filter length.

The reduction in interpolation error when prefiltering prior to the motion estimation can be dramatic. Figures 4.5 through 4.8 show curves of RMS error for keyframe spacing of two through five respectively. Keyframe spacing of two has a single interpolated frame between the keyframes; keyframe spacing of three has two intermediate frames; etc.

Chapter 5

Coding System for High Quality/Low Bit Rate Coding of Motion Sequences

In this chapter, I propose a system for the coding of image sequences which provides high quality at low bit rates. The receiver in this system is easily scalable in terms of spatial resolution, temporal resolution, bit rate and decoding complexity. Additionally, unlike predictive motion compensated systems, random access (with an atomicity of four frames) and variable frame rate play-back are easily incorporated.

5.1 Overview

I propose a motion compensated interpolation system for the transmission of images as shown in figures 5.1 and 5.2.

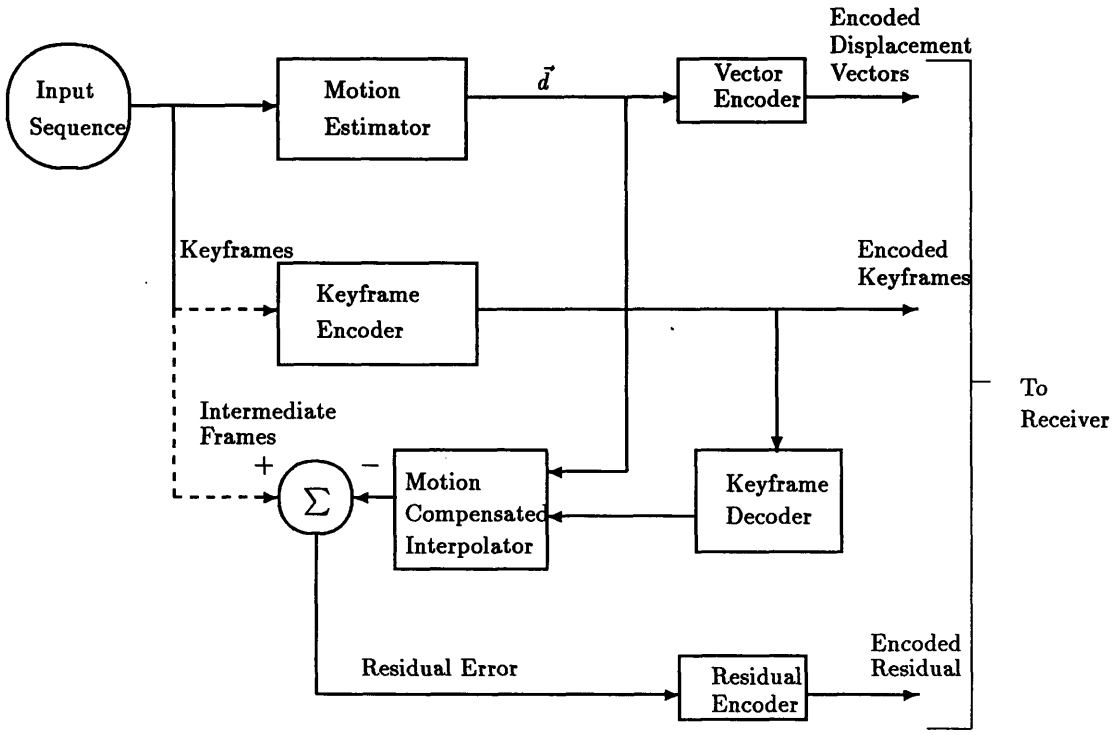

Figure 5.1: Motion compensated interpolation system coder

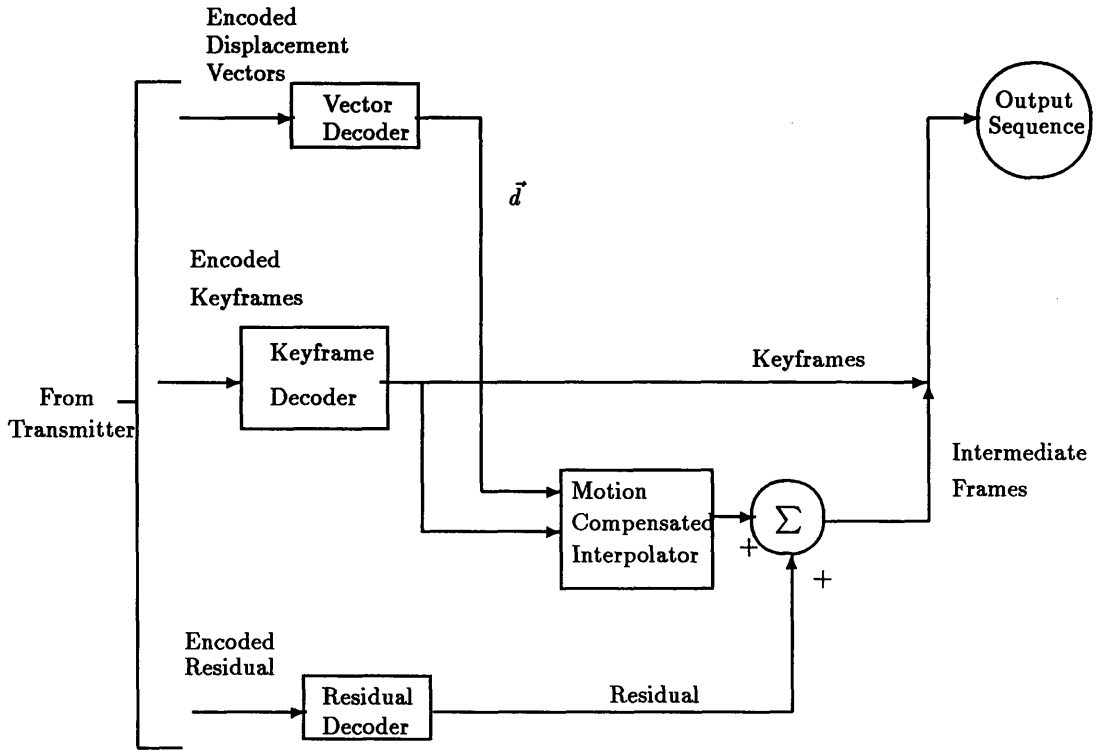

Figure 5.2: Motion compensated interpolation system decoder

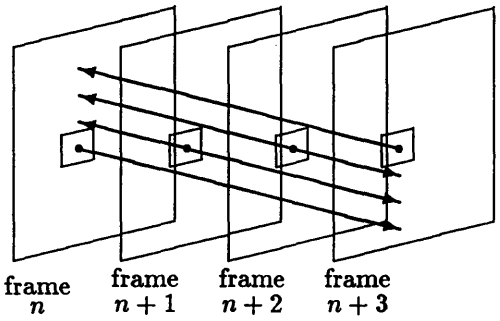

Figure 5.3: Multiframe Matching. Frames n and $n+3$ are keyframes. Data from the keyframes and intermediate frames are used in the pattern matching.

To obtain the velocity vector field, the hierarchical MFM technique discussed in chapter 4 is used. MFM differs from conventional block matching in that the intermediate frames are also used in the pattern matching search (Figure 5.3). This results in a more accurate vector field with reduced error variance over block matching [26]. The accuracy and speed of the motion estimator can be improved by first segmenting the images into moving and non-moving areas [71].

The MFM search is extended over a hierarchy of successively smaller block sizes. A motion estimate is made using a large block size (32x32) for the initial search. This gives a somewhat coarse estimate of the displacement, but is less prone to false matches than a smaller block size [9]. This estimate is then used as an initial estimate for the next level of the hierarchy, using smaller block size for a finer displacement estimate. This continues through as many levels of the hierarchy to reach the desired fineness of the displacement estimate. Subsampling within blocks can be used to reduce computations without a great loss in accuracy of the displacement estimate [35, 79].

The actual pattern matching search used is an extension of the three step search to an arbitrary number of steps, which I refer to as a "K-step search". The search is extended to K steps reducing the spacing between search windows logarithmicly, on each succeeding search step (figure 5.4). The

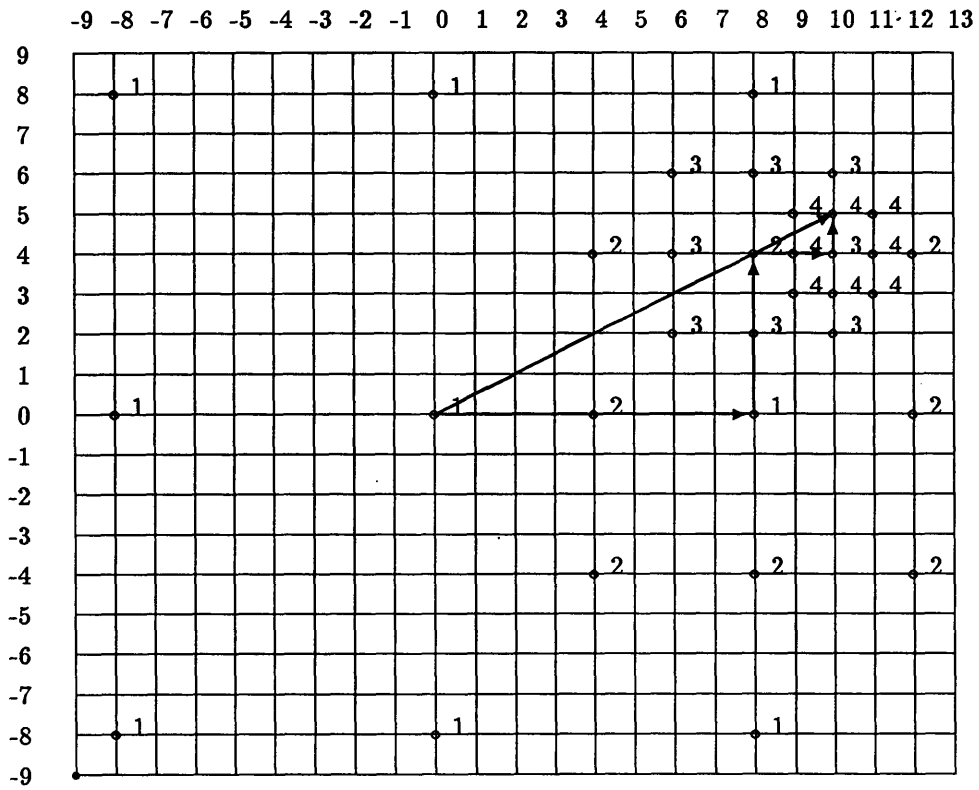

Figure 5.4: K-step search pattern

first search step has a maximum displacement

$$d_1 = d_{min} 2^{k-1} \tag{5.1}$$

where d_{min} is the step size at the final search step. This gives a maximum displacement of

$$d_{max} = d_{min} \sum_{i=1}^{k-1} 2^i \tag{5.2}$$

for a given level of the hierarchy.

Sub-pel accuracy can improve the performance of the motion compensated system [29]. However

the improvement that subpel accuracy yields is diminished as the signal to noise ratio decreases [29].

It has been reported [25] that the error signal from a motion compensated prediction system does not code well using DCT coding. 'Standard' DCT coding algorithms presume that the D.C. and lower spatial frequencies contain the most important information and are thus most finely quantized, while the higher frequencies are more coarsely quantized. Residual signals however, have a higher proportion of the energy distributed in the middle and high frequencies, and ideally have zero D.C. energy. In this system the keyframes and residuals are decomposed using a pyramid transform followed by vector quantization. The variance or the residual signal varies greatly from frame to frame (see chapter3). For this reason the frames are normalized before pyramid decomposition and vector quantization.

5.2 Preprocessing

The color space YUV was chosen for this work. Input signals in RGB format are converted to YUV using a matrix multiplication. The color components U and V are lowpass filtered using a Gaussian filter and subsampled spatially by a factor of four. The luminance signal, Y, is coded at full resolution.

If the input signal is in YUV format, then the only preprocessing necessary is to decimate the chrominance signals to one quarter of the luminance resolution.

5.3 Encoder

Figure 5.1 shows a block diagram of the encoder. The interface to the channel or storage device is not shown. Depending on the requirements of the storage device, there may be a buffering system to maintain a constant data rate.

5.3.1 Motion Estimator

The motion estimation is performed using the original (uncoded) keyframes and intermediate frames. The hierarchical multiframe matching technique discussed in section 4.2 is used for motion estimation. The motion estimation is performed with only the luminance signal; chrominance is not used.

5.3.2 Keyframe Encoder

The keyframe encoder [Figure 5.5] consists of a pyramid subband decomposition filter, a vector quantizer, and an arithmetic coder.

The pyramid decomposition is performed by recursively applying Quadrature Mirror Filters (QMF) to split the image into spatial octave subbands (Figure 5.6). Each successive level splits the lowest spatial frequency band into four more subbands.

The vector quantizer then encodes each subband separately or in groups. The outputs of this block are code books and look up tables (LUTs). The LUTs are determined for a one second block of keyframes. A new LUT is transmitted for each one second block. In this way, the cost (in bits) of the LUTs is amortized over a one second period. The multilevel arithmetic coder follows to losslessly compress the output of the vector quantizer.

5.3.3 Motion Compensated Interpolator

To obtain the interpolated intermediate frames, the keyframes are decoded as described below. The decoded keyframes are used together with the motion vectors to interpolate the intermediate frames. These interpolated intermediate frames are subtracted from the original to obtain the residual frames. The residual frames are coded in the same manner as the keyframes - a subband pyramid followed by the arithmetic coder.

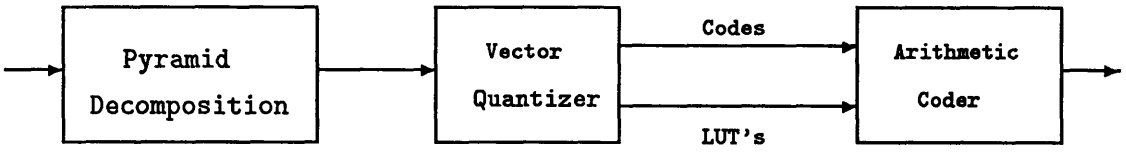

Figure 5.5: Block diagram of keyframe encoder.

5.4 Decoder

The decoder for this image system is shown in figure 5.2. The displacement vectors are decoded using the inverse of the arithmetic coder. The vector quantized subbands and LUTs for the keyframes and intermediate frames are also decoded through this arithmetic decoder. Each subband has a separate LUT associated with it. These LUT's are completely updated once per second.

The lowest level subbands are then reconstructed through their corresponding LUT. These four bands are then interpolated and summed to synthesize level 3 of the pyramid into the level 2 LL band (Figure 5.7), where LL refers to the horizontal and vertical lowpass filtered band.

The remaining three bands for level 2 are decoded and reconstructed, and together with level two LL band, they are interpolated and summed to synthesize the level one LL band. Level one is likewise decoded and interpolated into the full size image.

It is possible to interpolate up to any level without all of the subbands being present, as may happen on a heavily used network. In this case, zeros are input to the subband interpolation filter in the place of that band. This will cause the image to degrade somewhat, but the image will not be lost all together.

At the beginning of a sequence, two keyframes must be received and decoded prior to beginning the motion compensated interpolation. Once these two keyframes are decoded, only one additional keyframe is needed for each set of N intermediate frames.

The displacement vectors are applied to the decoded keyframes to recreate the intermediate

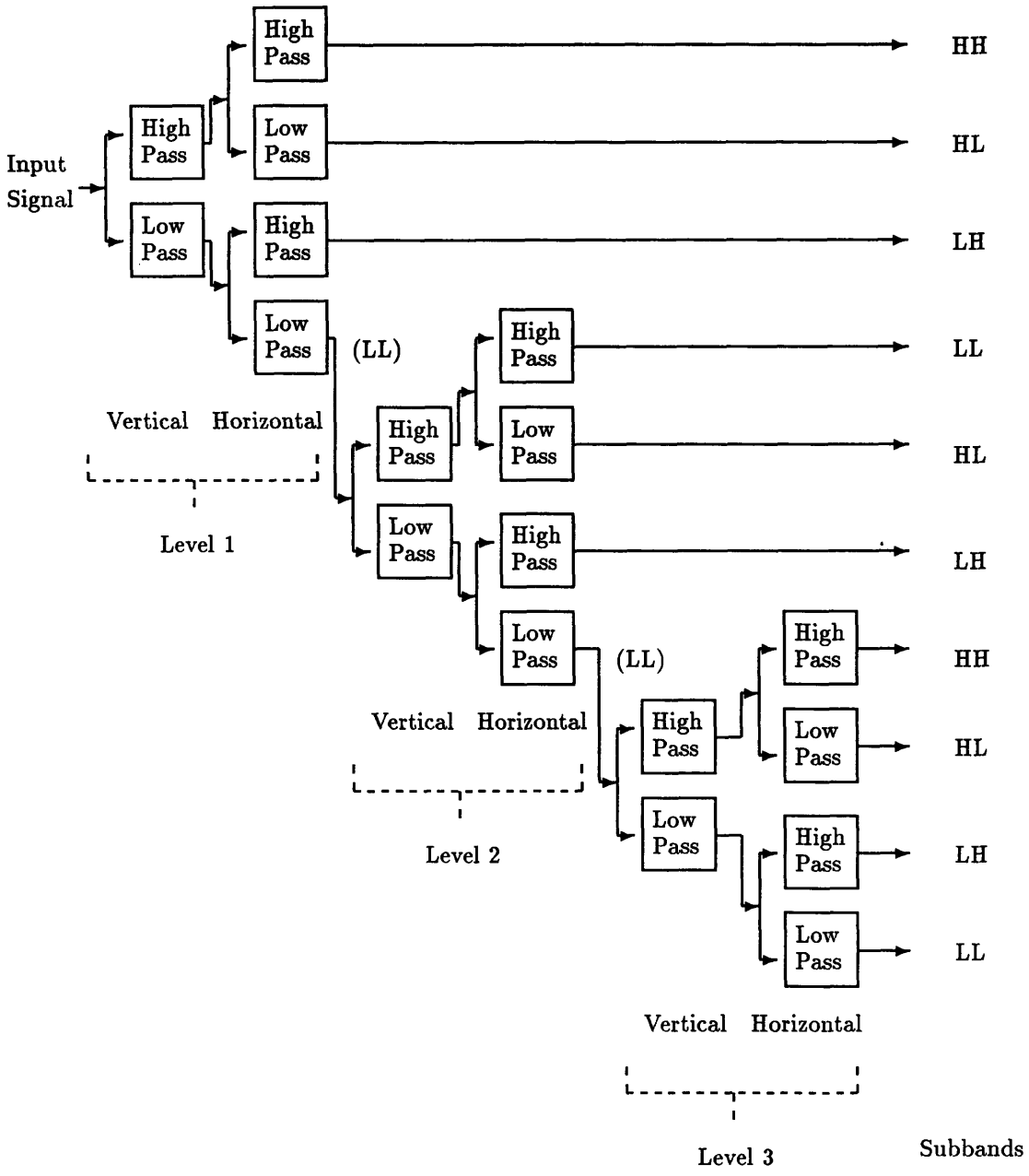

Figure 5.6: Block diagram of pyramid analysis QMF filters.

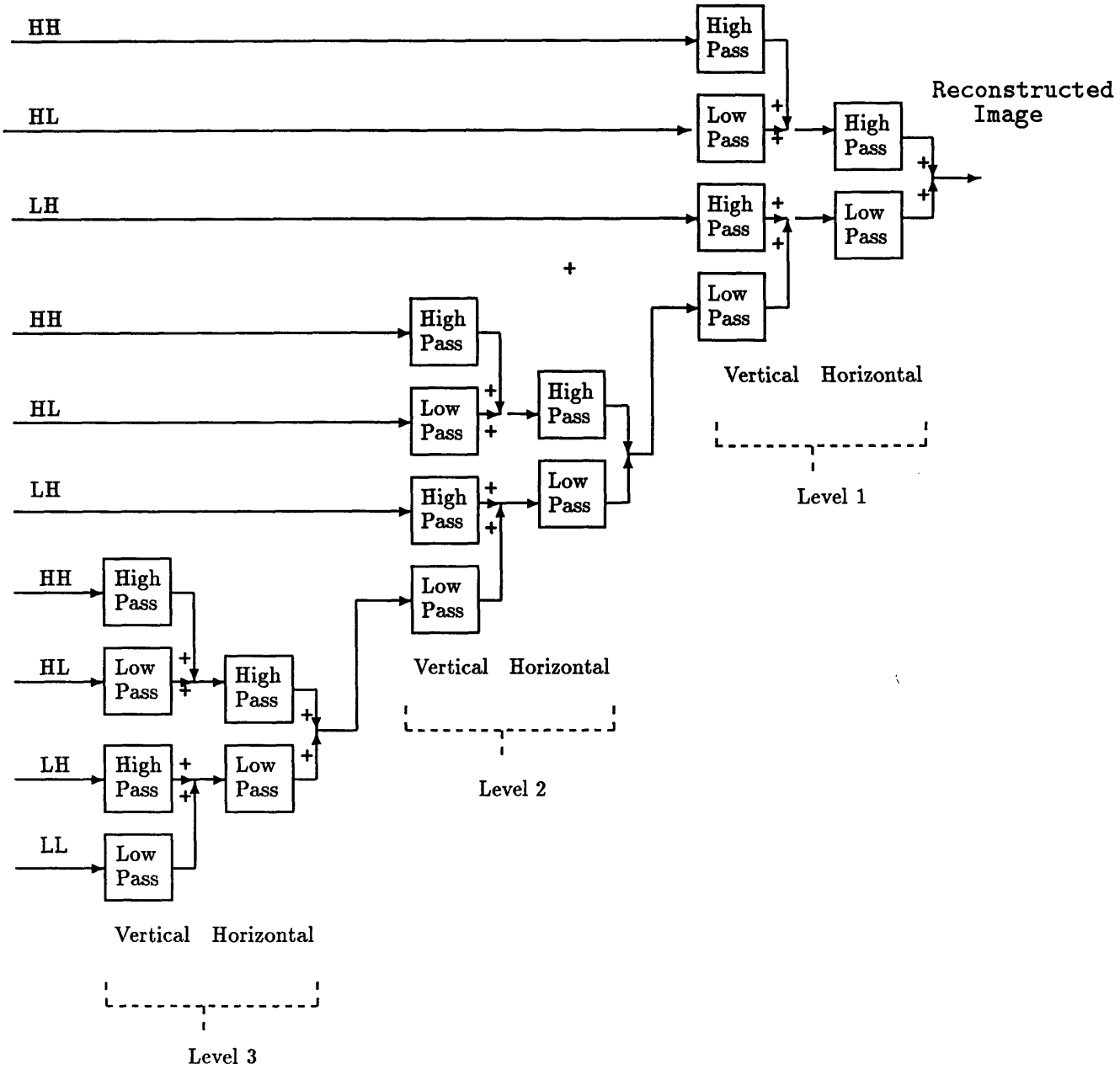

Figure 5.7: Block diagram of pyramid synthesis QMF filters.

frames. This will result in intermediate frames with some interpolation error. To reduce this error, the residual signal which has been processed by the pyramid decoder is added to these intermediate frames resulting in the decoded luminance signal for each output frame.

5.4.1 Chrominance Decoding

The chrominance signal is decoded in a fashion similar to the luminance. The UV frames are decoded by using the transmitted codewords to index into the LUTs. The chrominance information is then upsampled and interpolated from one quarter resolution horizontally and vertically to full resolution. Motion compensation is not applied to the chrominance signal in this system. Using motion compensation on the chrominance signals tends to exaggerate any motion artifacts in the luminance signal.

5.4.2 Postprocessing

The final step is to convert the decoded luminance/chrominance signals to the color space and frame rate of the output device. Color conversion is performed by a matrix multiplication (see section 2.8). If the original source was a 24 fps motion picture, and the display device is 60 Hz., the frame rate can be converted using 'three two pull down', a method which will alternately replicate frames from the 24 fps sequence two and three times to output a 60 Hz. sequence.

Chapter 6

Computer Simulations

Computer simulations of the system described in Chapter 5 were developed using the 'C' programming language.

The coding system was simulated on a test sequence *Alley*, a 24 fps non-interlaced RGB sequence of two figures in an alley in animated conversation, with one figure crossing in front of another. The processed image size is 640 pels horizontal by 480 pels vertical.

The *Alley* sequence presents the challenge of correctly detecting rapidly moving arms and legs, reflections of moving objects, occluding figures, panning and zooming.

6.1 Preprocessing

The input sequences were converted from RGB color space to YUV using the equations:

$$Y = 0.299R + 0.587G + 0.114B \tag{6.1}$$

$$U = \frac{B - Y}{2.03} \tag{6.2}$$

$$V = \frac{R - Y}{1.14} \tag{6.3}$$

or in matrix form, see equation (2.15).

6.2 Motion Estimation

Keyframe spacing of 4 for *Alley* was used for these simulations. These numbers were chosen as a reasonable trade off between reduced keyframe rate and increased interpolation error. This spacing corresponds to a time interval between keyframes of 167 milliseconds.

The multiframe search used the parameters in Table 6.1. The number of search stages is the number of levels in the hierarchical search. X and Y direction window spacing is the block size in the interpolated image that is assigned one displacement vector. In this case, each 8x8 block is given a displacement vector.

Window size X and Y are the size of the window used in the displacement estimation search for each level of the hierarchical search. The first level used a 32x32 window, the second level a 16x16 window, and the last level an 8x8 window. Subsample factor is the rate of subsampling within the search window, ie. the first level search uses subsampling by four both horizontally and vertically.

The number of search steps and minimum search step size determine the range of the logarithmic search pattern. For example, the first level of the search hierarchy performs the motion estimation in four steps with the minimum displacement of four pels. The first step will search over $\pm$ 4 * (1 * 2 * 2 * 2) = $\pm$ 32 pels, the next step over $\pm$ 4 * (1 * 2 * 2) = $\pm$ 16 pels, etc. This gives a maximum search area through the three levels of the hierarchy of $\pm$ [(4+8+16+32) + (2+4) + (1+2)] = $\pm$ 69 pels. Movement in real scenes rarely exceeds this displacement.

The thresholds at the bottom of Table 6.1 are explained in section 6.2.1 below.

number of search stages	3 levels		
x direction window spacing	8 pels		
y direction window spacing	8 pels		
window size x	32 pels	16 pels	8 pels
window size y	32 pels	16 pels	8 pels
boxfilter length	33 pels	5 pels	3 pels
subsample factor x	4	2	1
subsample factor y	4	2	1
number of search steps	4	2	2
minimum step size	4 pels	2 pels	1 pels
average pel difference motion threshold	6		
pel difference over threshold count	18		

Table 6.1: Parameters used in the multifield matching motion estimation search.

6.2.1 Segmentation

Prior to beginning the displacement estimate the images are segmented into moving and non-moving areas based on 8x8 blocks. The classification is done as follows. Each pel in the first keyframe is compared to the corresponding pel in the next keyframe. If the absolute value of this difference exceeds a threshold, a counter is incremented. If the counter exceeds a maximum the block is declared moving. A threshold of 6 and a maximum count of 18 were used for these simulations. The segmentation is done over a temporal block, eg. all blocks in the intermediate frames with the same x,y location have a single moving/non-moving determination. A bar graph of the percentage of non-moving areas for the sequence *Alley* is shown in Figure 6.1.

The motion estimation is performed only on those blocks which are classified as moving. This segmentation has two effects:

1. Search time is improved significantly since only moving areas are searched.

2. False motion around moving objects is reduced, resulting in a more accurate interpolation.

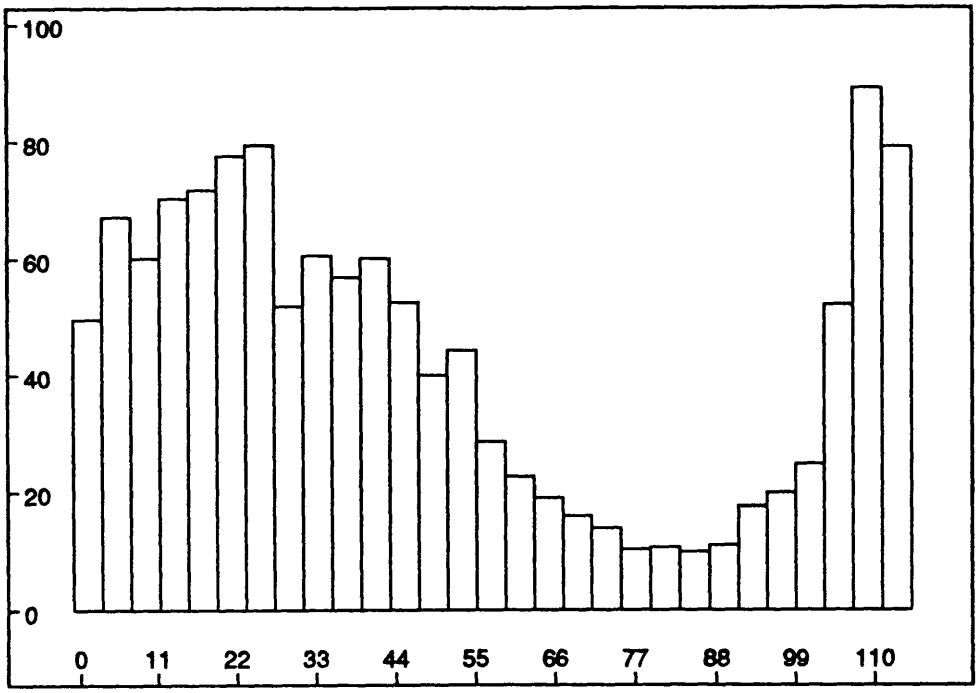

Figure 6.1: Graphical illustration of the percentage of the frame area classified as non–moving in *Alley* sequence.

6.2.2 Prefiltering

The next step is to filter the images before performing the motion estimation search. This is done to reduce the possibility of converging on a local minimum of the matching criterion and to force the cross correlation functions to act in a more isotropic manner. The filtering also reduced low level noise present in the image. The lowpass filtering is performed using a simple two dimensional box filter. This gives adequate performance at very low computational cost. Each sample requires only four operations (two additions and two subtractions) regardless of the filter length (Section 4.4).

The original images are filtered before each level of the hierarchical search. The length of the two dimensional filter is determined by the maximum search distance for a given level of the hierarchy. The length used was

$$L = \left(\text{minstep} * 2^{(\text{numstep}-1)}\right) + 1 \tag{6.4}$$

For this simulation that gives filter lengths of 33,5, and 5 for the three levels of the hierarchy.

6.2.3 Displacement Search

The motion estimation is carried out on the uncoded luminance signal. The first level of the hierarchical search is performed on 32 x 32 blocks extending ± 32 pels, with a minimum granularity in the search of four pels. During this initial estimate, the blocks are also subsampled by a factor of four. By subsampling within the search window, this considerably speeds up the search without affecting the accuracy of the displacement estimate significantly.

The next level of the hierarchical search is performed using 16x16 windows. The estimate from the previous search is used as the initial estimate for this search. This is performed over ± 4 pels with a minimum granularity of two pels. Subsampling by a factor of two within the window is used at this level.

The final search is performed using 8x8 windows. This search extends over ± 2 pels in single pel intervals. No subsampling is used at this level. The motion vectors obtained are then coded using the arithmetic coder (section 6.6) before transmission.

A visual display of the displacement vectors superimposed on the original frames is shown in Figures 6.2 to Figure 6.6.

6.3 Keyframe Coding

The keyframes are coded in one second segments, eg., six keyframes at a time. This gives a reasonably small atomicity for random access, while allowing the use of the redundancies between keyframes to reduce the size of the LUTs. The cost in bit rate of transmitting the LUTs for the six keyframes is amortized over a one second period.

The keyframes are decomposed into a three level pyramid using the QMF filters plotted in Figures 6.7 and 6.8, and listed in Appendix B, B.1. These analysis filters are separable two-dimensional symmetric filters of length 9. The two highest levels (Level 1 and 2) are noise cored before further processing. The noise coring is a simple thresholding; if the absolute value of a sample is less than three, it is replaced by zero. The noise coring considerably reduces the size of the LUT and increases the efficiency of the k-d tree search. There are many samples near zero that are relatively unimportant. The noise coring avoids this low level noise being split into many regions increasing the code book size. A typical keyframe pyramid decomposition is shown in Figure 6.9.

Next, the code books are calculated using an error limit based k-d tree search. This simulation coded each subband separately; no joint redundancies between subbands at a given level are utilized. It was found that the code book size increased significantly when the subbands were jointly coded eg., there is little correlation between the subbands at a given level.

Level one is coded using 8x8 blocks of pels, level two using 4x4 blocks, and level three is scalar

Figure 6.2: Displacement vectors for *Alley* sequence frames 1 to 3

Figure 6.3: Displacement vectors for *Alley* sequence frames 25 to 27

Figure 6.4: Displacement vectors for *Alley* sequence frames 49 to 51

Figure 6.5: Displacement vectors for *Alley* sequence frames 73 to 75

Figure 6.6: Displacement vectors for *Alley* sequence frames 96 to 99

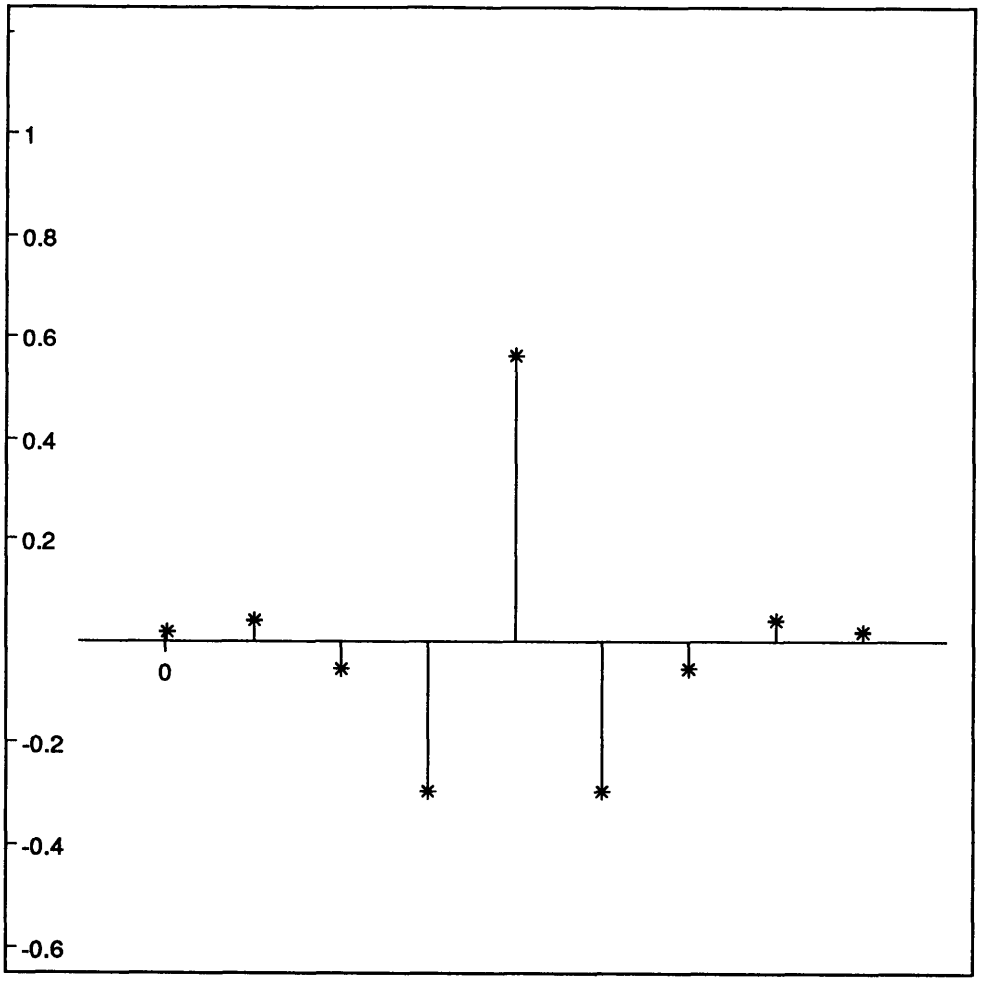

Figure 6.7: Lowpass filter used both horizontally and vertically in subband pyramid analysis

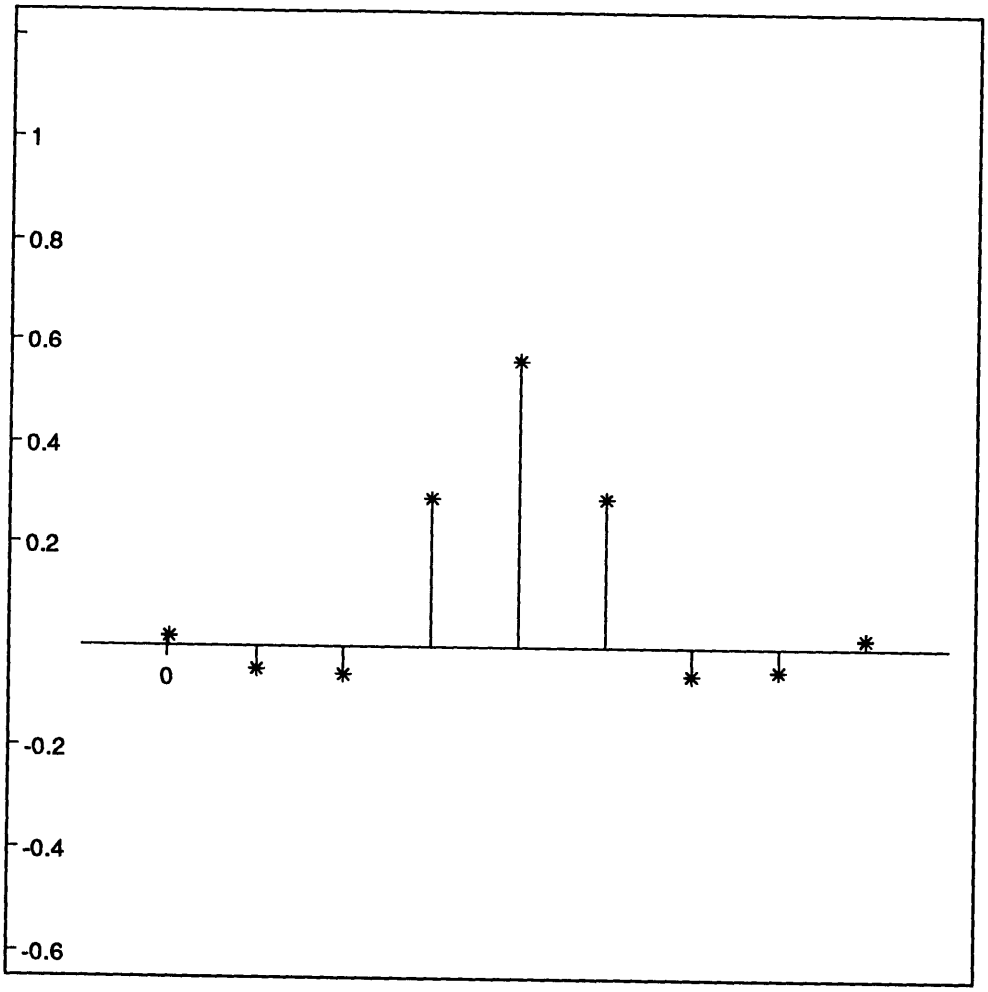

Figure 6.8: Highpass filter used both horizontally and vertically in subband pyramid analysis

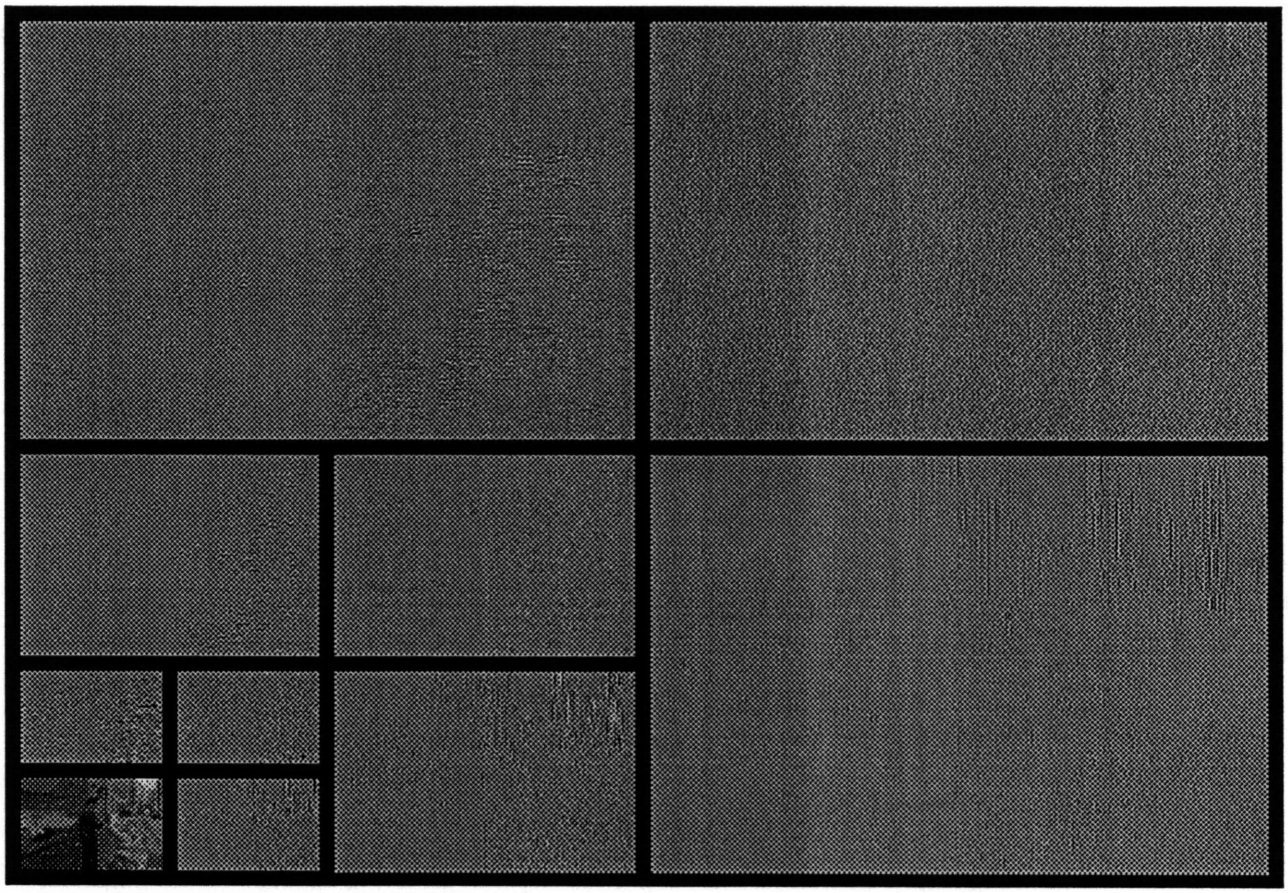

Figure 6.9: A typical keyframe pyramid decomposition from the *Alley* sequence

level	block size	select	split	bound			
				HH	HL	LH	LL
1	4x4	peakerr	mean	maxe mse 8.0	maxe mse 8.0	maxe mse 8.0	na
2	2x2	peakerr	mean	maxe mse 20	maxe mse 20.0	maxe mse 20.0	na
3	1x1	mse	mean	psnr 44	psnr 44	psnr 44	psnr 48

Table 6.2: Error limits used in coding the keyframes for *Alley*

level	block size	select	split	bound			
				HH	HL	LH	LL
1	4x4	peakerr	mean	maxe mse 23.5	maxe mse 23.0	maxe mse 23.0	na
2	2x2	peakerr	mean	maxe mse 23.5	maxe mse 23.0	maxe mse 23.0	na
3	1x1	mse	mean	el psnr 34.0	el psnr 34.5	el psnr 43.5	el psnr 48.0

Table 6.3: Error limits used in coding the residuals for *Alley*

quantized. The parameters used are summarized in Table 6.2. In this table, 'Select' refers to the method used to select the branch at each split point. 'Peakerr' selects the branch with the lower peak error, and 'mse' selects the branch with the lower mean square error. 'Split' is the method used to split the vector space. 'Mean' splits at the geometric mean. Termination of the splitting is determined by 'bound'; splitting ceases when the error metric falls below the bound. 'Maxe mse' refers to maximum error measured as a mean square. 'Psnr' refers to peak signal to noise ratio.

The bounding criteria of maximum error was chosen for the higher levels (levels 1 and 2) of the pyramid. These higher frequency bands contain spatially localized information such as edges. The 'maxe' criteria will assign code words to these higher amplitude blocks, preserving more important information. The peak signal to noise bound used for the lowest levels will maintain good quality for these important bands.

The keyframes are rendered using the code books calculated with the k-d tree search. The indices (codes) and LUTs are entropy coded using the arithmetic coder before being sent to the channel.

The keyframes must be decoded at the transmitter before MCI and residual coding. The rendered keyframe subbands are decoded by indexing into the LUTs. The keyframes are then synthesized from the three levels of the pyramid using separable symmetric analysis filters as in Table B.1.

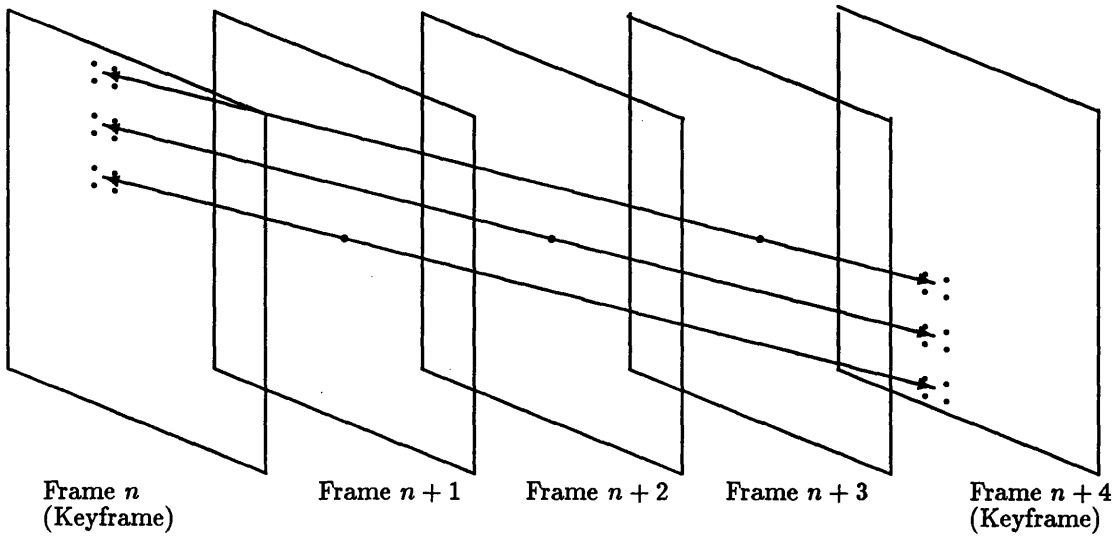

Figure 6.10: Interpolated intermediate frame pels from bilinearly interpolated pels in the keyframes

These decoded keyframes used in the transmitter are identical to the ones that the receiver will decode and use for its MCI.

6.4 Calculating Residual Frames

The keyframes are now used for motion compensated interpolation of the intermediate frames. The motion vectors are applied to each 8x8 block of the keyframes to obtain the interpolated frames.

Since a single motion vector is used to interpolate blocks in each of the three intermediate frames, they are divided into forward and reverse pointing vectors. This division creates fractional parts in the displacements. To accommodate this, each sample in the keyframes is two dimensionally bilinearly interpolated from the four nearest pels (Figure 6.10). The subpel precision resulting performs better than rounding the divided displacements to integer precision. These bilinearly interpolated samples from each of the keyframes are then weighted according to the distance of the intermediate frame to each keyframe.

Level	Band	Average Energy per Sample	
		Keyframes	Residual
Lev 1	HH	131.393474	2.928426
	HL	134.559582	4.098564
	LH	137.870853	5.743887
Lev 2	HH	238.262517	3.511255
	HL	243.495536	3.955221
	LH	248.691099	4.237471
Lev 3	HH	327.676258	1.808828
	HL	334.162721	2.058322
	LH	340.412076	2.387653
	LL	347.019380	2.842444

Table 6.4: Comparison of the average energy of keyframes and residuals in *Alley*.

The resulting interpolated intermediate frame is subtracted from the original to give the residual frame. Typical interpolated frames and their corresponding residuals are shown in Figures 6.11 through 6.16. The residuals subbands are vector coded in one second temporal sections.

The residual frames contain considerably less energy than the keyframes. The average energy per sample within the subbands from the first second of *Alley* are shown in Table 6.4.

6.5 Residual Coding

The variance of the residuals changes with its distance from the keyframes (see section 3.1). Coding these residuals directly will result in large LUTs and the resulting higher entropy in the codes. Before decomposing the residuals into a pyramid they are normalized to create more similarities between frames. The highest variance among the residual frames is determined for each one second segment. Each sample in the frame is then normalized using

$$p' = p\frac{maxvariance}{framevariance} \tag{6.5}$$

Figure 6.11: Interpolated recreation of frame 1 of the *Alley* sequence

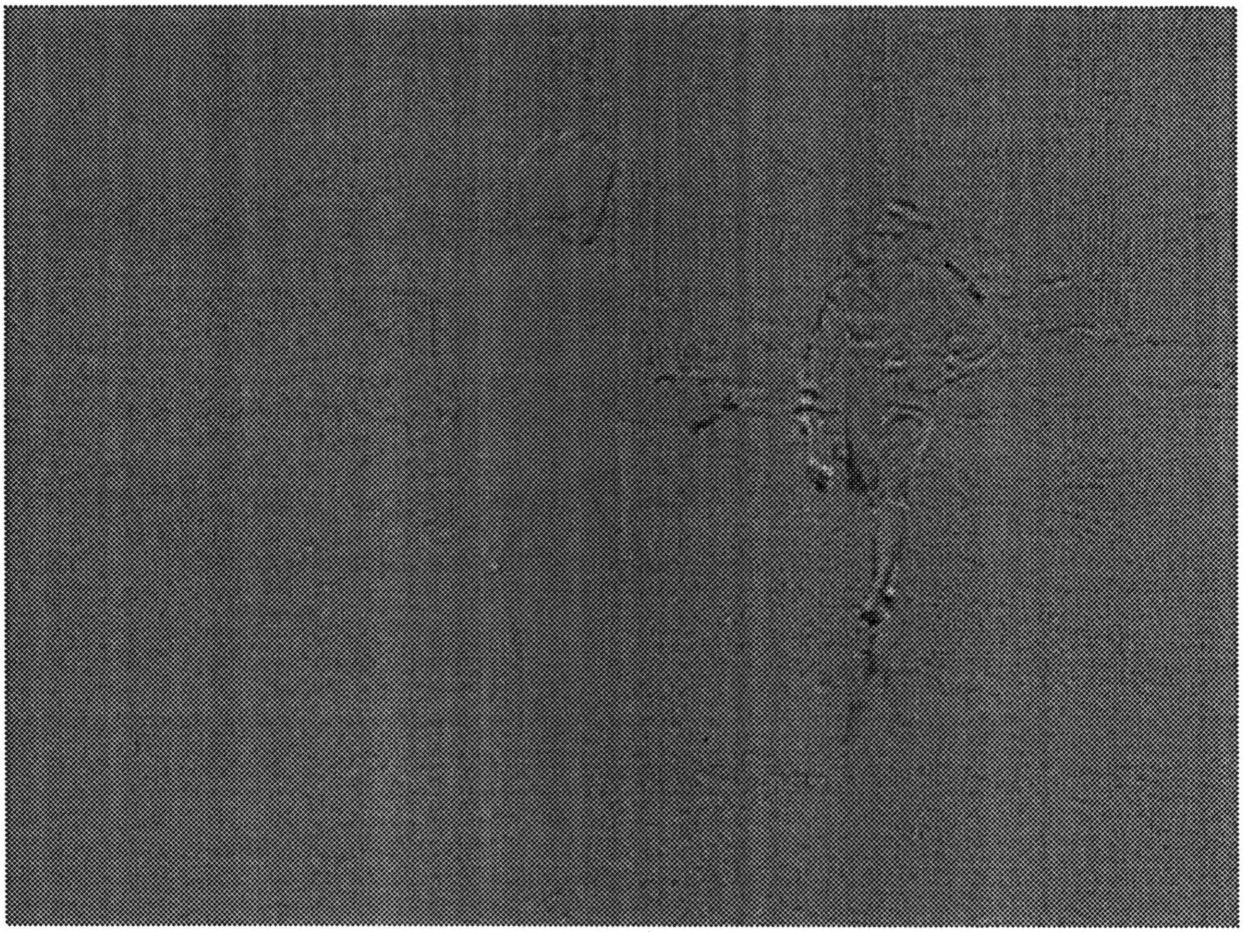

Figure 6.12: Residual from frame 1 of the *Alley* sequence

Figure 6.13: Interpolated recreation of frame 25 of the *Alley* sequence

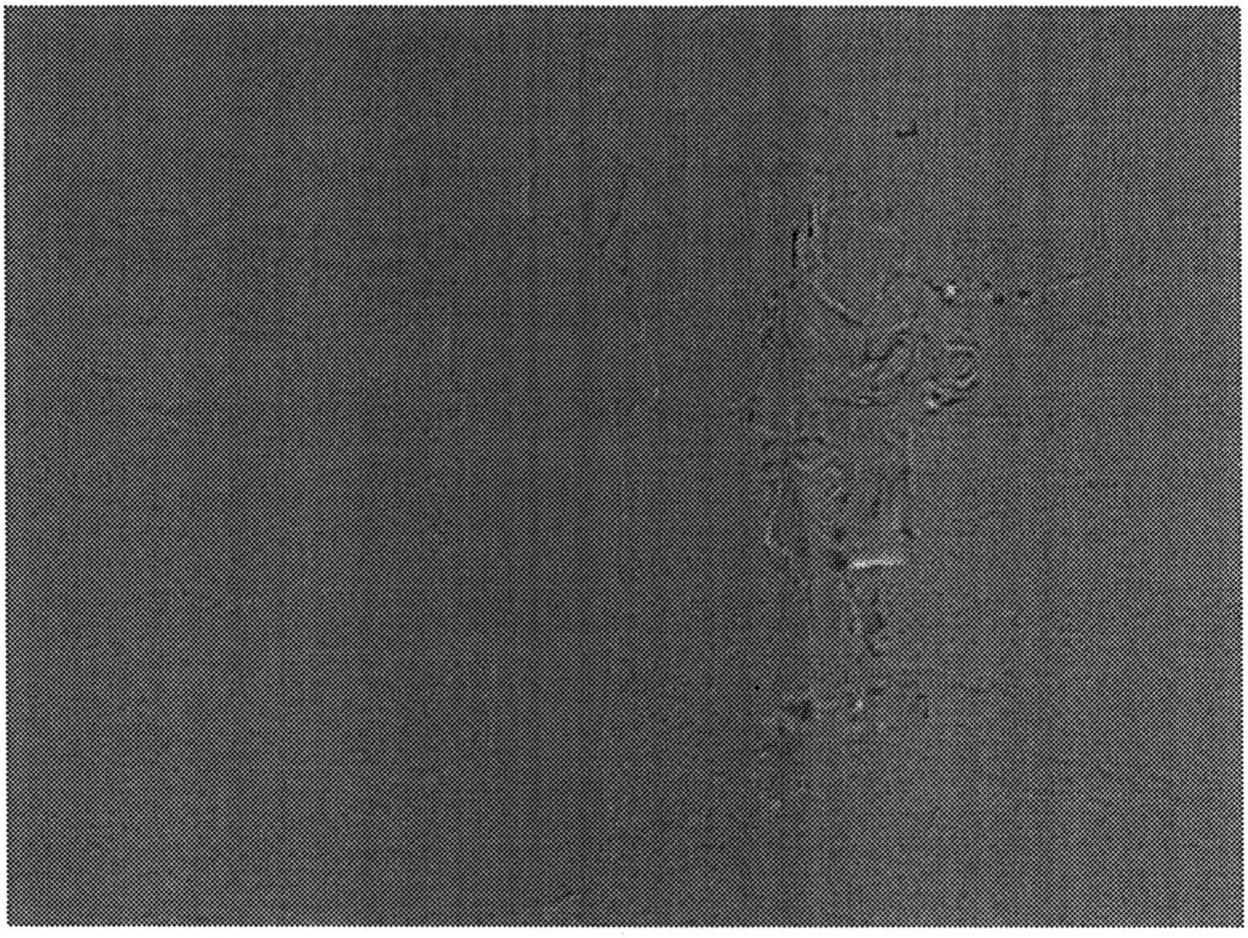

Figure 6.14: Residual from frame 25 of the *Alley* sequence

Figure 6.15: Interpolated recreation of frame 49 of the *Alley* sequence

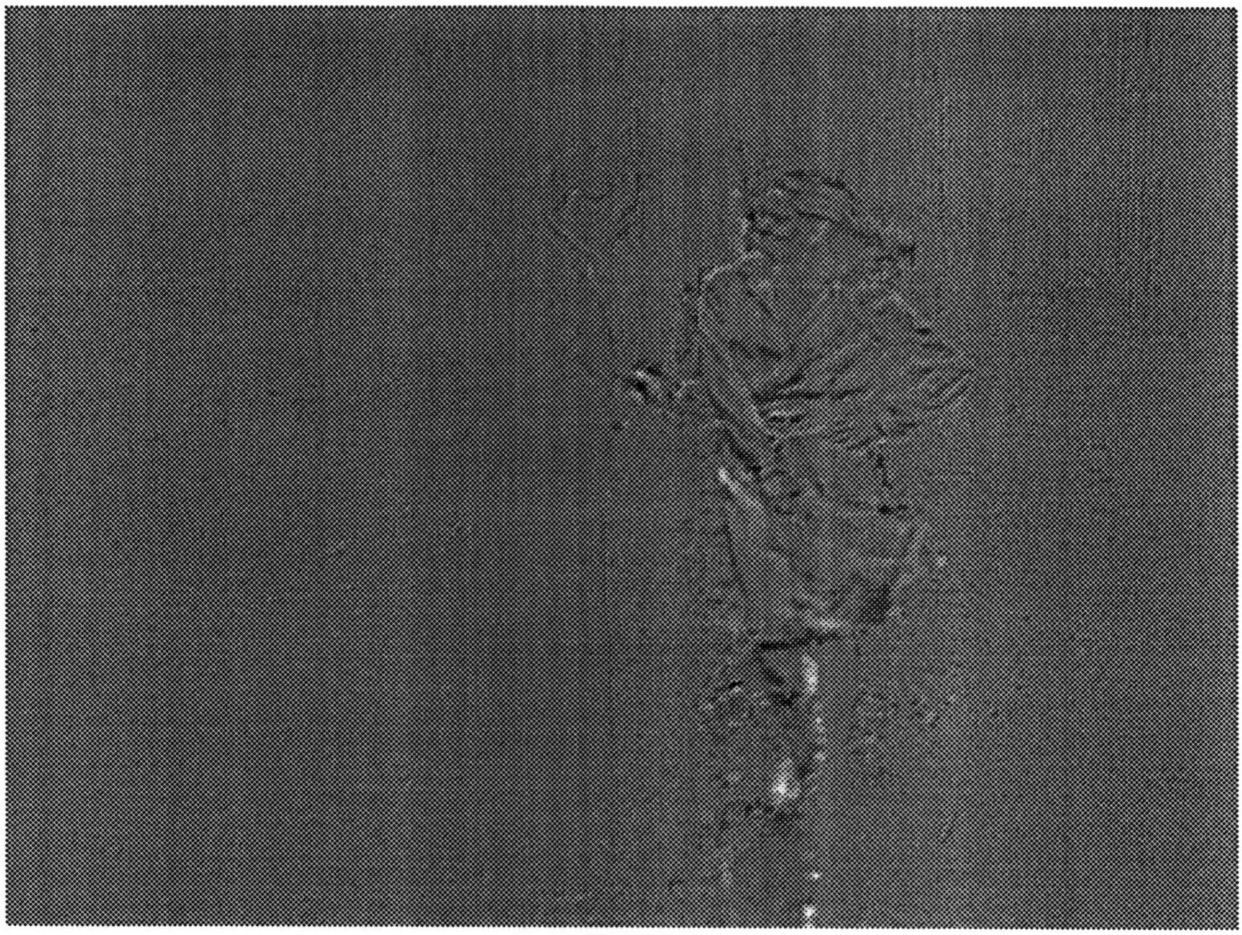

Figure 6.16: Residual from frame 49 of the *Alley* sequence

where p is the sample in the original frame, and p' is the normalized sample.

The normalized residual signals are coded in a manner similar to the keyframes. After normalization, they are decomposed into a three level pyramid using the same analysis filters as the keyframes. Noise coring is performed in the same manner as with the keyframes.

The code books are built using error limits that parallel those used in the keyframe coding. If the intermediate frames and keyframes do not have closely matched noise levels, there will be noticeable flicker at the keyframe rate (6 Hz.). At this temporal rate the HVS is very sensitive to flicker [63]. By keeping similar noise levels in both the keyframes and intermediate frames, flickering is avoided.

6.6 Arithmetic Coding

All the codes, LUTs and vectors pass through the arithmetic coder. A simple first order predictor is used adaptively in the coding of the level 3 bands. The entropy of a one dimensional pel difference is calculated vs. the entropy of the input to the predictive coder. If the entropy is lower, then the subband is put through the predictive coder and then through the arithmetic coder. If the predictive coder output has a higher entropy than the original signal, the subband is put directly through the arithmetic coder. The level 1 and 2 quantized subbands are also sent directly through the arithmetic coder. The saving on the level 3 was in the range of 1 to 2 bits, typically around 1.5 bits. For the level 3 LL, this translates to a savings of about 129 Kbits per second.

The arithmetic coder used is a multilevel arithmetic coder which operates on single ended signals. The number of levels in this coder may be up to 65,535 (16 bits). The source indices (codes) from the vector quantizer are by definition greater than zero, so no preprocessing before arithmetic coding is necessary. The displacement vectors, LUTs and predictive coded level 3 subbands are bipolar, so first they are biased to ensure that the most negative value is mapped to zero.

Both the arithmetic coder and decoder need to model the probability distribution (histogram) of

the source. This biasing reduces the total size of the histogram significantly. It is difficult to develop an accurate model of the distribution of the codes, as any vector can be arbitrarily mapped to any index. For this reason, the histogram is sent as side information from the arithmetic coder. As the histogram is typically very sparse (all unused levels have a frequency of zero), it is sent using run length coding.

The displacement vectors are also coded one frame at a time using the arithmetic coder. The Ziv-Lempel [59, 99, 105] adaptive compression algorithm was also tried, but it was found that the arithmetic coder provided a lower bit rate for the displacement vectors.

6.7 Chrominance Coding

The chrominance U and V signals are first prefiltered and decimated by a factor of four both horizontally and vertically using the decimation filters in Table B.2. A plot of the decimation filter impulse response is shown in Figure 6.17. This removes much of the irrelevant information, as the HVS has reduced acuity to chrominance over luminance.

The decimated chrominance are not decomposed into a pyramid, but are quantized directly. Motion compensation was not used on the chrominance for two reasons. Dissimilarities between chrominance keyframes and intermediate frames would contribute to flickering in the displayed sequence. Also, it was found that if the displacement vectors obtained from the luminance signal were applied to the subsampled chrominance signal (with the vectors appropriately scaled), coding the interpolation error required higher bit rates to code than the subsampled chrominance signal directly. Coding parameters for the chrominance are given in Table 6.5. The code books are generated using the k-d tree method with an error limit of 28.0 dB peak signal to noise ratio terminating the search.

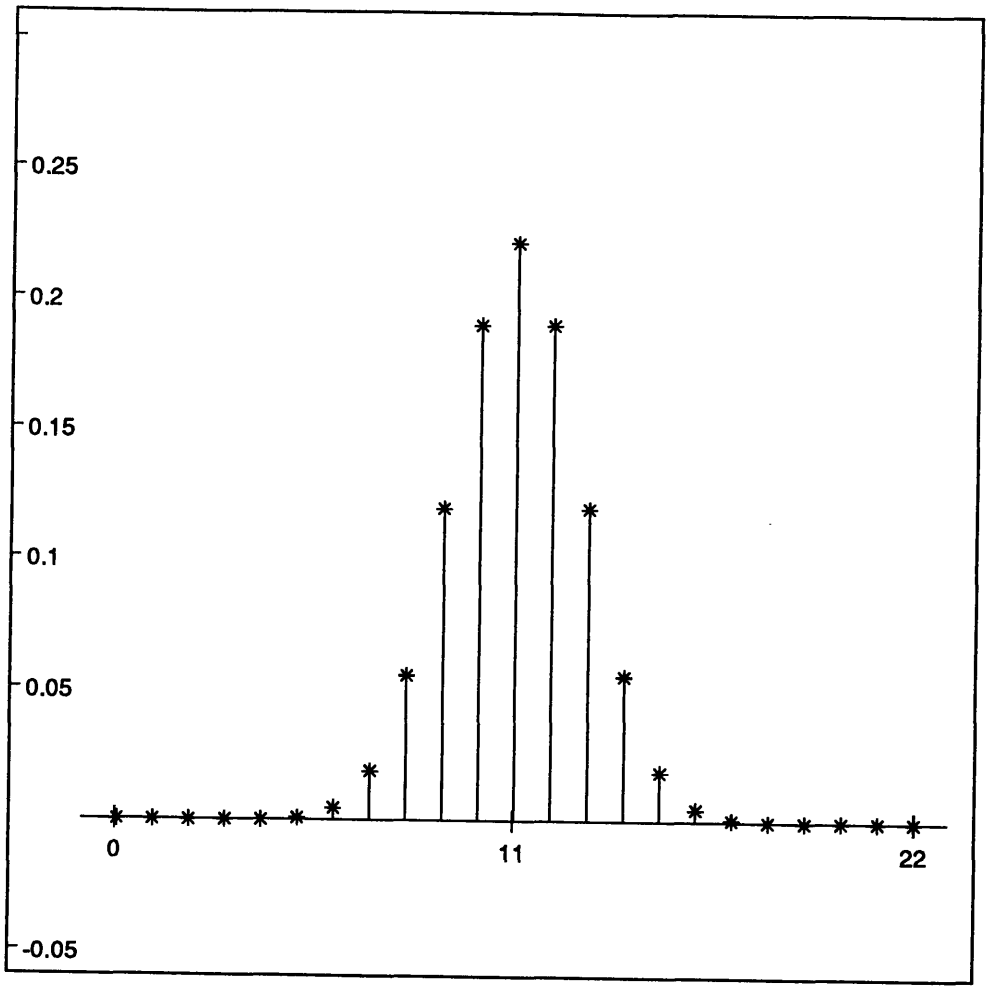

Figure 6.17: Gaussian decimation filter used in chrominance coding

Table 6.5: Error limits used in chrominance coding.

type	block size	select	split	bound
Chrominance	2x2	peakerr	mean	-el psnr 28.0

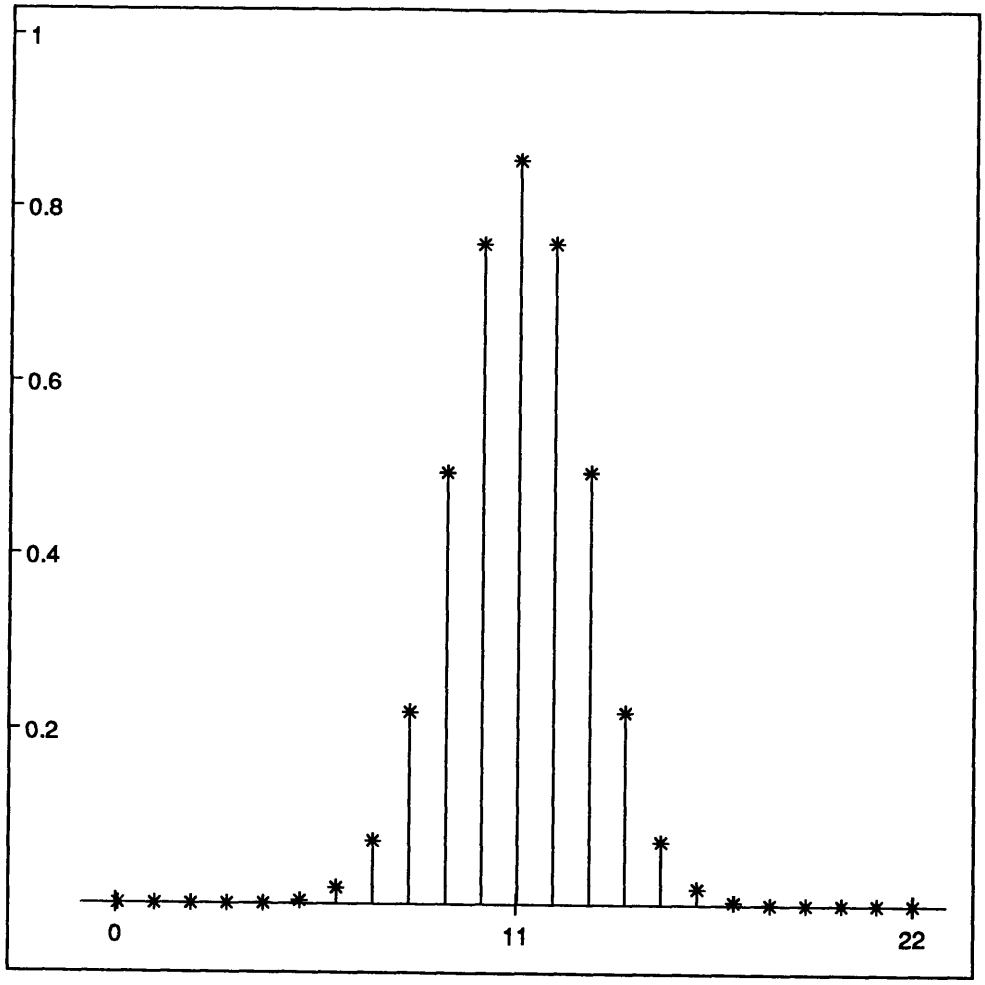

Figure 6.18: Gaussian interpolation filter used in chrominance coding

6.8 Decoding

At the receiver, each one second segment of keyframes is decoded using the codes as indices into the LUTs. The output image is synthesized by filtering the frequency subbands as in Figure 5.7. The synthesis filter coefficients are plotted in Figures 6.19 and 6.20. The values are also listed in Appendix B, Table B.1.

The keyframe pairs are then used with the displacement vectors to interpolate each of the intermediate frames. Residual frames are decoded in the same manner as the keyframes, and added to the interpolated frames to give the luminance signal for these intermediate frames.

The chrominance is decoded directly from the chrominance codes and LUTs.

6.9 Postprocessing

The chrominance frames are spatially interpolated up to full resolution using the interpolation filters in Table B.2 (see also Figure 6.18). The output sequences are converted from YUV color space back to RGB using the equations 2.16 and 2.15.

6.10 Results

The bit rates required to the one second groups of frames are shown in Tables 6.6 to 6.10. Bit rates for the displacement vectors are shown in Table 6.10, and bit rates for the chrominance signal are displayed in Table 6.12. The resulting peak signal to RMS noise ratio (PSNR) are graphed in Figure 6.21. PSNR is defined here as:

$$PSNR(\hat{x}) = 20 \log \left(\frac{MAX(x) - MIN(x)}{\sqrt{\frac{1}{N} \sum_{i=0}^{N-1} (x_i - \hat{x}_i)^2}} \right) \tag{6.6}$$

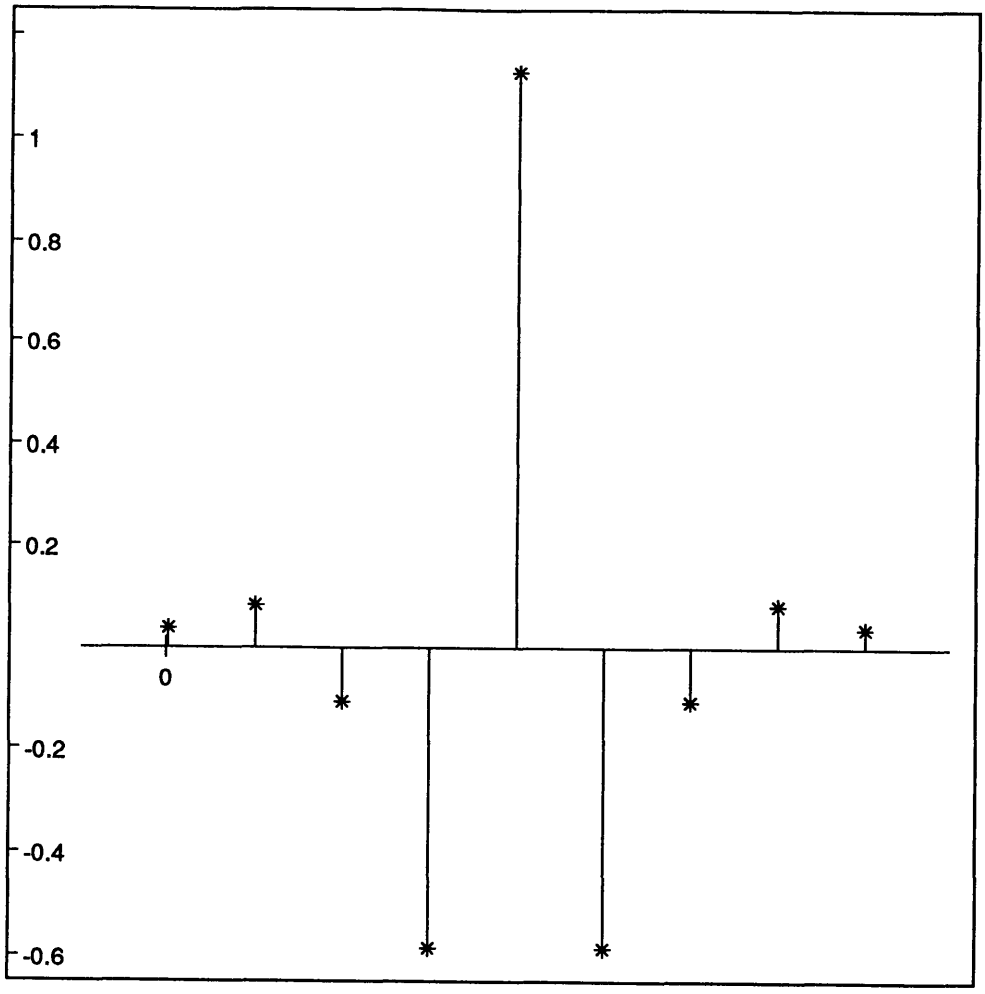

Figure 6.19: Lowpass filter used both horizontally and vertically in subband pyramid synthesis

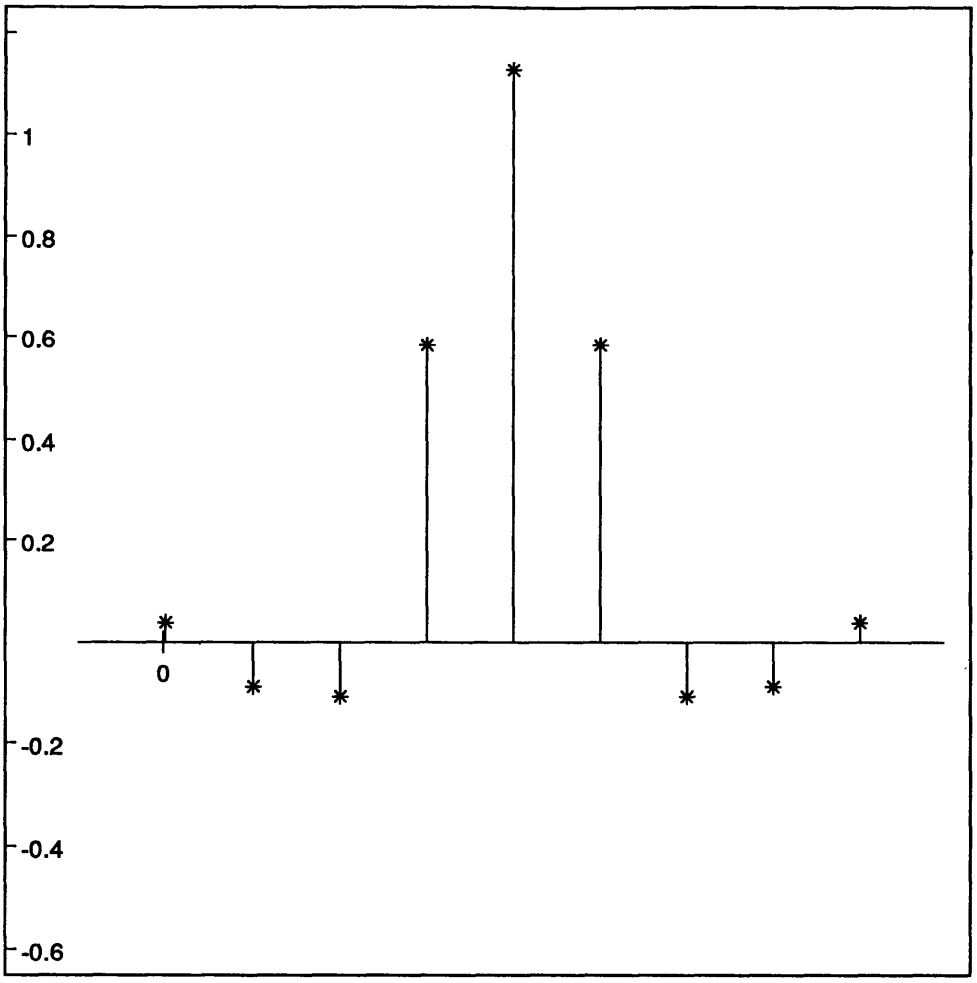

Figure 6.20: Highpass filter used both horizontally and vertically in subband pyramid synthesis

		Keyframes			Residuals		
Level	Band	Code (bits)	LUT (bits)	CBS	Code (bits)	LUT (bits)	CBS
Level 1	HH	96	16	1	240	16	1
	HL	50157	13042	362	19099	2064	71
	LH	60972	32099	606	29439	5534	140
Level 2	HH	5350	177	13	3418	127	12
	HL	141231	17220	897	44843	3168	194
	LH	120782	14061	741	50448	3376	215
Level 3	HH	82283	76	28	149008	45	7
	HL	100681	144	54	95186	43	3
	LH	89905	121	54	153102	52	7
	LL	126433	173	7	163046	1598	48
Totals		777890	77129		707829	14584	
Rate Totals (bits)		855,019			722,413		
Total Luminance (bits)		1,577,432					
Total Chrominance (bits)		141,170					
Vectors (bits)		57,409					
Grand Total		1,776,011					
Average Bits per Pel		0.275					

Table 6.6: Channel bit rate for *Alley* sequence frames $0 \rightarrow 20$

The fraction of bandwidth required for the keyframe luminance, residual luminance, chrominance, and displacement vectors over the entire simulation are shown in Table 6.13.

One point to note when interpreting the tables of bit rates is that since MCI requires two keyframes in order to interpolate the intermediate frames, there is a 'fencepost' problem. The test sequence was stored in files each of which has 24 frames of RGB images. I chose to code six keyframes at a time, which resulted in the first segment having five sets of intermediate frames (three intermediate frames in each set), resulting in a total of 21 frames. The last keyframe of each segment is used for interpolating the next segment, resulting in six sets of intermediate frames in each successive segment for a total of 24 frames. Subsequently, the first segment had 21 frames, and

		Keyframes			Residuals		
Level	Band	Code (bits)	LUT (bits)	CBS	Code (bits)	LUT (bits)	CBS
Level 1	HH	96	16	1	288	16	1
	HL	52366	14012	371	46946	5659	163
	LH	58762	30199	596	50999	9179	224
Level 2	HH	5170	149	11	8189	186	17
	HL	139925	17655	914	90606	7485	423
	LH	118117	12776	687	81705	4220	270
Level 3	HH	82542	89	29	175134	49	7
	HL	109351	145	53	138068	45	4
	LH	90024	112	45	158049	48	5
	LL	126934	185	78	252300	194	67
Totals		783287	75338		1002284	27081	
Rate Totals (bits)		858,625			1,029,365		
Total Luminance (bits)		1,887,990					
Total Chrominance (bits)		179,138					
Vectors (bits)		73,837					
Grand Total		2,140,965					
Average Bits per Pel		0.290					

Table 6.7: Channel bit rate for *Alley* sequence frames 21 → 44

		Keyframes			Residuals		
Level	Band	Code (bits)	LUT (bits)	CBS	Code (bits)	LUT (bits)	CBS
Level 1	HH	96	16	1	4304	65	4
	HL	55359	15974	417	106590	18760	478
	LH	57827	28418	561	95422	19049	441
Level 2	HH	5314	108	8	12247	332	26
	HL	145994	16913	874	169638	12804	676
	LH	118069	11438	629	139772	7999	468
Level 3	HH	84197	82	29	201356	57	7
	HL	112141	165	69	194293	53	6
	LH	91599	108	46	174695	42	5
	LL	132554	195	84	233776	253	98
Totals		803150	73417		1332093	59414	
Rate Totals (bits)		876,567			1,391,507		
Total Luminance (bits)		2,268,074					
Total Chrominance (bits)		232,959					
Vectors (bits)		143,919					
Grand Total		2,644,952					
Average Bits per Pel		0.359					

Table 6.8: Channel bit rate for *Alley* sequence, frames 45 → 68

		Keyframes			Residuals		
Level	Band	Code (bits)	LUT (bits)	CBS	Code (bits)	LUT (bits)	CBS
Level 1	HH	96	16	1	4896	49	3
	HL	39333	11910	317	137845	21509	512
	LH	11818	5512	131	116177	19017	452
Level 2	HH	1485	68	4	16667	657	51
	HL	127972	16154	840	227349	16640	865
	LH	78461	7305	416	185316	14046	746
Level 3	HH	79693	98	29	203153	51	8
	HL	111011	154	63	151358	52	5
	LH	88484	116	48	178331	71	8
	LL	133448	164	80	254400	246	65
Totals		671801	41497		1475492	72338	
Rate Totals (bits)		713,298			1,547,830		
Total Luminance (bits)		2,261,128					
Total Chrominance (bits)		235,569					
Vectors (bits)		230,066					
Grand Total		2,726,763					
Average Bits per Pel		0.370					

Table 6.9: Channel bit rate for *Alley* sequence frames 69 → 92

		Keyframes			Residuals		
Level	Band	Code (bits)	LUT (bits)	CBS	Code (bits)	LUT (bits)	CBS
Level 1	HH	96	16	1	7222	130	9
	HL	17486	3826	112	98902	23509	560
	LH	18309	4496	116	94377	29865	614
Level 2	HH	2988	110	8	37704	1956	138
	HL	101442	9414	526	294546	23706	1206
	LH	72734	4599	267	260528	17637	945
Level 3	HH	74478	83	2	149753	45	4
	HL	88664	127	51	56511	34	2
	LH	90095	146	52	143311	41	4
	LL	124508	169	85	194954	238	41
Totals		590800	22986		1337808	97161	
Rate Totals (bits)		613,786			1,434,969		
Total Luminance (bits)		2,048,755					
Total Chrominance (bits)		262,279					
Vectors (bits)		155,875					
Grand Total		2,466,909					
Average Bits per Pel		0.335					

Table 6.10: Channel bit rate for *Alley* sequence frames 93 → 116

Frames	Bits
0 → 20	57,409
21 → 44	73,837
45 → 68	143,919
69 → 92	230,066
93 → 116	155,875

Table 6.11: Displacement vector bit rates for *Alley* sequence

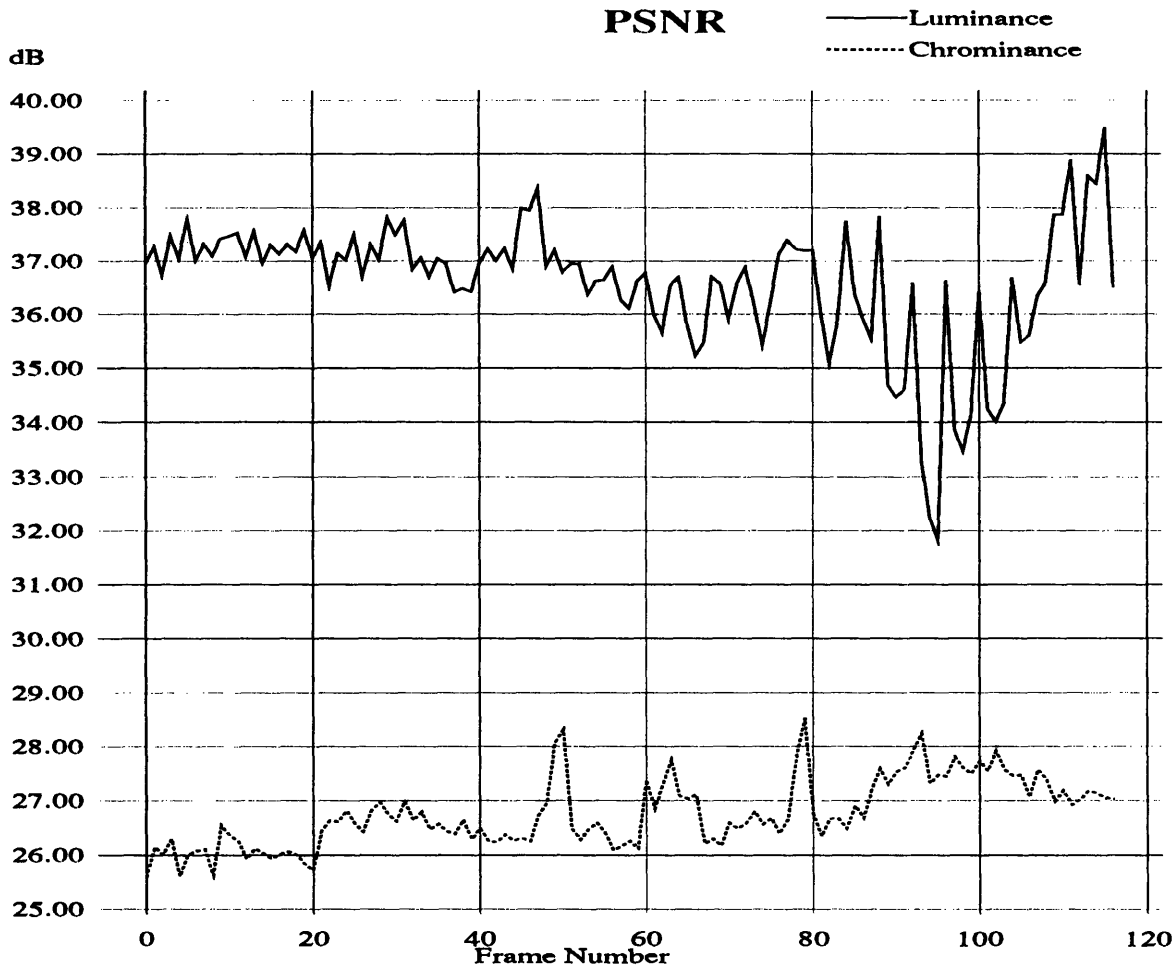

Figure 6.21: Luminance and Chrominance PSNR for *Alley sequence.*

Frames	Code (bits)	LUT (bits)	Total (bits)	CBS
$0 \rightarrow 20$	130582	10588	141,170	73
$21 \rightarrow 44$	169892	15390	179,138	106
$45 \rightarrow 68$	207016	33622	232,959	230
$69 \rightarrow 92$	200658	34911	235,569	249
$93 \rightarrow 116$	214807	47472	262,279	310

Table 6.12: Channel bit rate for *Alley* sequence Chrominance

Keyframe Luminance	33.33 %
Residual Luminance	52.11 %
Chrominance	8.94 %
Displacement Vectors	5.62 %

Table 6.13: Fraction bandwidth allocation for *Alley* simulation

each successive segment had 24 frames. For ease of comparison, I have included an average number of bits per pel at the bottom of each table. This average is derived from the total bit rate required for luminance and chrominance codes and LUTs, and vector fields.

An explanation of the bit rate tables is in order. The 'level' refers to the level of the octave pyramid decomposition, with level 1 being the first split. The two indices in the band column refer to the horizontal and vertical filtering. For example, band HL is highpass filtered horizontally and lowpass filtered vertically. 'CBS', code book size, is the number of unique symbols in the LUT for a given subband.

Rate totals are the total number of bits of all the subbands, codes, and LUTs for the keyframes and residuals, respectively. The keyframes and residuals are summed under 'Total Luminance'. This is the total number of bits required to transmit the luminance information for the frames listed in the caption below each table.

'Total chrominance' is the total number of bits required for the UV signal. This is also listed in Table 6.12. The row 'vectors' refers to the number of bits required to send the displacement vector fields for the listed frames. The above is summarized in the row 'Grand Total'. Following that is the average number of bits per pel to display at the 640x480 output resolution.

The signal to noise ratio roughly follows the amount of motion in the original sequence. The sharp drop in signal to noise ration of the luminance at frame 95 occurs when the two figures cross, one occluding the other. The displacement estimator performance drops at this point. It is interesting to note that the PSNR varies between keyframes as discussed in Chapter 3, especially where large amounts of motion are present. The keyframes of frames 80 through 104 could have been coded to a lower PSNR, shifting the the bit rate saved to the intermediate frames. This would result in a more consistent overall PSNR to improve the subjective quality.

The chrominance was coded without motion compensation. The PSNR for the chrominance signal is fairly consistent throughout the sequence. It should be noted that good quality was obtained with the chrominance coded at a significantly lower SNR than luminance.

Chapter 7

Conclusions

Motion compensated interpolation in conjunction with subband coding and vector quantization has been shown to be an effective method for bit rate reduction in a moving image sequence. The hierarchical multiframe matching estimation method can reduce the residual error over that of conventional block matching, when the displacement vectors are used for motion compensated interpolation.

Two important lessons learned in regards to motion compensated interpolated image coding systems are that the accuracy of the displacement estimation has a large impact on performance, much larger than the coded quality of the keyframes. For this reason, an accurate displacement estimation method should be used to obtain the motion vectors. The second is that the signal to noise ratios of the keyframes and residual frames should be closely matched to avoid flickering at rates which the human visual system is most sensitive.

The residual error signal in a motion compensated system tends to have energy distributions that are spatially localized. A subband pyramid decomposition of the residual signal will still have the energy spatially localized within each subband. However, by using a peak error criteria when building

the vector quantization look up tables, the areas of greatest error can be adequately reconstructed. The energy in the residual signal is distributed more in the middle spatial frequency subbands, in comparison to the distribution of energy of the keyframes which is primarily in the baseband. By coding using subbands, the bit rate allocation can be distributed in a manner appropriate to the energy distribution in the keyframes and residuals separately to achieve good results.

Appendix A

Computational Complexity and Hardware Implementation

The displacement estimation method outlined in this thesis using the minimum absolute difference criteria (MAD) uses simple calculations, yet there is a tremendous amount of these calculations required. Using the N-step method, there are $(N*8)+1$ block comparisons made. The subsampling methods used in each block resulted in a uniform 64 points per block regardless of the block size.

The spacing and search parameters used in the simulation were given in Table 6.1. These values required a worst case number of calculations as presented in Table A.1.

Level	Steps	Block Comparisons	Point Comparisons	
1	4	33	2112	60,825,600
2	2	17	1088	31,334,400
3	2	17	1088	31,334,400
			Total	123,494,400

Table A.1: Worst-case comparison operations for a multiframe search as used in the computer simulations on *Alley* sequence using 4800 vectors per four frame set.

Type	Steps	Block Comparisons	Point Comparisons	
Fast	3	25	1600	7,680,000
Full	na	225	14,400	69,120,000

Table A.2: Worst–case comparison operations for block search in a motion compensated predictive system using ± 7 pel search area, using 4800 vectors per four frame set .

Each comparison takes two operations (a subtract and an accumulate) so the total number of operations for a four frame set is 246,988,800. There are 6 four frame sets per second , so the requirement is for 1.48 billion operations per second (GOPS). In comparison, a simple block matching scheme using the same parameters would require one sixth of the operations or 246 million operations per second (MOPS).

The actual method used requires significantly less operations than the above calculations indicate. This is for two reasons:

- The search is performed only on areas classified as moving. This area ranged from 20 to 80 percent of the image.
- The simulation included a comparison with the minimum absolute difference after each line in the comparison block. If the current sum exceeded the minimum, the comparison to this test vector is ended.

The predictive system requires one vector set for each frame so there are 368.6 MOPS in the fast search case, and 3.318 GOPS in the full search case.

Appendix B

Filter Coefficients

The following table (B.1) gives the values of the quadrature mirror filters (QMFs) used in the analysis and synthesis of the recursive pyramid splitting for both the keyframes and residuals. Floating point values are shown, while the actual values used were converted to 16 bit fixed point.

The Gaussian decimation and interpolation filtered used in chrominance coding as given in Table B.2. As with the QMF filters, these are converted to 16 bit fixed point before applying them in the simulation.

Point	Highpass		Lowpass	
	Analysis	Synthesis	Analysis	Synthesis
0	1.995484e-02	3.990967e-02	1.995484e-02	3.990967e-02
1	-4.270508e-02	-8.541015e-02	4.270508e-02	8.541015e-02
2	-5.224239e-02	-1.044848e-01	-5.224239e-02	-1.044848e-01
3	2.927051e-01	5.854102e-01	-2.927051e-01	-5.854102e-01
4	5.645751e-01	1.129150e+00	5.645751e-01	1.129150e+00
5	2.927051e-01	5.854102e-01	-2.927051e-01	-5.854102e-01
6	-5.224239e-02	-1.044848e-01	-5.224239e-02	-1.044848e-01
7	-4.270508e-02	-8.541015e-02	4.270508e-02	8.541015e-02
8	1.995484e-02	3.990967e-02	1.995484e-02	3.990967e-02

Table B.1: QMF analysis and synthesis filters used in pyramid decomposition

Point	Decimate	Interpolate
0	1.722366e-09	6.889467e-09
1	4.401103e-08	1.827576e-07
2	8.259543e-07	3.303818e-06
3	1.138437e-05	4.392394e-05
4	1.152445e-04	4.609782e-04
5	8.568212e-04	3.557984e-03
6	4.678628e-03	1.871452e-02
7	1.876314e-02	7.239321e-02
8	5.526507e-02	2.210603e-01
9	1.195514e-01	4.964418e-01
10	1.899402e-01	7.597609e-01
11	2.216346e-01	8.551257e-01
12	1.899402e-01	7.597609e-01
13	1.195514e-01	4.964418e-01
14	5.526507e-02	2.210603e-01
15	1.876314e-02	7.239321e-02
16	4.678628e-03	1.871452e-02
17	8.568212e-04	3.557984e-03
18	1.152445e-04	4.609782e-04
19	1.138437e-05	4.392394e-05
20	8.259543e-07	3.303818e-06
21	4.401103e-08	1.827576e-07
22	1.722366e-09	6.889467e-09

Table B.2: Gaussian Decimation / Interpolation Filters used in chrominance coding

Appendix C

Acknowledgments

I would like to thank the many people who have supported me through this endeavor:

Andy Lippman for suggesting the topic of motion compensated interpolation; Pat Romano who developed much of the image processing utility programs in the Garden; Foof for all his good and bad music; Walter, Mike, Wad, Pascal and the rest of the Garden crew for their help with equipment and computer systems.

The greatest thanks go to my wife, Susan, who has endured both the sacrifices of being married to a poor graduate student and the long days and nights with two children while I finished my thesis and held down a full time job. Your patience is beyond anything I could have asked for.

Bibliography

[1] H. Abut, B. P. Tao, and J. L. Smith. Vector quantizer architectures for speech and image coding. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 18.9.1–18.9.4. IEEE, 1987.

[2] S. R. Adams. A real-time image compression system for use in workstation environments. *SID 89 Digest*, SID 89:334–337, 1989.

[3] K.-I. Aihara. Universal data compression scheme for enhancing multi-media communication. In *IEEE Globecom*, pages 1873–1877, 1986.

[4] H. Amor, D. Biere, and A. G. Tescher. Technical issues in low bit rate transform coding. In *Visual Communications and Image Processing '88*, volume SPIE 1001, pages 164–177, 1988.

[5] M. J. Badge. Interframe predictive coding of images using hybrid vector quantization. *IEEE Transactions on Communications*, COM-34:411–415, Apr. 1986.

[6] R. L. Baker and H. Shen. A finite-state vector quantizer for low-rate image sequence coding. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 18.10.1–18.10.4. IEEE, 1987.

[7] J. Barrilleaux, R. Hinkle, and S. Wells. Efficient vector quantization for color image encoding. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 18.5.1–18.5.4. IEEE, 1987.

[8] W. Bender. Adaptive color coding based on spatial/temporal features. In *Proceedings of SPSE Electronic Imaging Devices and Systems Symposium.* SPSE, Jan. 1988.

[9] M. Bierling. Displacement estimation by hierarchical blockmatching. In *Visual Communications and Image Processing '88*, volume SPIE 1001, pages 942–951. Proceedings SPIE, 1988.

[10] R. E. Blahut. *Principles and Practices of Information Theory.* Addison-Wesley Publishing Co., Reading, MA, 1988.

[11] M. E. Blain and T. R. Fischer. Optimum rate allocation in pyramid vector quantization transform coding of imagery. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 18.2.1–18.2.4. IEEE, 1987.

[12] B. Braun. Luminance adaptive chrominance coding. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 25.7.1–25.7.4. IEEE, 1987.

[13] W. Butera. Motion compensated interpolation for low bit rate image coding. Oct. 1988.

[14] W. J. Butera. Multiscale coding of images. Master's thesis, Massachusetts Institute of Technology, 1988.

[15] C. H. Chen. Laplacian pyramid image data compression. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 18.4.1–18.4.3. IEEE, 1987.

[16] T. N. Cornsweet. *Visual Perception.* Academic Press, Orlando, Florida, 1970.

[17] P. Douglas, G. Karlsson, and M. Vetterli. Statistical analysis of the output rate of a sub-band coder. In *Visual Communications and Image Processing '88*, volume SPIE 1001, pages 1011–1025. Proceedings SPIE, 1988.

[18] W. H. Equitz. Fast algorithm for vector quantization picture coding. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 725–728. IEEE, 1987.

[19] W. H. Equitz. A new vector quantization clustering algorithm. *IEEE Transactions on Acoustics Speech and Signal Processing*, 37:1568–1574, 1989.

[20] S. Ericsson. Fixed ans adaptive predictors for hybrid predictive / transform coding. *IEEE Transactions on Communications*, COM-33:1291–1302, Dec. 1985.

[21] D. Esteban and C. Galand. 32 KBPS CCITT compatible split band coding scheme. pages 320–325. IEEE, 1978.

[22] A. Furukawa, T. Koga, and K. Niwa. Coding effficiency analysis fo motion-compensated interframe dpcm with transform coding. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 22.4.1–22.5.5, 1985.

[23] R. G. Gallager. *Information Theory and Reliable Communications.* John Wiley & Sons, New York, 1968.

[24] R. G. Gallager. Variations on a theme by Huffman. *IEEE Transactions on Information Theory*, IT-24:668–674, Nov. 1978.

[25] M. Gilge. A high quality videophone coder using hierarchical motion estimation and structure coding of the prediction error. In *Visual Communications and Image Processing '88*, volume SPIE 1001, pages 864–874. Proceedings SPIE, 1988.

[26] B. Girod. Displacement estimation by multi-field-matching. Presented at Picture Coding Symposium, 1987.

[27] B. Girod. The efficiency of motion–compensating prediction for hybrid coding of video sequences. *IEEE Journal on Selected Areas in Communications*, SAC-5:1140–1154, Aug. 1987.

[28] B. Girod. Eye movements and coding of video sequences. In *Visual Communications and Image Processing '88*, volume SPIE 1001, pages 398–405. Proceedings SPIE, 1988.

[29] B. Girod. Motion–compensating prediction with fractional pel accuracy. Submitted to the IEEE Transactions on Communications, 1989.

[30] B. Girod and R. Thoma. Motion-compensated field interpolation from interlaced and non-interlaced grids. *SPIE Image Coding*, 594:186–193, 1985.

[31] R. M. Gray. Vector quantization. *IEEE ASSP Magazine*, pages 4–29, Apr. 1984.

[32] M. H. Groves. Calculation of displacement fields by means of the motion detection transform. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 23.6.1–23.6.4. IEEE, 1984.

[33] B. Hammer, A. V. Brandt, and M. Schielein. Hierarchical encoding of image sequences using multistage vector quantization. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 25.2.1–25.2.4. IEEE, 1987.

[34] H.-M. Hang and B. G. Haskell. Interpolative vector quantization of color images. *IEEE Transactions on Communications*, COM-36:465–470, Apr. 1988.

[35] H. Hashimoto, H. Watanbe, and Y. Suzuki. A 64 kb/s video coding system and its performance. In *Visual Communications and Image Processing '88*, volume SPIE 1001, pages 847–853. Proceedings SPIE, 1988.

[36] D. J. Heeger. Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, pages 279–302, 1987.

[37] D. A. Huffman. A method of construction of minimum redundancy codes. *Proceedings of the IRE*, 40:1098–1101, Oct. 1952.

[38] T. Ishiguro and K. Iinuma. Television bandwidth compression transmission by motion-compensated interframe coding. *IEEE communications magazine*, 20:24–30, Nov. 1982.

[39] J. R. Jain and A. K. Jain. Displacement measurement and its application to interframe image coding. *IEEE Transactions on Communications*, COM-29:1799–1804, Dec. 1981.

[40] P. Jakatdar and W. A. Pearlman. An optimal transform coding method applied to images. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 29.11.1–29.11.4, 1984.

[41] C. N. Judice. Entertainment video-on-demand at T1 transmission speeds (1.5 Mb/s). In *Visual Communications and Image Processing '88*, volume SPIE 1001, pages 396–397. Proceedings SPIE, 1988.

[42] G. Karlsson and M. Vetterli. Sub-band coding of video signals for packet-switched networks. In *Visual Communications and Image Processing II (1987)*, volume SPIE 845, pages 446–456, 1987.

[43] D. H. Kelly. Visual response to time dependant stimuli. i. amplitude sensitivity measurements. *Journal of the Optical Society of America*, 51(4):422–429, 1961.

[44] J. Kim and Rae-HongPark. Local motion-adaptive interpolation technique. In *Visual Communications and Image Processing '88*, volume SPIE 1001, pages 432–439. Proceedings SPIE, 1988.

[45] T. Koga. A 384 kbit/sec motion video codec with scene change detection. In *Proceedings of IEEE International Conference on Communications*, pages 366–370. IEEE, June 1986.

[46] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Isigur. Motion-compensated interframe coding for video conferencing. *NTC 81, Proceedings*, pages G5.3.1–G5.3.5, Dec. 1981.

[47] Y. Kosugi, K. Sakai, T. Hosokawa, and K. Matsuda. A realization of MC/DCT by video signal processors. In *Visual Communications and Image Processing '88*, volume SPIE 1001, pages 875–880. Proceedings SPIE, 1988.

[48] E. Krause. Motion estimation and interpolation in time-varying imagery. Master's thesis, Massachusetts Institute of Technology, 1984.

[49] E. Krause. *Motion Estimation for Frame-Rate Conversion.* PhD thesis, Massachusetts Institute of Technology, 1987.

[50] G. G. Langdon, Jr. An introduction to arithmetic coding. *IBM Journal of Research and Development*, 28:135–149, Mar. 1984.

[51] Y. Linde, A. Buzo, and R. M. Grey. An algorithm for vector quantizater design. *IEEE Transactions on Communications*, TCOMM 28:84–95, Jan. 1980.

[52] A. Lippman and W. Butera. Coding image sequences for interactive retrieval. *Communications of the ACM*, 32:852–860, 1989.

[53] J. Makhoul. A fast cosine transform in one and two dimensions. *IEEE Transactions on Acoustics Speech and Signal Processing*, ASSP-28:27–34, 1980.

[54] H. S. Malvar and D. H. Staelin. The LOT: Transform coding without blocking effects. *IEEE Transactions on Acoustics Speech and Signal Processing*, 37, No. 4:553–559, Apr. 1989.

[55] M. Margoudakis and J. D. Gibson. Experiments on video teleconferencing algorithms at 56 kilobits/sec. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 25.6.1–25.6.4. IEEE, 1987.

[56] V. J. Mathews, R. W. Waite, and T. D. Tran. Image compression using vector quantization of linear (one-step) prediction errors. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 18.2.1–18.3.4. IEEE, 1987.

[57] R. W. McColl and G. R. Martin. Quad-tree modeling of colour image regions. In *Visual Communications and Image Processing '88*, volume SPIE 1001, pages 231–238. Proceedings SPIE, 1988.

[58] G. W. Meeker. Signal processing advances in the Avelex video codec. In *IEEE Globecom*, pages 684–688, 1985.

[59] Miller and Wegman. Variations on a theme by Ziv and Lempel. Research Report RC 10630(No. 47798), IBM Laboratories, July 1984.

[60] J. L. Mitchell and W. B. Pennebaker. Optimal hardware and software arithmetic coding procedures for the Q-Coder. *IBM Journal of Research and Development*, 32:727–736, 1988.

[61] H. G. Musmann, P. Pirsch, and H.-J. Grallert. Advances in picture coding. *Proceedings of the IEEE*, Apr. 1985.

[62] H. Nagel. Detection and interpretation of changes in image sequences. In *Proceedings IEEE Colloquium on motion adaptive image processing*, pages 1.1 - 1.4, June 1986.

[63] A. H. Netravali and B. G. Haskell. *Digital Pictures, Representation and Compression.* Plenum Press, New York, 1988.

[64] Y. Ninomiya and Y. Ohtsuka. A motion compensated interframe coding scheme for NTSC color television signals. *IEEE Transactions on Communications*, COM-32:328–334, 1984.

[65] R. A. Nobaht and S. A. Rajala. An image coding technique using a human visual system model and image analysis critera. pages 1358–1361. IEEE, 1987.

[66] J. B. O'Neal, Jr. and T. R. Natarajan. Coding isotropic images. *IEEE Transactions on Information Theory*, IT-23:697–707, Nov. 1977.

[67] A. V. Oppenheim and R. W. Shafer. *Digital Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1975.

[68] D. E. Pearson and J. A. Robinson. Visual communication at very low data rates. *Proceedings of the IEEE*, 73:795–812, Apr. 1985.

[69] K. A. Prabhu and A. N. Netravali. Motion compensated composite color coding. *IEEE Transactions on Communications*, COM-31:216–223, Feb. 1983.

[70] A. Puri and H.-M. Hang. Adaptive schemes for motion-compensated coding. In *Visual Communications and Image Processing '88*, volume SPIE 1001, pages 925–935. Proceedings SPIE, 1988.

[71] A. Puri, H.-M. Hang, and D. L. Schilling. An efficient block-matching algorithm for motion-compensated coding. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 25.4.1–25.4.4. IEEE, 1987.

[72] Q. Qureshi and T. R. Fischer. A hardware pyramid vector quantizer. pages 1402–1405. IEEE, 1987.

[73] S. A. Rajala, M. R. Civanlar, and W. M. Lee. A seconf generation image coding technique using human visual system based segmantation. pages 1362–1365. IEEE, 1987.

[74] B. Ramamurthi and A. Gersho. Classified vector quantization of images. *IEEE Transactions on Communications*, COM-34:1105–1115, Nov. 1986.

[75] G. Ramponi and G. L. Sicuranza. 2- and 3-D nonlinear predictors. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 25.8.1–25.8.4. IEEE, 1987.

[76] J. Rissanen and G. G. Langdon. Arithmetic coding. *IBM Journal of Research and Development*, 23 No. 2:149–162, Mar. 1979.

[77] J. Rissanen and G. G. Langdon. Universal modeling and coding. *IEEE Transactions on Information Theory*, IT-27:12–23, Jan. 1981.

[78] J. Rissanen and K. M. Mouhiuddin. A multiplication free multialphabet arithmetic code. *IEEE Transactions on Communications*, COM 37:93–98, Feb. 1989.

[79] J. D. Robbins and A. N. Netravali. Spatial subsampling in motion compensated television coders. *Bell System Technical Journal*, 61:1895–1910, Oct. 1982.

[80] P. Romano, Jr. Vector quantization for spatiotemporal sub-band coding. Master's thesis, Massachusetts Institute of Technology, 1989.

[81] S. Sabri. Movement compensated interframe prediction for NTSC color TV signals. *IEEE Transactions on Communications*, COM-32:954–968, Aug. 1984.

[82] W. F. Schreiber. Psychophysics and the improvement of television image quality. *SMPTE Journal*, 93 Number 8:717–725, Aug. 1984.

[83] W. F. Schreiber. *Fundamentals of Electronic Imaging Systems.* Springer-Verlag, Berlin, 1986.

[84] G. R. Seabrook. Arithmetic coding – an alternative VLC strategy for video coding.

[85] C. E. Shannon. The mathematical theory of communication. *Bell System Technical Journal*, July and October 1948.

[86] H. Shen and R. L. Baker. An adaptive finite-state vector quantizer with conditional replenishment for low rate video compression.

[87] M. J. Smith and S. L. Eddins. Subband coding of images with octive band tree structure. pages 1382–1385. IEEE, 1987.

[88] R. F. Sproull. Refinements to nearest neighbor searching k-d trees. (unpublished), Feb. 1988.

[89] R. Srinivasan and K. R. Rao. Predictive coding based on efficient motion estimation. *ICC 1984 Proceedings*, pages 521–526, May 1984.

[90] J. B. Stampleman. Multiscale codebooks. Bachelor's Thesis, Massachusetts Institute of Technology, 1989.

[91] K. S. Thyagarajan and M. Viswanatahan. Low bit rate coding techniques. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 18.7.1–'18.7.4. IEEE, 1987.

[92] A. Tran and K.-M. Liu. An efficient pyramid image coding scheme. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, page 18.6.1. IEEE, 1987.

[93] R. Y. Tsai. Multiframe image point matching and 3-D surface reconstruction. *IEEE Transactions on pattern Analysis and Machine Intelligence*, PAMI-5:159–174, Mar. 1983.

[94] S. Tsuruta, K. Mitsuhashi, M. Mera, and K. Niwa. Intellegent communication terminal for integrating voice, data and video signals. In *IEEE Globecom*, pages 1509–1513, 1986.

[95] G. Tu, L. Van Eycken, and A. Oosterlinck. Hybrid image coding based on local-variant source models. In *Visual Communications and Image Processing '88*, volume SPIE 1001, pages 239–245. Proceedings SPIE, 1988.

[96] D. J. Vaisey and A. Gersho. Variable block-size image coding. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 25.1.1–25.1.4. IEEE, 1987.

[97] G. K. Wallace. Overview of the JPEG (ISO/CCITT) still image compression standard. In *Visual Communications and Image Processing '89*. SPIE, 1989.

[98] L. Wang and M. Goldberg. Progressive image transmission by multistage transform coefficient quantization. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 14.2.1–14.2.5. IEEE, 1986.

[99] T. A. Welch. A technique for high performance data compression. *IEEE Computer*, 17:8–19, June 1984.

[100] P. H. Westerink, J. Biemond, and D. E. Boekee. Quantization error analysis of sub-band filter banks. In *ISCAS-88*, 1988.

[101] P. H. Westerink, D. E. Boekee, J. Biemond, and J. W. Woods. Subband coding of images using vector quantization. *IEEE Transactions on Communications*, COM-36:713–719, June 1988.

[102] J. W. Woods and S. D. O'Neil. Sub-Band coding of images. In *ICASSP 86, Tokyo*, pages 1005–1007. IEEE, 1986.

[103] K.-M. Yang, L. Wu, H. Chong, and M.-T. Sun. VLSI implementation of motion compensation full-search block-matching algorithm. In *Visual Communications and Image Processing '88*, volume SPIE 1001, pages 892–899. Proceedings SPIE, 1988.

[104] C.-L. Yeh. Color image-sequence compression using adaptive binary-tree vector quantization with codebook replenishment. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 25.2.1–25.3.4. IEEE, 1987.

[105] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IT-23:337–343, May 1977.