INTERFRAME IMAGE CODING WITH THREE-DIMENSIONAL GRADIENT MOTION ESTIMATION

by

Choon Lee

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

APPROVED:

Morton Nadler, Chairman

A. A. (Louis) Beex

Richard W. Conners

John C. McKeeman

Clifford A. Shaffer

November, 1990

Blacksburg, Virginia

INTERFRAME IMAGE CODING WITH THREE-DIMENSIONAL GRADIENT MOTION ESTIMATION

by

Choon Lee

Committee Chairman: Morton Nadler

Electrical Engineering

(ABSTRACT)

New methods for coding image sequences in video conferencing systems are developed in this dissertation. A simple 3-D gradient operator is developed to estimate motion in an image sequence on a pixel-by-pixel basis. A stochastic 3-D gradient operator is also developed for a gradient calculation which is more robust to the noise effects of the image sequence. These gradient operators are used to estimate motion in the next frame using information from three previous frames. The concept of **tangent plane**, which is perpendicular to the gradient vector, is introduced to guide searching for the motion vector. The simplified 3-d gradient motion estimation (GME) technique is compared to Netravali's pixel-recursive method with scalar quantization. It was found that performance of the GME technique is very close to that of Netravali's technique with less computational complexity.

To adapt the motion estimation techniques to vector quantization, two new supplementary methods of motion estimation using the pixel motion vectors from the motion estimator were studied. The methods use either the pixel motion vectors directly on the moving block (pixel motion estimation) or calculate the block motion vector from the pixel motion vectors (block motion estimation). For both methods, the differences between the prediction block and the moving block are vector quantized.

The stochastic gradient motion estimation (SGME) technique made the motion estimation more robust by smoothing the valleys in the SNR made by the GME technique. The gradual degradation of images occurring in vector quantization of difference images could be suppressed with a smoothing filter for the prediction blocks. Combined with GME and block motion estimation (GME-B-PS), good quality images for video conferencing applications, about 37 dB in terms of SNR, could be obtained with a bit rate approximately 0.15 bits/pixel. To ensure the elimination of the gradual degradation, blocks with the code vectors not matching the prediction error block to a certain degree were refreshed with the original pixel blocks. The result shows an increase of overall bit rate of about 0.03 bits/pel and about the same SNR of 37 dB for the Miss USA sequence.

Acknowledgements

I would like to thank Dr. Morton Nadler, my advisor, for providing me time and support to complete this dissertation and teaching me an important lesson that age is not an obstacle to your desire. It was my honour and pleasure to work with Dr. Conners, the director of SDA laboratory and one of my committee members. I also would like to thank my other committee members, Dr. Beex, Dr. McKeeman, and Dr. Shaffer for their time and efforts. I appreciate Dr. Ha and Dr. Cho for their encouragement during my years in Ph.D. program. Special thanks to all the current and former members of SDA Lab for their supports and encouragements.

I would like to dedicate this dissertation to my parents in Korea without whom this dissertation couldn't exist. Lastly, I would like to thank my wife without whose endurance and support this work couldn't haven been completed.

Table of Contents

Table of Figures

Table of Tables

Chapter 1

Introduction

Coding of moving pictures is a central issue in the field of image compression. Many coding techniques have been developed for applications such as video conferencing, videophone, transmission of satellite images and HDTV. Existing techniques are well described in the survey papers by Netravali and Limb (1980) and Musmann *et al.* (1985). The limited bandwidth of communication channels demands that a very high compression ratio be used when sending image sequences. Therefore, existing coding techniques use complex algorithms to reduce the bit rate as much as possible.

For instance, consider transmitting an image sequence using the Common Intermediate Format (CIF) which was adopted by the CCITT Specialists Group for the video conferencing applications. Assume that the frame rate is 30 frames/second which is normal for television signals. Also assume that each sample is quantized to 8 bits/pixel. If only the luminance component of the image of size of 288x360 is sent, the uncompressed bit rate will be 24.3 Mbits/sec. Adding the chrominance components (two frames each of size 144x180) increases the uncompressed bit rate to 36.5 Mbits/sec. Suppose this signal is sent through an ISDN network. To use one B channel, which admits has a bandwidth of 64 kbits/sec, the required compression ratio will be approximately 570:1 and it is questionable that the quality of the compressed picture would be acceptable with current coding techniques. To preserve reasonable quality in coded pictures, several of these B channels will have to be used.

To code an image sequence with a low bit rate as mentioned above, it is inevitable that the coding algorithm will become complex. The aim of this research is to examine

several coding techniques which can compress an image sequence while using techniques with moderate complexity which are within the reach of current technology. Systems which can be implemented for the real-time compression of image sequences will be considered. Further, the image quality should be acceptable for practical use.

In addition to intraframe correlations which have been studied for use in still image coding, the interframe correlation needs to be investigated. Many intraframe techniques such as predictive coding and transform coding can be used in image sequence coding. These techniques must be modified to reduce the data size in three-dimensional space still further. Motion compensation is one of the most widely used interframe coding techniques. Among others, Netravali *et al.* (1979), Robbins *et al.* (1983), Bergman (1983) and Cafforio *et al.* (1983) have produced successful results with pixel-recursive methods but, because of the complex computations involved, there has not been much effort to incorporate them into a real-world image coder-decoder (codec). On the other hand, block motion estimation is an alternative method (Koga *et al.* (1981), Jain and Jain (1981), Srinivasan and Rao (1984)), and has been used in the image coding community for a long time. Although this method is computationally cheaper and has a higher compression ratio than pixel-recursive methods, the overall subjective quality of the coded results is very poor.

One of the main purposes of motion compensation or any other predictive coding method is to find the predicted value that is closest to the pixel to be coded. Here, we introduce a concept of a spatio-temporal (3-d) gradient to find the motion vector. Labit *et al.* (1982, 1983) is one of the first research groups to conduct a study on motion estimation with edges in the spatio-temporal domain. In the research presented here, the motion vector is estimated from the past few coded frames with a 3-d gradient. Using the 3-d gradient reduces the search space required to find the path of the moving pixel.

Searching for the corresponding pixel in the previous frame creates an enormous search space if large motion is assumed. To find the motion vector from the previous frame, the pixels along the ***tangent plane*** normal to the 3-d gradient vector are searched. This gradient vector is obtained with a simple spatio-temporal edge operator developed for this coding. Several different variations of this algorithm have been devised. It is shown that the developed algorithm works as well as the complex pixel-recursive motion estimation method by Netravali *et al.* (1979).

The 3-d gradient operator developed in this dissertation requires large amounts of memory to calculate the gradient components in 3-d space with a mask operations. Most of the motion compensation methods mentioned above do not use more than one frame for prediction. The reason seems to stem from the fact that memory was expensive in the past. At most one frame of memory was used in the above systems. But, the price of memory continues to plummet with advances in VLSI technology, and the justification for minimizing memory in a codec has faded rapidly.

In predictive coding systems, the prediction error is quantized to reduce the bandwidth of the signal which is sent to the receiver. The prediction error can be quantized in many ways such as scalar quantization and vector quantization. For an image sequence to be sent over data channels with narrow bandwidths, the compression ratio must be large. For this purpose, the vector quantization method is very attractive, however, the application of vector quantization to prediction error is not easy.

When the prediction error block is vector quantized, the prediction errors within the block are less correlated than the original image pixels. This is due to the pulsive components in the prediction error image, which appear as sharp pulses above the smooth surface. These components occur because of erroneous prediction along the moving

object boundary. If these components are not coded faithfully, the image will be severely degraded and show a lot of spot noise. Another problem with vector quantization is the accumulation of the prediction error. If the quantization is too coarse, the gradual degradation of the coded image sequence occurs. In the prediction error block, the pulselike components appears because of the inaccurate motion estimation. This effect increases the rate of image degradation.

To fix these problems, algorithms are developed which lessen the degradation. The prediction block, which is a product of the 3-d gradient operator and is subtracted from the original block, is filtered to give more reliable predictions. A coded block can be classified into either compensable and uncompensable depending upon whether the noise content is below a threshold or not. An algorithm is developed to refresh an uncompensable block using the original pixel values. Also the pixel motion vectors from the 3-d gradient motion estimator are converted to a block motion vector, which gives a more dependable prediction block for the vector quantizer.

The main focus of this research is on the design of a coding system with low bit rate and high image quality, evaluated both objectively and subjectively. These two requirements are often contradictory. An adequate tradeoff between the two requirements should be set to meet the purpose of the coding algorithm. Also for different types of systems, different quality requirements can be imposed. For instance, applications in HDTV images will require higher quality reproduction than those of videoconferencing or videophone images. It should also be clear that for a given bandwidth, the quality of the images coded by this technique is superior or comparable to other techniques. The mean-square-error (mse) method often used in judging the quality of coded pictures will be used as an objective quality measure. Although this method has a poor correlation with the subjective quality, no other method has been proven to be better.

Hardware realization is a main concern in image coding. To compress an image to an acceptable SNR and yet still transmit through an accessible channel, will require tremendous complexity. The complexity is due to the extensive decorrelation process required for coding techniques in both the spatial and temporal domain. The overall complexity of the system should be moderate. The word moderate was used here taking into account of the fact that the system can be fairly complex when a high compression ratio is desired, but the structure of the system should be adequate for the hardware design with current technology, possibly in VLSI. The proposed method is suitable for integration in VLSI since many blocks in the system can be duplicated. Also an algorithm for reducing the computation time required for gradient calculation has been developed. This method decreases the computation time by reducing duplicate operations and storing the gradient components in the memory.

The speed requirement for the hardware for motion estimation can be too burdensome if the limitation of current technology is not taken into account. In most coding systems the motion estimation has to be done within the allowed time between the quantizations of two blocks. As will be explained later, the parallel operation of the 3-d gradient operator and the vector quantization alleviates the problem of delay involved in the calculation of motion vectors between blocks.

Chapter 2 presents the background for the foregoing discussion and literature surveys. The main focus is on 3-d edge detection, predictive coding, motion estimation, and vector quantization. The papers most related to the ongoing research will be discussed.

Chapter 3 describes the 3-d edge detection methods developed for motion estimation. In the first part, the concept of a 3-d gradient vector is explained. In the second

part, a simple 3-d edge detection method will be demonstrated with a time-saving algorithm. Lastly, a stochastic 3-d edge operator will be developed for use in noisy image sequences.

In Chapter 4, motion estimation techniques using the 3-d edge detection will be presented. The concept of a tangent plane is introduced along with the methods to search through this plane.

Chapter 5 presents the vector quantization (VQ) methods to code the prediction error images from the 3-d motion estimator. Simple VQ methods combined with the 3-d motion estimator have been developed. Problems with these methods, such as gradual degradation of a coded image sequence, are discussed and methods to solve the problems will be demonstrated.

Chapter 6 consists of details of the simulation procedure and the results. In the first part, the result from the combination of the proposed 3-d motion estimator and scalar quantization are presented. The performance of the system is evaluated against that of the existing pixel-recursive motion estimation method. In the second part, the results from the combination with VQ are shown. The results of newly developed algorithms which can eliminate the problem of gradual degradation are shown both subjectively and numerically.

Chapter 7 discusses what has been achieved and what can be concluded.

Chapter 2

Background and Literature Review

2.1 Predictive Coding

Predictive image coding has been widely used in image coding as well as in speech coding. It is widely used because of the simplicity of its hardware realization. This method is cost effective if the required bit rate is not too low (below 1.0 bits/pixel). A predictor in the negative feedback loop predicts the sample to be encoded from the previously coded samples. The error between the predicted value and the sample to be coded is quantized and sent to the decoder. The predictive coder has three components: predictor, quantizer and code assigners, shown in Fig. 1. The predictive coder can be further classified into two different systems, depending upon the number of levels in the quantizer. Namely, if the number of levels in the quantizer is 1, it is called **delta modulation** (DM) and if the number is greater than two, it is called **differential pulse code modulation** (DPCM). Because the motion estimator to be discussed in the next section is based on DPCM, this discussion will focus on DPCM.

The predictor is linear or nonlinear, depending upon whether the predicted value is a linear or nonlinear function of previously coded values. First, consider the linear predictor, which can be described by the following equation:

$$\hat{x}(n) = \sum_{i=1}^{N} h_i \hat{x}(n-i) \tag{2.1}$$

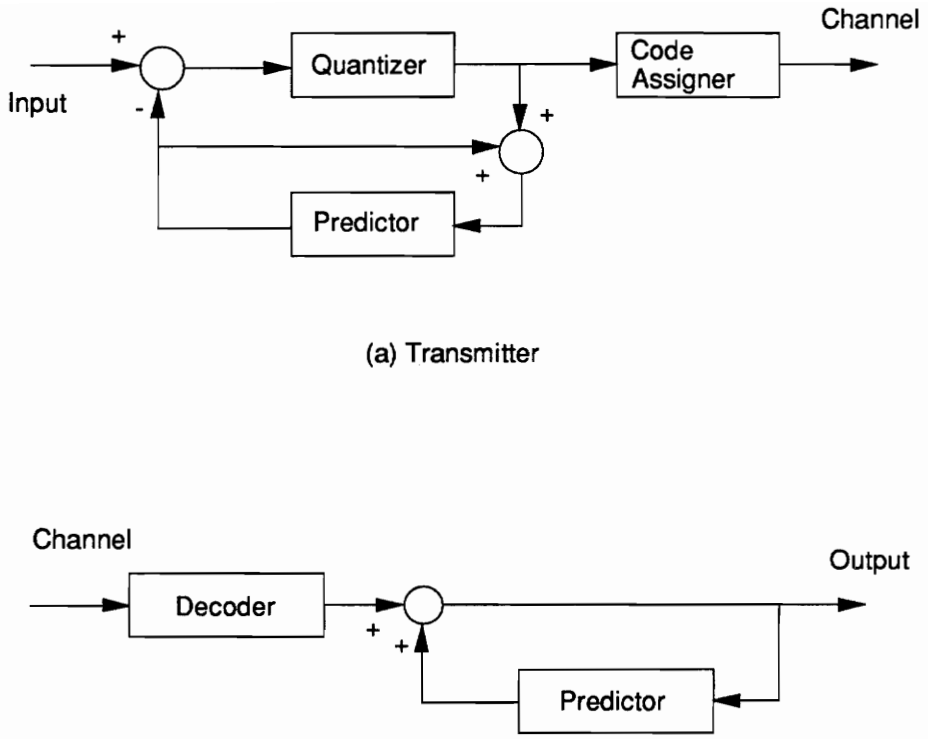


Figure 1. Block diagram of DPCM codec

where $x(i)$ are the sample value of the video signal and the $\hat{x}(n)$ is the predicted value from N previous samples. The h_i's are also called the predictor coefficients. However, in a real DPCM coder, the sample x is predicted using the encoded representations of the past samples, not by using the original uncoded sample values. The predictor becomes an all-pole predictor because the present output depends on the past outputs, not the inputs. In the video signal coding applications, the signal into the predictor loop in Fig. 1 is digital. The input to the transmitter is A/D converted and a digital filter is used in the predictor.

The optimum values of h_i, the predictor coefficients, can be solved by minimization of the mean-square prediction error (Jain, 1989). The equation to solve is given by

$$R_{xx}(k) = \sum_{j=1}^{N} h_{j,opt} R_{xx}(k-j) \qquad k = 1, 2, \ldots N \tag{2.2}$$

where R_{xx} is the autocorrelation function of x. This equation can be expressed in matrix form and it is called the Yule-Walker prediction equations or Wiener-Hopf equations. There are many methods to solve this equation such as Levinson or Durbin's algorithms (Jain, 1989).

Habibi (1971) has studied on the correlation between the performance of the linear predictor and the number of coefficients. His results show that the decrease in mean-squre-error (mse) is not significant if more than three picture elements are used in the prediction within a frame. The SNR gain in going from N=1 to N=3 is approximately 4 dB. Unless the coefficients are closely matched to the image statistics by finding the optimum coefficients for each image, the decrease in mse found by increasing the number

of elements from one to three is insignificant.

In intraframe DPCM image coding, two-dimensional predictors are normally used. In the mse sense, using a two-dimensional predictor has slight advantage over an one-dimensional predictor (O'Neal and Natarajan, 1977). However, a tremendous improvement occurs in subjective quality due to the rendition of vertical edges (Conner *et al.*, 1971).

Some predictors use elements from the previous frame in addition to the elements in the present frame. Also the elements from previous fields in interlaced video signals, such as the television signal, can be used. When there is low detail and little motion, interframe prediction works better, while with greater motion, the intrafield predictor works better (Limb *et al.*, 1974). This is due to the fact that when the motion increases, the correlation between the present and the previous frame pixels decreases. Also, the low pass filtering effect (integration) of the camera increases the spatial correlation in the direction of motion (Conner and Limb, 1974).

Next, the nonlinear prediction will be discussed. The linear prediction methods assume that the video signal is stationary over the whole picture. In fact, video signals are highly nonstationary. In areas where a lot of detail is present, the reproduction of signals will be very poor when the parameters for the global statistics are used. Instead, the local statistics on the neighborhood pixels can be applied to find a better prediction. In one of the nonlinear prediction methods, called the adaptive prediction, a predictor can be selected from a set of several predictors according to the directional correlations. This is known as contour prediction. Normally, the contour direction is estimated from previous picture elements and no information about the contours is sent to the receiver.

Graham (Netravali and Limb, 1980) used a very simple adaptive scheme using only vertical and horizontal predictions. The switching is done by comparison of the surrounding line and element differences. This method tries to estimate the contour with too few pixels and often causes inaccurate prediction.

Zschunke (1981) uses more neighborhood pixels than the previous method. First, contour pixels are separated from the non-contour pixels. If the pixel being coded is a non-contour pixel, the prediction direction of the previous pixel is used. The contour direction is estimated by comparison of the differences between the previous pixel and its three neighboring pixels in the previous line.

Dukhovich and O'Neal (1977, 1978) developed a three-dimensional nonlinear prediction technique used in coding television signals. This technique doesn't use motion estimation. The prediction is a weighted sum of three estimates called representatives. The three representatives are derived from the pixel values in three mutually perpendicular planes. The authors claim that error in the input of such a predictor decays fairly rapidly and does not result in catastrophic failure of the predictor.

2.2 Motion Estimation

In the previous section, some of the representative predictive coding techniques were introduced. The concept of predictive coding can be extended to interframe predictive coding. One of the evident differences with intraframe (still) image coding is that we have to deal with the correlations between the frames (interframe) in addition to the correlations within frames (intraframe). Interframe coding techniques minimize the redundancy between the successive frames of an image sequence instead of intraframe redundancy. There are techniques such as frame repetition, conditional replenishment,

adaptive predictive coding, motion compensation, interframe hybrid coding, etc. (Netravali and Limb (1980), Musmann *et al.* (1985)). Among the various methods, the most active area of research is that of motion compensation.

Even though the method of Dukhovich *et al.* (1977, 1978) uses the previous frame pixels for prediction, adaptive prediction cannot be successful until it incorporates an estimate of motion. Motion estimation tries to estimate the translations of objects in between subsequent images. Several researchers such as Huang (1981), Roach and Aggarwal (1980), Tsai and Huang (1981), Schalkoff and McVey (1982) and Nagel (1983) have studied the estimation of parameters of motion models from a sequence of moving pictures. But such elaborate models have not been adopted in image coding because the complexity of the algorithms makes it almost impossible for the motion estimator to be implemented in real-time image coding applications with the current technology at hand.

One of the pioneering works on motion estimation algorithms was done by Limb and Murphy (1975). Their algorithm calculates one displacement estimate for the entire moving area. They assume that the whole TV picture has only one moving object and only horizontal motion is present. The displacement is estimated by forming a parallelogram with the summations of magnitude of the frame difference signal and that of the element difference signal. The sum of the element differences becomes the height of the parallelogram and that of the frame differences becomes the area of parallelogram. A problem arises when the integral over the frame differences is not a linear function of the displacement. The parallelogram relationship does not hold in real television signals. This algorithm further has the disadvantage of calculating the displacement in only one direction of motion, and is unable to estimate the direction itself.

Cafforio and Rocca (1976) have developed a more general method which not only includes the previous algorithm but is also easier to implement. They apply a Taylor series expansions to estimate the frame difference assuming that the movement is purely translational. Optimal estimates for the x- and y-directional translations are obtained by linear regression. This algorithm performs well down to 2.5 pixels/frame (Cafforio and Rocca, 1979).

A new approach, recursive motion estimation, was disclosed by Netravali and Robbins (1979). Let DFD be the displaced frame difference defined as

$$DFD(x,y,\hat{D}_i) = s_k(x,y) - s_{k-1}(x-\hat{dx}_i, y-\hat{dy}_i) \tag{2.3}$$

where s_k and s_{k-1} are the image samples in the present frame and the previous frame, respectively. $\hat{D}_i$ is the estimated displacement while $\hat{dx}_i$ and $\hat{dy}_i$ are its x- and y-components. To find the motion estimate, the squared motion frame difference DFD^2 is minimized. The algorithm continuously improves $\hat{D}_i$ by updating. It iterates from one pixel to the next one either along a scanning direction, from line to line, or from frame to frame. A gradient method is used to minimize the squared displaced frame difference recursively. The following equation describes the optimization procedure.

$$\hat{D}_{i+1} = \hat{D}_i - \frac{1}{2}\varepsilon\nabla_{\hat{D}_i}[DFD(x,y,\hat{D}_i)]^2 \tag{2.4}$$

$\nabla_{\hat{D}_i}$ is the gradient operator with respect to $\hat{D}_i$ and ε is a positive constant. The following equation can be derived from the equation above.

$$\hat{D}_{i+1} = \hat{D}_i - \varepsilon DFD(x,y,\hat{D}_i)\cdot\nabla s_{k-1}(x-\hat{dx}_i, y-\hat{dy}_i) \tag{2.5}$$

There is another approach to motion estimation that is worth mentioning. Instead of estimating motion pixel by pixel as in the previous method, block matching can be employed to find the motion in blocks. A cross correlation function is used to find the peak in a search area. If it is assumed that the block size is $M \times N$ and maximum displacement in x- and y-directions are m and n, respectively, the search area is $(M+2m)\times(N+2n)$. But the correlations are calculated $(2m+1)^2$ times at most when m=n.

There are several methods of reducing the complexity of block matching. The first one is simplifying the matching function. Besides the normalized cross-correlation function which is too time-consuming in real-time systems, the mean square error method by Jain and Jain (1981) and the mean of the absolute frame difference by Koga *et al.* (1981) can be used. The search time can be reduced significantly if a better search method is devised. Jain and Jain (1981) introduced the 2-d logarithmic search procedure. In their method, five search points are examined in each step, reducing the volume of the search greatly. At the same time, Koga *et al* (1981, 1983) developed a three-step search procedure which is very similar to the previous method. Eight search points are scanned except the center point. The spaces between the search points are reduced as the steps advance. The strong point of this algorithm is that it guarantees that the displacement estimate is found in three steps. Srinivasan and Rao (1984) delivered another search algorithm called conjugate direction search. Instead of searching in two dimensions, this method locates the minimum error points one dimension at a time. After the points in the x-direction are searched for the minimum distortion point, the points in the y-direction are investigated.

2.3 3-d Edge Detection

The luminance edges are local discontinuities in image luminance or amplitude level. They are the important primitive features which provide an indication of physical extent of objects within the image. The two-dimensional edge is characterized by its height, the slope angle, the slope midpoint and the orientation with respect to the x-axis. Edge detection consists of gray scale edge enhancement by linear or nonlinear processing. The resultant image with its amplified luminance change is thresholded to determine the location of edges.

Edge detection can be divided into two main approaches: difference operators and edge matching/fitting. Difference operators consist of methods such as the gradient and Laplacian. The edge matching and fitting methods include mask matching, step fitting, and ramp/surface fitting. Of these, gradient operators present particular importance for the present research. The gradient operator gives the maximum rate of change of gray values. The magnitude of the gradient is used as the measure of edge strength and its angle gives the direction of the edge. In a spatial coordinate system, the gradient of the image sample f can be written as

$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} = \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta \tag{2.6}$$

where r is an arbitrary coordinate along which the gradient is measured. Equation (2.6) can be maximum when $(\partial/\partial\theta)(\partial f/\partial r) = 0$. At the maximum, the magnitude and the phase of $\partial f/\partial r$ are given by

$$\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad \text{and} \quad \tan^{-1}\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right), \tag{2.7}$$

respectively (Jain, 1989). The Roberts operator (Roberts, 1965), the Prewitt operator (Prewitt, 1970), the Sobel operator (Jain, 1989) and the isotropic operator (Jain, 1989) are some examples of gradient operators.

The concept of a gradient operator can be expanded to three-dimensional (3-d) space. An edge in 3-d space no longer represents a line. Instead, the edge forms a surface along the boundary of a 3-d object. The details of 3-d edge detection will be explained in the next section. There has not been much work on 3-dimensional edge detection compared to 2-d edge detection. The reason seems to be that there were few areas in which this method could be applied. The first applications of 3-edge detection were in medical image processing. The main purpose of 3-d edge detection in this application was to find a 3-d object in a sequence of X-ray or computed tomography pictures. The next application was in time-varying image sequence analysis. Estimation of object motion in a sequence of pictures was the main objective in edge detection.

Liu (1977) is one of the first authors to report use of the 3-d edge detection. A 3-d gradient operator was employed to reconstruct images of organs from a set of X-ray images. He introduced the term ***voxel***, which designates a volume element. The symmetric gradient operator of Roberts (Jain, 1989) was modified to find the three gradient components in three-dimensional space. Note that the point for which the gradient is obtained is located at the corners of pixels, i.e. between pixels.

Zucker and Hummel (1981) approached 3-d edge detection with mathematical derivations rather than with ad hoc methods. Their operator was mainly aimed at three-dimensional imagery of internal structures, such as in computed tomography. They approximated an image cube with a sphere as in Heuckel's (1971, 1973) work. The plane which separates the dark hemisphere from the light hemisphere is found through the

minimization of the norm between the image defined on the unit solid sphere and the approximation of the image divided by the plane. The coefficients for the plane are obtained by representing the image by a set of orthogonal basis functions. The coefficients of 3x3x3 and 5x5x5 cubes were obtained. The planes of a sample image which passes through the voxels (center of the cube) were displayed to reconstruct the 3-d object. The coefficients of the 3-d cube masks resembled those of the 2-d masks by Sobel (Jain, 1989). The Sobel operator is well-known for its good performance in detecting edges in 2-d images. Their resemblance justified that the coefficients of the 3-d edge operator can also be obtained by generalization of the Sobel operator.

In the research conducted by Horn and Schunck (1981), the partial derivatives of image brightness with respect to the x, y and t-axes were used to estimate optical flow, which is another term for motion. Each of the derivatives was calculated by averaging the differences along four parallel edges in a 2x2x2 cube.

Labit and Benveniste (1982, 1983) first introduced the concepts of ***spatio-temporal filtering*** and ***spatio-temporal edge***. They have defined the surface constructed with moving edges in the spatio-temporal domain as a ***spatio-temporal edge***. The edge following is performed by a finite state automaton. The edge motion is estimated with an assumption that no edge slides on itself so that only the motion normal to the direction of edge is considered. This algorithm is used to reinitialize the motion estimation along boundaries of objects where the pixel-recursive motion estimation methods fail to give accurate displacements.

Topa and Schalkoff (1989) designed a set of three-dimensional templates to be used to calculate the three-dimensional gradients at about the same time as Lee and Nadler (1989) developed a 3-d gradient calculation method. The calculation of the mask was

based on the method of Frei and Chen (1977). Topa *et al.* found two coefficients such that the isotropic condition for the gradient magnitude is satisfied. Dependent on whether a diagonal edge is moving or not, the gradient magnitudes for both cases are found. Then two equations with two unknowns are formed by equating the gradient magnitudes and that of the x-direction gradient, assuming that an ideal step edge is passed through the masks. The gradient estimates are quantized into a set of 24 different groups (8 spatial orientations and 3 edge motions). To eliminate the trigonometric calculations in gradient quantization, each mask is tuned for one of eight possible spatial orientations and three temporal motions.

2.4 Vector Quantization

Vector quantization (VQ) is a relatively new coding method in image coding, but it has been used extensively in speech coding (Gray, 1984). VQ is especially useful in areas where a high compression ratio is desired. This is due to the fact that VQ techniques quantize a signal in the form of vectors. The conventional coding methods such as predictive coding employ scalar quantization. Transform coding operates on vectors and achieves a high compression rate but quantization is still accomplished in terms of scalars. Because of this characteristic, conventional methods are not optimal in the sense that they can't discard the intersample correlations. Shannon's rate-distortion theory (Shannon, 1948) states that coding of vectors rather than scalar values gives better performance.

One of the unique characteristics of VQ is that high compression ratios can be achieved with small block sizes contrary to the case of transform coding, where large block size is needed. This fact allows the coder to achieve higher subjective quality because the details can be preserved with smaller block sizes. Another advantage of VQ

is that the decoder is very simple to implement. This is particularly attractive in applications such as videotext, archiving, broadcasting, and possibly in HDTV transmissions, where a simple decoder structure is essential.

The idea of VQ will now be briefly introduced. Suppose the input to the encoder is x, a vector of dimension k which is a block size. In image coding applications, the vector is usually a contiguous, nonoverlapping, square block from an image which is in the order of left-to-right and top-to-bottom. Figure 2 shows the basic block diagram of a VQ compression system. The encoder selects the best matching code vector to the input vector from a previously generated codebook. The criterion for choosing the best code vector depends on the distortion measure $d(\mathbf{x}_n, \mathbf{y}_i)$. Here, $\mathbf{x}_n$ is the nth input to the encoder and $\mathbf{y}_i$ is the code vector. Normally, the mean squared error (mse) method is used. Note that the code vectors selected by this method don't necessarily deliver subjectively good images.

The channel symbols, u_i, are produced by the encoder for transmission through the channel. This is the index of the code vector selected in the encoder with a distortion measurement. The decoder receives this channel symbol and looks for the ith code vector in the codebook. The output from the decoder is $\hat{\mathbf{x}}_n$ which is $\mathbf{y}_i$. If there are N vectors in the codebook, $\log_2 N$ bits are needed to code one index. The final bit rate becomes $(1/k)\log_2 N$ bits/pixel.

Codebook design plays a major role in the performance of the VQ system. For a codebook to be optimal, the average distortion between all the code vectors and input vectors should be minimum among all possible codebooks. However, this is impossible in a practical sense. Instead, a well-known suboptimal codebook generation method called the LBG (Linde, Buzo and Gray) algorithm (Linde *et al.*, 1980) is often used. This

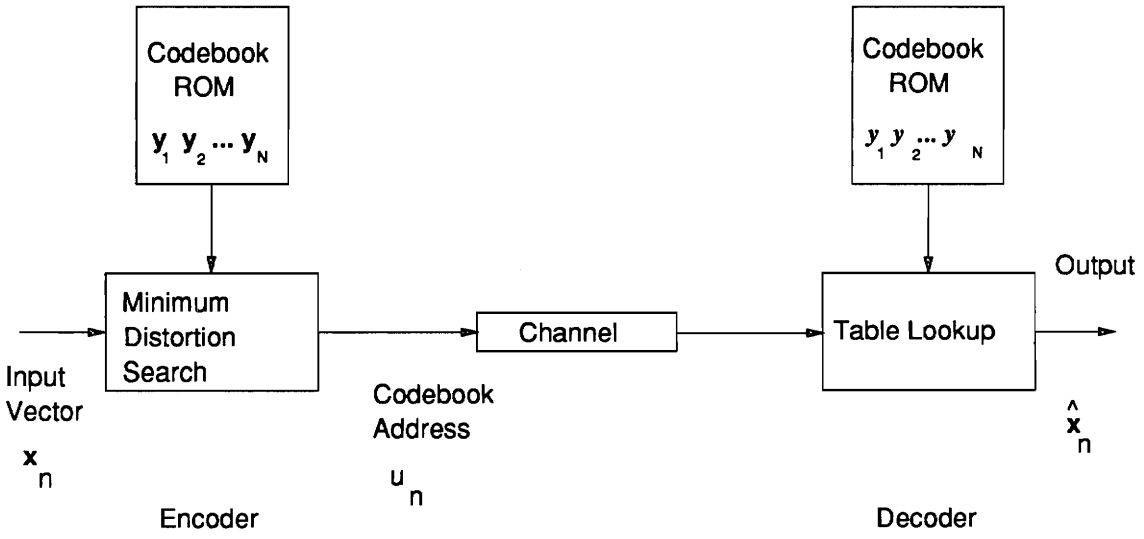


Figure 2. Block diagram of VQ codec

produces a locally optimum codebook using a clustering algorithm which is widely used in pattern recognition. The codebook thus created is not globally optimum, which means that one can find numerous codebooks which give the same or even better performance.

For the LBG algorithm to generate a codebook, an initial codebook must be supplied. This is a starting point for the algorithm and it affects the final performance of the VQ. The initial codebook can be generated in various ways. One of the methods is to use a training vector sequence to generate the initial codebook (MacQueen, 1967). Many variations are possible, such as selecting widely spaced vectors from the training sequence if the sequence is highly correlated. An alternative method is to use a product code [Gray, 1984] which is a collection of scalar codes, such as a uniform quantizer. For each element of a vector, scalar quantization is applied. Another method is the splitting technique (Linde *et al.*, 1980). This method starts from a zero rate code and builds larger rate codes by successively splitting the centroid of each partition into two components.

Vector quantization of prediction errors is a relatively new application. The source of prediction errors can be from intraframe predictors or interframe predictors such as in motion compensated prediction. The focus of this survey will be on coding interframe prediction errors. Murakami *et al.* (1984) has developed a vector quantizing coding technique for interframe prediction error in moving areas. The mean and the standard deviation of the moving block are first obtained and separated from the block. The normalized block is then vector quantized. A transmission data buffer constantly monitors the bit rate and tries to match the channel capacity using a feedback loop to the motion detector. The authors claim that good images for teleconferencing are transmitted at a bit rate of 0.2 bit/pixel but it is questionable if the images are indeed acceptable with this simple technique.

Bage (1986) designed a different approach for vector quantizing the interframe prediction error signal. A binary tree is used to organize the codebook to support searching for the code vector. In the last levels of the binary tree the training set is not represented as closely by the code vectors as in the first few levels. To reduce this effect, the quantization of the prediction error is accomplished by cascading two different VQ stages. The first stage is a VQ constructed from the LBG algorithm where its codebook is defined by the first five levels of a binary tree. In the second stage, the difference between the input vector and the output from the first stage is lattice quantized. One problem with this quantizer is that the overhead due to the transmission of codewords from the second stage can be quite large. The distortion versus entropy graph presented in their work shows that improvement can be obtained by coding sequences far from the training sequence. However, this improvement is acquired at the expense of a large increase in the bit rate.

Furner *et al.* (1986) suggested combining VQ with motion compensation. For motion compensation, a simple block matching technique is used. Two systems using motion compensation are introduced. The first one vector quantizes the blocks for which the distortion after motion estimation is above a threshold and sends the displacement coordinates for the rest of the blocks. The second system vector quantizes the motion estimated error signals. The difference between the original signal and the estimated signal is vector quantized if the distortion of the block is above a threshold. The problem with the differential approach is that the mse diverges as more frames are processed. To avoid this, the authors suggest two different refreshment methods. One is to refresh a fixed number of blocks in each frame. The other is to refresh with fixed threshold for block distortion.

Recently, Nasrabadi *et al.* (1989a, 1989b) have developed an interframe hierarchical vector quantizer which uses the regular decomposition quadtree method. VQ is done with variable block size depending on whether the block represents moving boundaries or not. Small blocks of size 2x2 or 4x4 are encoded by a vector quantizer and larger blocks are substituted by the scalar quantized sample mean. The pulsive component in the difference signal due to the incorrect reconstruction of the object boundaries can be represented accurately by small blocks of size 2x2.

The error image is first subdivided into 32x32 blocks. Then by top-down quadtree segmentation, the block is further decomposed into 16x16, 8x8, 4x4, and 2x2 blocks, depending on the energy information. The authors claim that images suitable for teleconferencing applications can be generated at bit rates below 0.3 bits/pixel. The reduction in the bit rate mainly comes from the quadtree representation in block segmentation. In particular, the ability to represent a large inactive area with an average value contributes to the bit rate reduction. However, the hardware requirement and computation time are not considered. Even though the largest block size is 32x32, the calculation of variances in block division involves a fair amount of computation.

Chapter 3

3-d Edge Detection

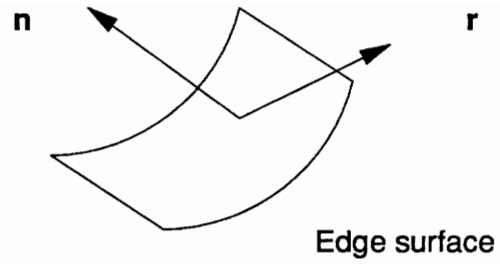
3.1 Concept of a 3-d Edge

Edge detection has been widely used in image analysis. Edge points are the feature points of an image where the gray level changes abruptly. They provide an indication of the physical extent of objects within the image. The derivative of a continuous image indicates the local maximum perpendicular to the direction of the edge. The concept of an edge in the 2-d domain can be extended to the 3-d where the time axis (t-axis) is added. As can be seen in Fig. 3(a), the 3-d edge is not a line. Rather, it is a surface consisting of many edge points.

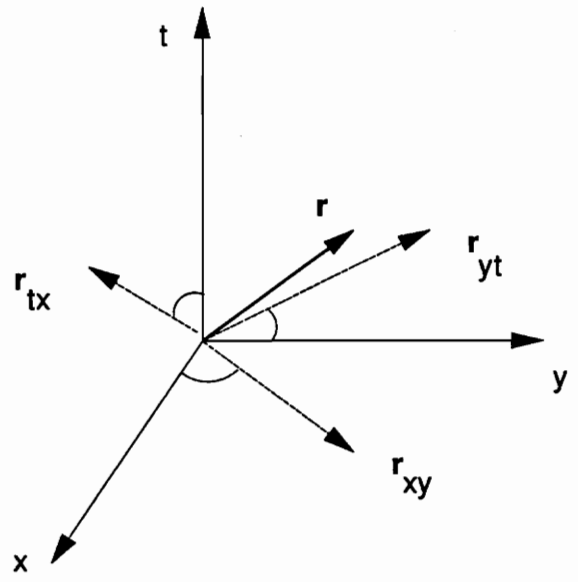
The concept of a 3-d edge can be explained with the gradient of a continuous image along r, an arbitrary vector. Suppose an image is represented by a continuous function f. If the derivative of f with respect to r is taken and it is maximized, the gradient can be found. The direction and magnitude of this gradient will show the direction and magnitude of the maximum change of gray values. Consider the equation (3.1) which shows the derivative of an image function f.

$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} + \frac{\partial f}{\partial t}\frac{\partial t}{\partial r} \tag{3.1}$$

The maximum of $\partial f/\partial r$ can be solved by dividing the problem onto three two-dimensional sections, rather than thinking in three dimensions. The vector r is projected into three major planes: x-y, y-t, and t-x planes. It follows that the derivatives with respect to the angles between the projection of r and the major axes in three major planes



(a)



(b)

Figure 3. Concept of 3-d gradient
(a) 3-d edge surface
(b) projections of r into three planes

should be zero. This can be explained more clearly with Fig. 3(b). The directions and magnitudes of the 2-d gradients, which are the projections of the 3-d gradient onto x-y, y-t and t-x planes, can be obtained separately. Equation (3.2) is the gradient projected onto the x-y plane.

$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} = f_x \cos\theta_1 + f_y \sin\theta_1 \tag{3.2}$$

To find the maximum value of $\partial f/\partial r$, the derivative of the above equation with respect to θ_1 must be zero.

$$\frac{\partial}{\partial \theta_1}\frac{\partial f}{\partial r} = 0$$

$$-f_x \sin\theta_{1m} + f_y \cos\theta_{1m} = 0 \tag{3.3}$$

This results in the two equations for the solution,

$$\theta_{1m} = \tan^{-1}\left(\frac{f_y}{f_x}\right) \tag{3.4}$$

$$\left(\frac{\partial f}{\partial r}\right)_{\max} = \sqrt{f_x^2 + f_y^2} \tag{3.5}$$

Similarly, for the y-t plane the following angle and magnitude for the gradient can be obtained.

$$\theta_{2m} = \tan^{-1}\left(\frac{f_t}{f_y}\right) \tag{3.6}$$

$$\left(\frac{\partial f}{\partial r}\right)_{\max} = \sqrt{f_y^2 + f_t^2} \tag{3.7}$$

Also for the t-x plane,

$$\theta_{3m} = \tan^{-1}\left(\frac{f_x}{f_t}\right) \tag{3.8}$$

$$\left(\frac{\partial f}{\partial r}\right)_{\max} = \sqrt{f_t^2 + f_x^2} \tag{3.9}$$

Equations (3.4) through (3.9) can be merged together to form the following 3-d representations.

$$\left(\frac{\partial f}{\partial r}\right)_{\max} = \sqrt{f_x^2 + f_y^2 + f_t^2} \tag{3.10}$$

$$\theta_{1m} = \tan^{-1}\left(\frac{f_y}{f_x}\right)$$

$$\theta_{2m} = \tan^{-1}\left(\frac{f_t}{f_y}\right)$$

$$\theta_{3m} = \tan^{-1}\left(\frac{f_x}{f_t}\right)$$

The gradient of the image can be represented by a discrete form in three orthogonal directions. The magnitude and the direction of the gradient vector can be described by the following four equations.

$$g(m,n,l) = \sqrt{g_1^2(m,n,l) + g_2^2(m,n,l) + g_3^2(m,n,l)} \tag{3.11}$$

$$\theta_{1m} = \tan^{-1}\left(\frac{g_2(m,n,l)}{g_1(m,n,l)}\right)$$

$$\theta_{2m} = \tan^{-1}\left(\frac{g_3(m,n,l)}{g_2(m,n,l)}\right)$$

$$\theta_{3m} = \tan^{-1}\left(\frac{g_1(m,n,l)}{g_3(m,n,l)}\right)$$

To find the gradient component values of $g_1(m,n,l)$, $g_2(m,n,l)$ and $g_3(m,n,l)$, different masks can be designed, similar to the methods used in the spatial domain (Pratt (1978), Jain (1989)). The same concepts can be applied to the spatio-temporal domain, assuming that the same type of edge exists in the temporal direction. The spatio-temporal edge operators obtain the gradient value by calculating the sum of the differences of neighborhood pixels in the vertical, horizontal and temporal directions. The masks will form a cube in the spatio-temporal domain. The length of the mask in each of the three dimensions should be an odd number so that the size of the mask becomes 3x3x3, 5x5x5, etc. The reason for using an odd number for one side of a mask is that the gradient can be obtained at the center pixel, not at some arbitrary points between pixels. Suppose we try to find a motion vector from a pixel to another in the next frame using spatio-temporal gradients. In this case, the center of the spatio-temporal gradient mask is the starting point of a motion vector. This concept will be discussed in detail in later sections.

3.2 3-d Edge Detection

An image sequence is a set of image frames that are highly correlated. The continuous variation of gray values in three dimensions allows a gradient vector to be calculated. This gradient is three-dimensional and is a combination of spatial and temporal components. The 3-d gradient vector points to the maximum change of gray level in three dimensions as shown in Fig. 4. Here, the plane normal to the gradient vector at the

edge surface is defined as a ***tangent plane***. This plane can be considered as a boundary dividing a small region of the spatio-temporal image space into two half regions, depending upon the intensity. This ***tangent plane*** will be used as a guide in searching for the motion vector in the next chapter.

The three-dimensional gradient estimate is obtained with the cube mask as shown in Fig. 5(a). The 3-d gradient mask cube is a modified version of the Sobel (Jain, 1989) spatial edge detection mask. Sobel gradient masks have the advantage that they are more robust in the presence of noise than other simple gradient operators. The masks are simple extensions of the Sobel masks to three dimensions. The coefficients on this cube and every pixel in the image are convolved. The coefficients in the back of the cube are sign reversed from the pixels in the front. The masks on the cube were designed to reduce the effects of the noise.

The masks in the cubes of Fig. 5(a) can be redrawn as in Fig. 5(b). They are the masks of the cube seen from the spatial (x-y) domain. It shows that eight basic convolutions have to be calculated at every pixel. The first row corresponds to the time axis component, the second row to the x-component in the spatial domain, and the third row to the y-component. In fact there are duplicate computations and there are only three basic operations; the others can be done by multiplying by -1 or 2. Figure 5(b) shows that only the components G_{t2}, G_{x2} and G_{y2} need to be calculated. G_{x1} and G_{y1} are obtained during the computations of the components, G_{x2} and G_{y2}, respectively. Also note that G_{x3} and G_{y3} are the duplicates of G_{x1} and G_{y1}, respectively.

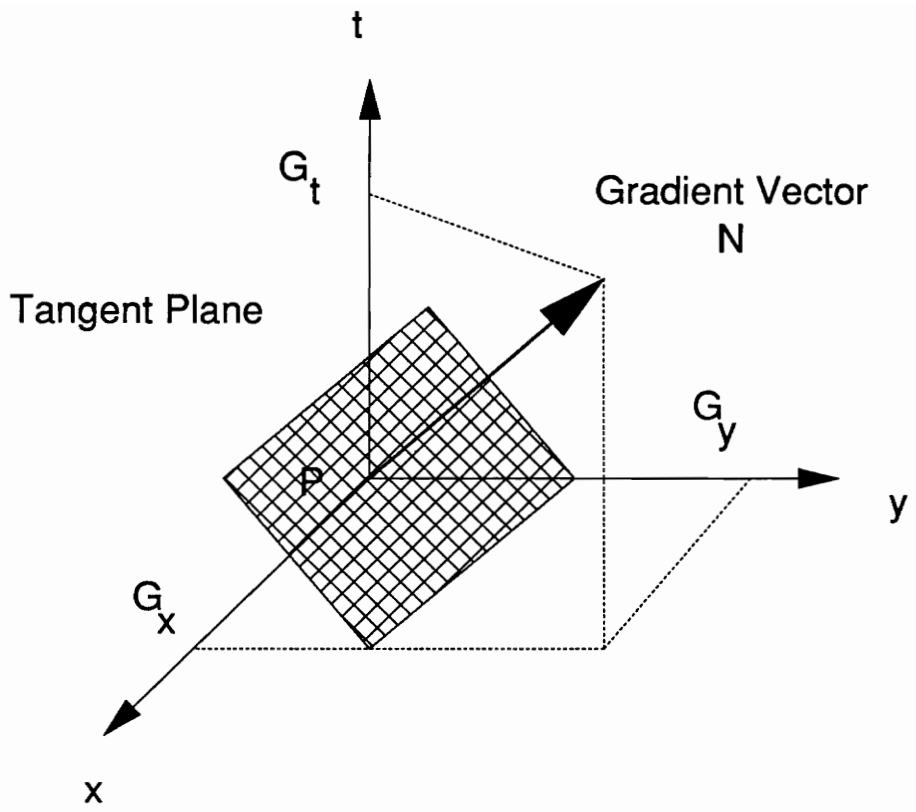


Figure 4. Gradient vector and tangent plane in spatio-temporal domain

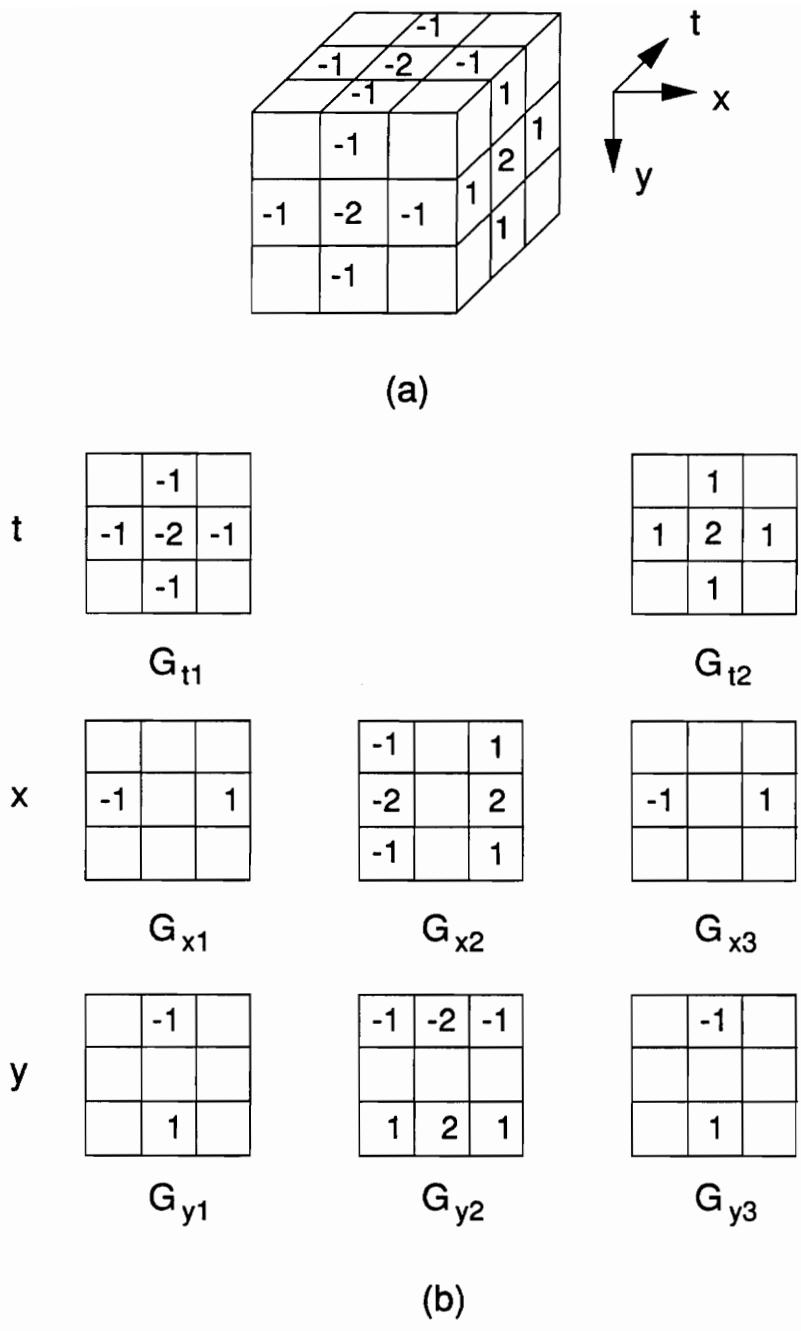


Figure 5. 3-d gradient masks

The masks used in detecting edges in this section are formed from a generalization of the Sobel masks. The edge detection used here works better than the method with just one pixel on each side of the mask because of the fact that it tends to reduce the effects of noise by using neighboring pixels and giving different weights to each pixel. In the following section, a formal approach for finding optimum weights in the edge detection mask will be presented.

3.3 Stochastic 3-d Gradient Vector

Every coded image is corrupted by noise. The noise is normally called quantization noise. This added noise affects the calculation of the gradient. When an edge operator is used to find an edge, it tends to generate false edge points. In this research, the gradient operator is used to estimate the motion. This means that when the image is noise corrupted, there is more chance of inaccurate motion vectors. The prediction errors resulting from this process will in turn degrade the quality of the coded image and contribute to inaccurate motion estimation in coding the next frame. This vicious cycle should be slowed down if not stopped. In order to accomplish this, the gradient operator should be as robust as possible. The pixel values involved in the gradient calculation should be estimated from the pixel values at hand. Hence filtering must be employed in the gradient operator.

The appearance of the quantization noise is different for each type of coding. For instance, noise caused by predictive coding is completely different from noise inherent in transform coding. The reason is that the processes in which the noise is added are completely different. The noise from predictive coding will consist of high frequency noise which will directly affect the objects in an image while the noise addition process in transform coding usually can be described as low pass filtering.

Gradient masks used in the spatial edge operators such as the Sobel operator (Jain, 1989) have the property of averaging, which "somewhat reduces the noise." However, we need a generalized method to find the gradient so that the noise can be reduced in a controlled manner. In other words, the degree of filtering should be varied, depending upon the amount of noise present in the image. The stochastic edge operator suggested by Jain (1982, 1989) may alleviate problems due to the noise. This operator tends to smooth the pixel values before calculating the gradients. The degree of smoothing varies with the SNR of the local image. The 3-d stochastic edge operator, which is just an extension of the 2-d case, will be developed.

To develop a 3-d stochastic edge operator, we need an image model for the spatio-temporal domain. The isotropic image model (Jain, 1989) in the spatial domain assumes that the covariance function depends on the distance between the pixels equally in both the x- and y-directions. If we extend this concept to the spatio-temporal domain, the covariance function for the 3-d case becomes

$$r(m,n,l) = \sigma^2 \exp\left\{ -\sqrt{(\alpha_1 m^2 + \alpha_2 n^2 + \alpha_3 l^2)} \right\} \tag{3.12}$$

where α_1, α_2, and α_3 are the arbitrary constants. If $\alpha_1 = \alpha_2 = \alpha_3 = \alpha$, then r(m,n,l) becomes a function of the 3-d Euclidean distance

$$d = \sqrt{(m^2 + n^2 + l^2)} \tag{3.13}$$

And the covariance function is reduced to:

$$r(m,n,l) = \sigma^2 \rho^d \tag{3.14}$$

where ρ is the one pixel distance correlation and expressed as $\rho = \exp(-|\alpha|)$. If we assume that ensemble averages are equal to sample averages, the mean for the image cube of size $M \times N \times L$ is

$$\mu \cong \hat{\mu} = \frac{1}{MNL} \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{l=1}^{L} x(m,n,l) \tag{3.15}$$

and the covariance function is redefined as

$$r(m,n,l) \cong \hat{r}(m,n,l)$$
$$= \frac{1}{MNL} \sum_{m'=1}^{M-m} \sum_{n'=1}^{N-n} \sum_{l'=1}^{L-l} [u(m',n',l') - \hat{\mu}][u(m+m',n+n',l+l') - \hat{\mu}] \tag{3.16}$$

The symmetry of the covariance function in three directions can be verified by the following procedure. The first-order covariances in three axes, r(1,0,0), r(0,1,0), and r(0,0,1), were found for a test image sequence. The result was

$$r(1,0,0) = 835.71$$
$$r(0,1,0) = 834.61$$
$$r(0,0,1) = 824.11 \tag{3.17}$$

The values of m, n and l are 284, 360 and 50, respectively. As expected, the covariances along the x- and y-directions were about the same, while there is a slight difference in the t-direction. This shows that the proposed isotropic model can be applied to the image sequence globally even though r can vary quite a bit locally from what is indicated in (3.17).

The stochastic edge is found by the calculation of the gradient, which is the difference between two values: one of them is the forward semicausal estimate and the other is the backward semicausal estimate (Jain, 1981). A semicausal model is causal in one coordinate and noncausal in the other. This model is suitable for gradient calculations because it can separate a block into two subblocks. The forward semicausal model can be visualized in three dimensions as in Fig. 6. When the x-directional gradient is to be found in the spatio-temporal domain, the forward semicausal estimator for one side of the gradient can be expressed as

$$\hat{s}(m,n,l) = \sum_{(i,j,k)\in W}\sum a(i,j,k)x(m-i,n-j,l-k)$$

$$W = \{(i,j,k):0 \le i \le p, |j| \le q, |k| \le r\} \tag{3.18}$$

where $\hat{s}(m,n,l)$ is the estimate of the original signal and $x(i,j,k)$ is the corrupted signal. Note that W, the range of (i,j,k), includes the points at (0,0,0) because of the fact that the pixel at the point is already available. This is just an extension of the 2-d case (Jain, 1989) where W includes the point (0,0) where the estimated pixel is present. The coefficients, $a(i,j,k)$'s, for this model can be obtained with the Wiener filtering method (Jain and Ranganath, 1982). The Wiener image filter has the property of reducing both the low frequency and high frequency noise. Low frequency noise normally appears as blur in the image and high frequency noise appears as dots. If there is no blur, it becomes an optimum FIR smoothing filter. In an image sequence, the blur which arises with a abrupt motion is not noticeable because of the masking effect of motion (Netravali, 1980). This blur is not as critical as shot noise in the gradient calculations. Hence, we will consider only the high frequency noise which can occur very randomly. Using the minimum mean square error approach, for the 3-d case with no blur, the equation for the FIR Wiener filter can be written as

$$\left[\frac{\sigma_n^2}{\sigma_s^2}\delta(m,n,l)+r_0(m,n,l)\right]\otimes g(m,n,l)=r_0(m,n,l) \tag{3.19}$$

where $(m,n,l)\in W$ or

$$\sum_{m,n,l\in W} a_{m,n,l}\left[r_s(m'-m,n'-n,l'-l)+\frac{1}{SNR_i}\delta(m'-m,n'-n,l'-l)\right]$$
$$=r_s(m',n',l') \tag{3.20}$$

where $(m',n',l')\in W$. Note that the delta function is defined as

$$\delta(m'-m,n'-n,l'-l)=\begin{cases}1 & m'=m,n'=n,l'=l\\ 0 & otherwise\end{cases} \tag{3.21}$$

and SNR_i, the signal-to-noise ratio of the input is

$$SNR_i=\frac{\sigma_s^2}{\sigma_n^2} \tag{3.22}$$

In the isotropic model case, when we consider only the x-direction gradient and the mask size is $3\times3\times3$, the resulting equation to solve is

$$\sum_{l=-1}^{1}\sum_{n=-1}^{1}\sum_{m=0}^{0} a_{m,n,l}\left[r_s(m'-m,n'-n,l'-l)+\frac{1}{SNR_i}\delta(m'-m,n'-n,l'-l)\right]$$
$$=r_s(m',n',l') \tag{3.23}$$

If we express this as a matrix,

$$\begin{pmatrix} r'_s(0,0,0) & r'_s(0,0,-1) & r'_s(0,1,-1) & \dots & r'_s(0,1,1) \\ r'_s(0,1,0) & \cdot & \cdot & & \cdot \\ r'_s(0,2,0) & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ r'_s(0,2,2) & \cdot & \cdot & & \cdot \end{pmatrix} \begin{pmatrix} a_{0,-1,-1} \\ a_{0,0,-1} \\ a_{0,1,-1} \\ \cdot \\ \cdot \\ \cdot \\ a_{0,1,1} \end{pmatrix} = \begin{pmatrix} r_s(0,-1,-1) \\ r_s(0,0,-1) \\ r_s(0,1,-1) \\ \cdot \\ \cdot \\ \cdot \\ r_s(0,1,1) \end{pmatrix} \tag{3.24}$$

where $r'_s(m,n,l) = r_s(m,n,l) + (1/SNR_i)\delta(m,n,l)$. Note that the matrix R is a $(2M+1)\times(2M+1)$ block Toeplitz matrix of basic dimension $(2M+1)\times(2M+1)$. There are nine coefficients to be found.

The signal-to-noise ratio in (3.22) was obtained by first finding the image variance, σ_s^2, and then finding the noise variance, σ_n^2. σ_s^2 was found by obtaining the image variance for each frame of a test image sequence and averaging them. The same value of σ_s^2 was used in estimating the SNR throughout the whole image sequence. This is due to the fact that the image sequence does not change much between frames in a normal image sequence.

Next, σ_n^2 in (3.22) is estimated. In this case, the noise is the difference between the corrupted image and the original image. In other words, it is the quantization noise. The 3-d gradient operator is applied to a set of three coded frames. It follows that the noise variance should be found for the set of frames from which the 3-d gradient is calculated. In this case, the number of frames to be considered is three because of the 3x3x3 mask cube. The amount of computation to calculate the noise variance for a block of three

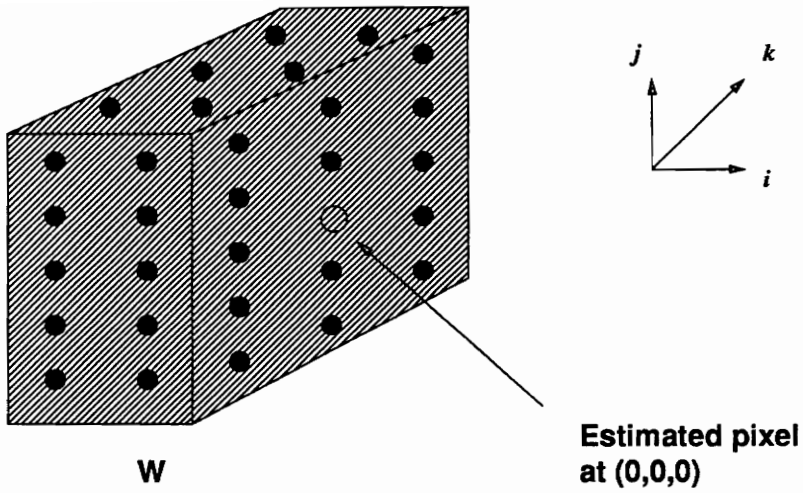


Figure 6. Finite estimation window for 3-d semicausal model

frames is fairly large. Jain (Jain, 1989) demonstrates an adaptive variance estimator for the delta modulator case. This can be applied to the noise variance. The modified equation for the noise variance estimator is

$$\sigma_e^2(j+1) = (1-\gamma)\sigma_e^2(j) + \gamma\,\sigma_{e,new}^2(j+1) \tag{3.25}$$

where $\sigma_e^2(j+1)$ is the estimated variance, $\sigma_e^2(j)$ is the variance of the previous group of three frames, and $\sigma_{e,new}^2(j+1)$ is the variance of the new error frame. The value of γ used in this paper is 1/3 to assign equal weights to the variance of all three frames.

The image and noise variances were used in (3.22) to find the SNR for the present frame. Equation (3.24) was then solved for the optimum coefficients. The coefficients in the opposite side of the mask are just the negatives of the ones found. Also the coefficients of the masks in all three principal directions are assumed the same.

An example of the calculated coefficients for the x-direction in a 3x3x3 cube is shown in Fig. 7. An SNR value of 26 db and $r_s(m,n,l) = (0.9841)^{\sqrt{(m^2+n^2+l^2)}}$ are used for the calculations. The size of the FIR Wiener filter can be selected depending upon the amount of blur and additive noise (Jain and Ranganath, 1982). It follows that the mask size for the gradient operator can be increased. If we want to limit the number of frame memory planes, we may increase only the lengths of the masks in the spatial directions. For instance, the size of the cube can be 5x5x3, 7x7x3, etc..

In real image coding applications it will be too time consuming to calculate the optimum mask coefficients for every pixel in every frame. Although the Wiener filter can be implemented in hardware, it is a complex part of the motion estimation module.

-0.4677		*0.4677*
-0.5535		*0.5535*
-0.4677		*0.4677*

-0.5535		*0.5535*
-1.0		*1.0*
-0.5535		*0.5535*

-0.4677		*0.4677*
-0.5535		*0.5535*
-0.4677		*0.4677*

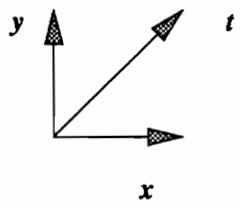


Figure 7. Masks for x-directional gradient in 3x3x3 cube

Instead, the mask coefficients can be calculated for several SNR values at once and stored in a look-up table. In other words, the SNR values are quantized and for each quantized SNR values the corresponding coefficients are calculated.

For maximum reduction of the overhead, the the mask coefficients are fixed for the whole frame. The SNR is calculated for each frame at the encoder, and the corresponding address of the coefficients in the lookup table is sent to the decoder. Both the encoder and the decoder have the lookup table. The SNR is calculated with the estimated noise variance and the fixed image variance, and compared with the SNR values in the lookup table. If the closest value to the present SNR is found in the table, the corresponding coefficients are used for the stochastic gradient calculation. The coefficients for the $3 \times 3 \times 3$ cube were obtained for a SNR range of 5 to 29 dB in steps of 2. The coefficients are shown in Table 1. Note that only two coefficients, A and B, need to be calculated for the mask size of 3x3x3 with isotropic image model. Figure 8 shows how the coefficients are located for obtaining x-directional gradient.

SNR	Coeff. A	Coeff. B
5.0	0.6161	0.6806
7.0	0.5328	0.6096
9.0	0.4677	0.5535
11.0	0.4154	0.5080
13.0	0.3726	0.4704
15.0	0.3370	0.4387
17.0	0.3068	0.4115
19.0	0.2810	0.3880
21.0	0.2587	0.3675
23.0	0.2393	0.3493
25.0	0.2221	0.3331
27.0	0.2070	0.3186
29.0	0.1935	0.3055

Table 1. Coefficients for various SNR values

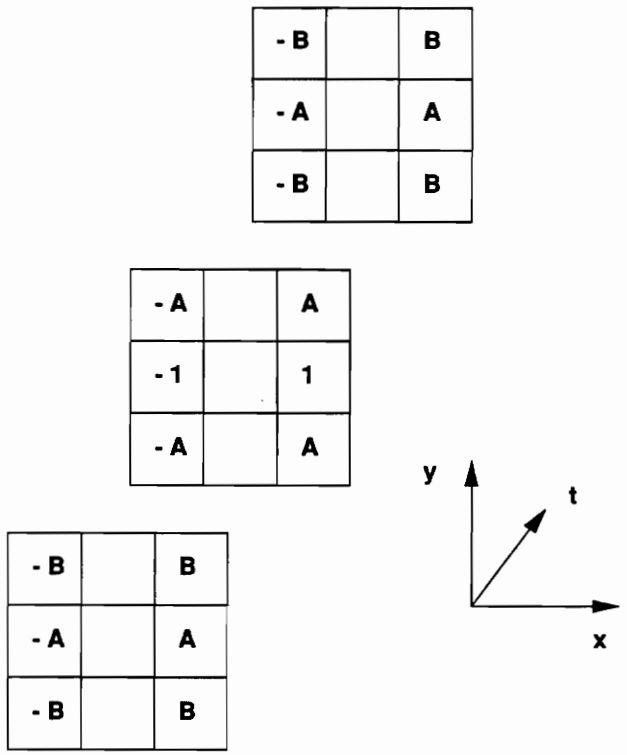


Figure 8. The distribution of coefficients in Table 1

Chapter 4

Interframe Coding with 3-d Edge Detection

4.1 Introduction

Interframe image coding is a direct descendent of intraframe image coding. Although interframe coding deals with a sequence of image frames, the objective is the same in both domains: to reduce the bit rate gracefully, that is without degrading subjective quality. Interframe coding uses the basic tools developed for intraframe coding. Motion compensation is one of the most important and popular interframe coding methods. Motion compensation techniques try to estimate the motion so that the predicted value can be found more accurately. The motion is estimated either on a pixel by pixel or block by block basis. The pixel-by-pixel methods calculate the motion more accurately, however, require greater computational complexity.

One of the disadvantages of the pixel-recursive approaches, such as Netravali *et al.*'s (1979), involve complex calculations, such as integer multiplications. Another disadvantage is the slow convergence to the correct motion vector at the boundaries of moving objects. The aim of this chapter is to find a coding technique that does not involve as much complexity as pixel-recursive approaches but still performs as well as them. The complexity should be moderate, therefore, the system should be easily implemented in hardware with current technology.

4.2 Motion Vector Estimation from the Spatio-Temporal Gradient

4.2.1 First method

The motion trajectory of a pixel in an image frame can be estimated with the ***tangent plane*** mentioned in Section (3.2). This ***tangent plane*** is normal to the direction of the gradient vector. Hence the probability of the differential motion trajectory being on this plane is high. The problem of finding the motion trajectory can be reduced to searching for the pixel on the ***tangent plane*** closest in value to the pixel at the center of the spatio-temporal gradient mask cube.

If the motion of an object is observed in the three-dimensional spatio-temporal domain, and is approximately linear in several consecutive image frames, its trajectory can be seen to form a three-dimensional object. If the object in the spatial domain is a circle, then its trajectory makes a cylinder as shown in Fig. 9. Consider one pixel, X in Fig. 9, on the circle at frame n. There exists a three-dimensional gradient vector at this pixel pointing towards the center of the circle and a gradient vector with same magnitude and direction at the corresponding pixel in frame n+1.

There is a ***tangent plane*** which is normal to the 3-d gradient vector. This plane contains both the pixel X in frame n and the corresponding pixel in frame n+1. The ***tangent plane*** can be obtained by simple geometry using just the directions of the 3-d gradient vector at pixel X of frame n.

As shown in Fig. 4, the ***tangent plane***, which is normal to the 3-d gradient vector N and which passes through a point P, can be expressed with the following dot product relation.

$$X \cdot N = P \cdot N \tag{4.1}$$

where $N = (G_x, G_y, G_t)$ and $X = (x, y, z)$. In this case, $P = (0,0,0)$ and it follows that

$$X \cdot N = 0 \tag{4.2}$$

which is the same as

$$((x, y, t), (G_x, G_y, G_t)) = 0 \tag{4.3}$$

or

$$G_x \cdot x + G_y \cdot y + G_t \cdot t = 0 \tag{4.4}$$

As shown in Fig. 10, the pixel on the moving object will change position. The ***tangent plane*** through the center pixel at frame n-2 is projected to the frames n-1 and n. Then one of the pixels on the line that is the intersection of the frame n and the ***tangent plane*** will be the center pixel at frame n-2 if the motion is linear.

To decide the correct path of the pixel, first the pixel in frame n-1 that corresponds to the center pixel at frame n-2 is found. The projection of the ***tangent plane*** from frane n-2 to frame n-1 will be a line which can be expressed as

$$y = \frac{-G_x x - G_t}{G_y} \tag{4.5}$$

and the dotted line in Fig. 10 represents this equation.

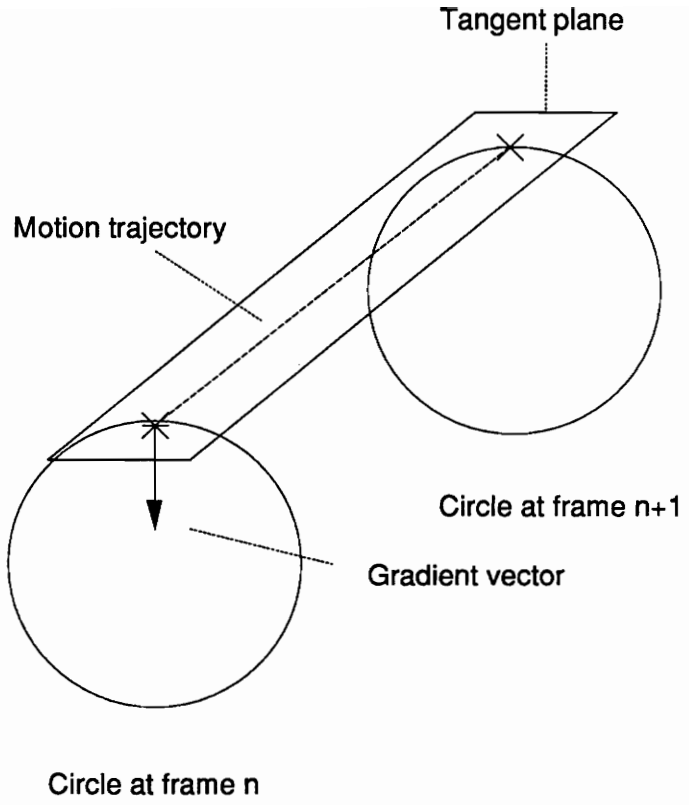


Figure 9. Motion of a circle

The pixels that are closest to this line are grouped, the differences between these and the projecting pixel at frame n-1 are sorted, then the minimum difference is found. A window is set up in frame n-1 to give the maximum range for the search of the closest pixel value.

If there is no pixel within the window which is close enough in value to the center pixel at n-2, then it is assumed that the motion is not linear. A threshold determines the maximum difference allowed. Then, as in Fig. 10, the vector to the pixel with minimum error is obtained and extended to frame n. If the vector to frame n-1 was (a,b) then the extended vector to the frame n would be $2 \cdot (a,b)$ from the center pixel at frame n-2. The location of the pixel at frame n which is projected by this vector is recorded with the projecting pixel value from frame n-1. Then when the projected pixel at frame n is to be coded, this previously stored value is used as the predicted value.

Equation (4.5) involves one division operation, which adds a large amount of computational complexity. However, the intersection line doesn't have to be calculated accurately if the search window is not too large. Hence the direction and deviation of the line can be quantized. The problem of complexity can be alleviated if a division lookup table is used. First, the gradient values have to be linearly quantized because the range of these values are too large. From the experiments with a test sequence, the range was found to be from -1530 to 1530. Next, the slope of the line can be limited to eight angles, from 0 degrees through 157.5 degrees in steps of 22.5 degrees. Also the intersections with the y-axis can be quantized into nine steps. The result is 72 addresses in the table. Combinational logic can be designed to map the previously obtained quantized gradient values into the 72 possible addresses. The entries in the table are the sets of candidate pixels for the search.

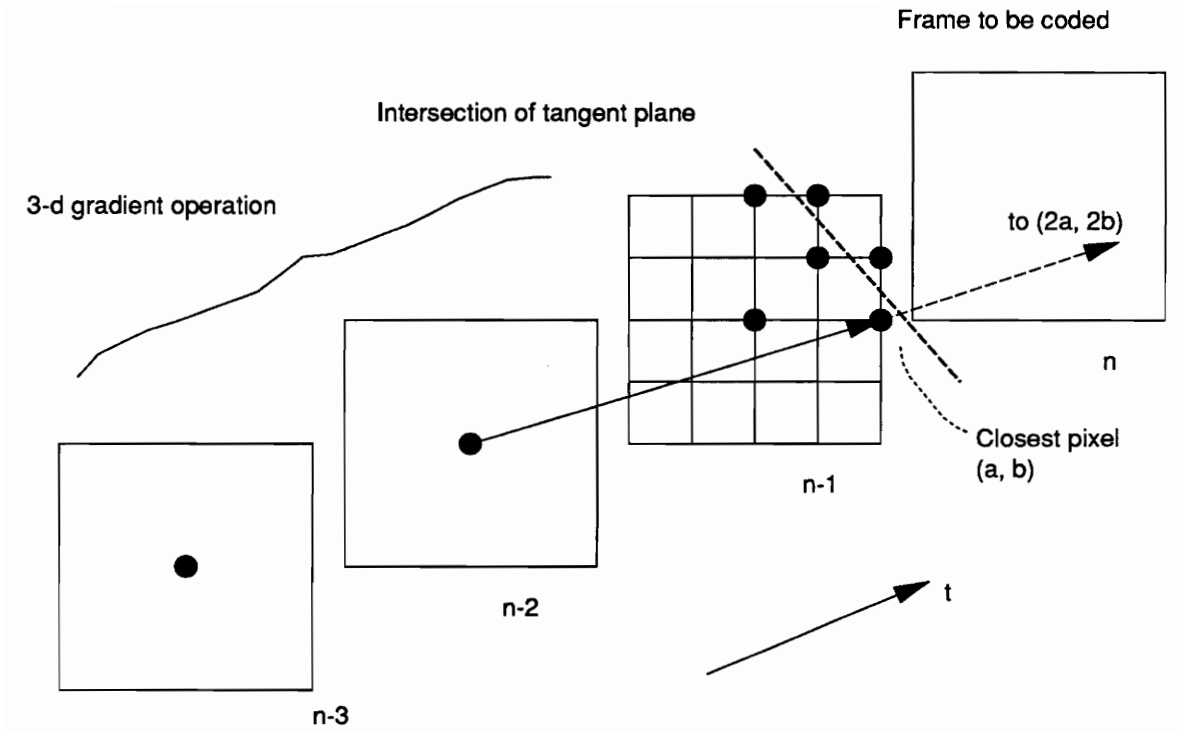


Figure 10. First method for searching motion vector

When one of the components of the motion vector is greater than 3 pixels, the masks considered above are too small to handle this. The maximum amount of motion in a normal videophone sequence is approximately 5 to 6 pixels. Hence the mask size should be large enough to handle such motions. The constraint is that the side of the mask should be an odd number of pixels, greater than 3, to be consistent with the gradient calculated previously. But if the mask size is increased, the complexity of the gradient estimation increases enormously. This problem can be partly solved by using the ***resolution pyramid.*** The size of the mask is increased only in the spatial domain, not in the temporal domain. The new window size is set to 9x9 and the 9 pixels in each 3x3 subarray are merged together to form one pixel in a higher level of the pyramid as in Fig. 11. But the problem of calculating the pixel values in the higher level remains.

It was noted that we have already calculated a component of the gradient masks of size 3x3 which works like a smoothing filter. These values are stored for later calculation of the time axis gradient component in 3x3x3 cube. With the filtered values, a new 3x3x3 cube is formed. The same operation applied to the original 3x3x3 cube can be employed here. In the hardware sense, the same arithmetic unit used in the previous gradient calculation can be reapplied. For further reduction in calculation time, another identical arithmetic unit can be operated in parallel.

In any case, the values calculated from the previous frames must be saved in order to obtain the gradient vectors. A new 3-d gradient cube can thus be formed, using the smoothed values three pixels apart. A new gradient vector is obtained with these new 3-d masks at the higher level.

The operations described above are done only when the spatial domain gradient at the center pixel of the mask cube is above a certain threshold. In other words, the tangent

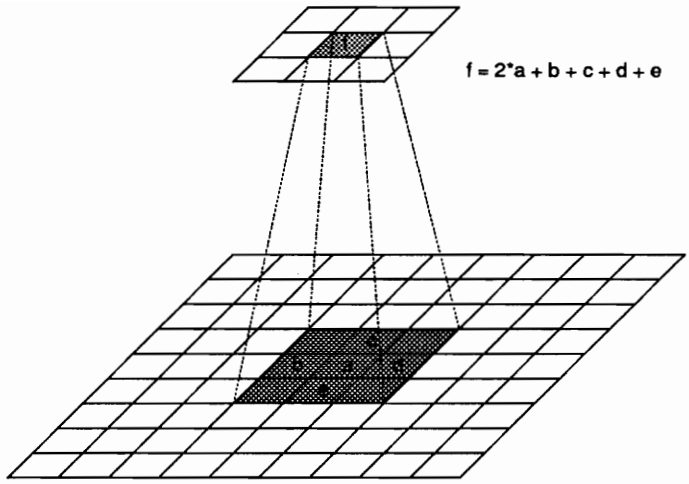


Figure 11. Resolution pyramid used in calculation of large motion vector

plane searching is done only from the center pixel which is a spatial edge point. If the maximum of the gradient components, G_{x2} and G_{y2}, is greater than a threshold, the center pixel is declared an edge point and the motion estimation is executed.

The algorithm can be described in the following pseudo-code.

(1) Calculate G_{t2}, G_{x1}, G_{x2}, G_{y1} and G_{y2}, the five components of the gradient vector (refer to Fig. 7).

(2) Together with the components from the previous frames, calculate the gradient vector, (G_x, G_y, G_t).

(3) If Max (G_{x2}, G_{y2}) > T_1,

project the ***tangent plane*** to the frame at n-1 from the frame at n-2. Search for the pixel having the value closest to the center pixel value at frame n-2 among the tangent plane projection pixels in frame n-1 as mentioned in Fig. 10. If the minimum value is less than a threshold, go to step 4. Else do the following.

(3.1) From the previously found component G_{t2}, form the new 3-d gradient masks in the higher level of resolution pyramid and calculate the new gradient vector for this level.

(3.2) Search for the pixel having the closest value in frame n-1 and if the minimum value is less than a threshold go to step 4. Otherwise, the predicted pixel is the pixel in the same coordinate in the frame n-1.

(4) The difference between the predicted value and the present pixel value is measured. If it is less than a threshold, do not send the difference. Otherwise, quantize it and send it to the channel.

The algorithm described above is applied only to moving pixels. The moving pixels are determined by comparison of the frame differences with a threshold. This is similar to the method applied in (Netravali *et al.*, 1979) except that this algorithm considers only the frame difference of the pixel to be coded, not the differences of the neighboring pixels. If the difference is less than the threshold, in this case 3, it is considered "not moving" and 0 is sent to the receiver. Otherwise, it is considered moving and the operations discussed above are applied. If the prediction error does not exceed another threshold, which is taken here the same as the former, it is considered compensable and 1 is sent to the receiver and the error is not sent. If it exceeds the threshold, it is regarded as uncompensable, the prediction error is quantized to be sent as in (Netravali *et al.*, 1979), and 2 is sent as the indicator.

The problem with the above scheme is that the search for the pixel with the closest value can be expensive. Also, even if the closest pixel to the pixel in frame n-2 has been found in frame n-1, there is no guarantee that the pixel projected with the extended vector in frame n will be the closest to the one in frame n-1. Taking this fact into account, a new scheme has been developed which does not involve much search in the next section.

4.2.2 Second method

The motion estimation method in the previous section involves searching which can add complexity to the estimator. When more restrictions are given to the motion trajectory, a simpler method without searching can be developed. Refer to Fig. 12. The pixel at frame n-2 has a 3-d gradient vector obtained with the gradient masks discussed above. Also there is a 2-d gradient vector which indicates the direction of the maximum change in spatial domain. If it is assumed that the motion is nearly translational, the direction of the 2-d gradient vector for the corresponding pixel in frame n-1 should be

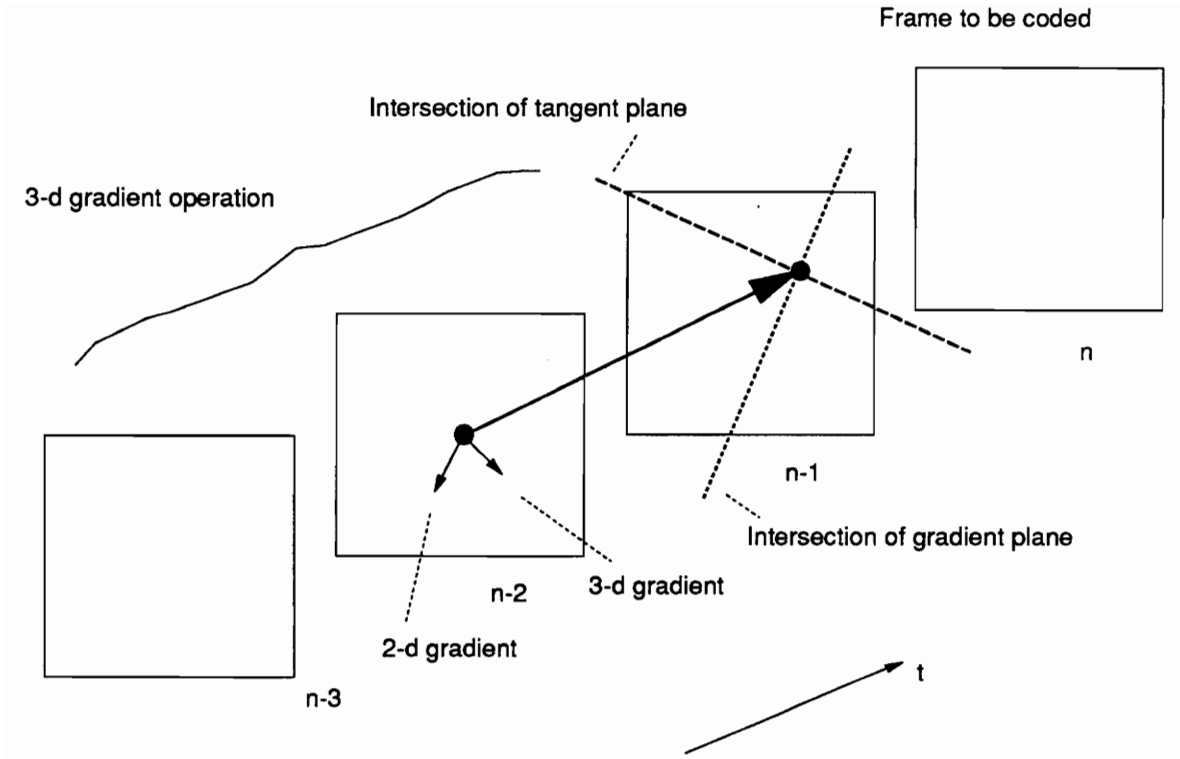


Figure 12. Second method for searching for the motion vector

same as the one in the center pixel in frame n-2. Here, the motion vector can be assumed to be orthogonal to the direction of the 2-d edge. From Section (4.2.1), the motion vector is normal to the 3-d gradient. One of the pixels in the ***tangent plane*** as mentioned in Section (4.2.1) will have the same 2-d gradient and 3-d gradient vectors. With the two previous constraints, the 2-d gradient and 3-d gradient vectors form a plane, which will be called a ***gradient plane***. The intersection of the ***tangent plane*** and this ***gradient plane*** is the position of the moving pixel. The equation for the ***gradient plane*** can be calculated from the following derivation. The equation for a plane passing through three known points (x_1, y_1, t_1), (x_2, y_2, t_2) and (x_3, y_3, t_3) in the spatio-temporal domain is

$$\begin{vmatrix} y_2-y_1 & t_2-t_1 \\ y_3-y_1 & t_3-t_1 \end{vmatrix}(x-x_1)+\begin{vmatrix} t_2-t_1 & x_2-x_1 \\ t_3-t_1 & x_3-x_1 \end{vmatrix}(y-y_1)+\begin{vmatrix} x_2-x_1 & y_2-y_1 \\ x_3-x_1 & y_3-y_1 \end{vmatrix}(t-t_1)=0 \tag{4.6}$$

For the gradient plane, we have three points, $(0, 0, 0)$, (a, b, c) and (a', b', c'). Then (4.6) becomes

$$(bc'-b'c)x+(a'c-ac')y+(ab'-a'b)t=0 \tag{4.7}$$

When t = 1, i.e., the next frame, (4.7) becomes

$$(bc'-b'c)x+(a'c-ac')y+(ab'-a'b)=0 \tag{4.8}$$

If we substitute the vectors (a, b, c) and (a', b', c') with $(G_x, G_y, 0)$ (the 2-d gradient vector at the center pixel in frame n-2) and (G'_x, G'_y, G'_t) (the 3-d gradient vector) respectively, (4.8) becomes

$$G_y x-G_x y+\frac{G_x G'_y-G'_x G_y}{G'_t}=0 \tag{4.9}$$

The expression for the ***tangent plane*** at t = 1 is, from (4.4),

$$G'_x x + G'_y y + G'_t = 0 \tag{4.10}$$

The pixel sought is on the line intersecting the ***gradient plane***. It follows that the intersection point of this line and the tangent line in frame n-1 becomes the candidate for the corresponding pixel. If we solve for x in (4.9) and (4.10),

$$\frac{-G'_x x - G'_t}{G'_y} = \frac{G_y G'_t x + G_x G'_y - G'_x G_y}{G_x G'_t} x \tag{4.11}$$

The x- and y-coordinates of the projection point can be calculated.

$$x = \frac{G'_x G_y G'_y - G_x G'^2_y - G_x G'^2_t}{(G_x G'_x + G_y G'_y) G'_t} \tag{4.12}$$

$$y = \frac{G_y}{G_x} x = \frac{G_y}{G_x} \frac{G'_x G_y G'_y - G_x G'^2_y - G_x G'^2_t}{(G_x G'_x + G_y G'_y) G'_t} \tag{4.13}$$

The equations above can be simplified if we assume that the 2-d gradient vector is simply the projection of the 3-d gradient vector onto the spatial domain (x-y plane). The reason for distinguishing the two vectors is that sometimes the two gradient vectors can be obtained with different types of operators with different constraints. For instance, the 2-d gradient can be calculated with just the samples from the current frame. If the above equations are simplified with the assumptions, $G_x = G'_x$ and $G_y = G'_y$, (4.12) and (4.13) become

$$x = \frac{-G_t' G_x}{G_x^2 + G_y^2} \tag{4.14}$$

$$y = \frac{G_y}{G_x} x = \frac{-G_t' G_y}{G_x^2 + G_y^2} \tag{4.15}$$

If this pixel value is close enough to the projecting pixel value in from frame n-2, the vector is simply extended to frame n as in the previous method. The equation for the projection vector is more complex than in the previous method, but it involves less searching than in the previous method. Intuitively, the point to be found is simply the intersection of the ***tangent plane*** and the line extended from the 2-d gradient vector at frame n-1. Note that this extended line is normal to the tangent line at frame n-1. This means that motion can be detected only when it is in the direction of the 2-d gradient vector, which is normal to the edge direction. This assumption matches with the one in the work of Labit *et al.* (1982, 1983). The difference is that the moving pixel can be estimated without searching.

4.2.3 Modifications

There are two main problems with the 3-d prediction method described above. Inspection of motion vector maps shows that most of the uncompensable pixels occur for two reasons. The first is that multiple projections from the previous frame can occur. In other words, there can be multiple hits in the same pixel in the frame to be coded. This phenomenon can be caused by incorrect motion vectors and noise in the image sequence.

The incorrect motion vectors are due to several factors. One of them is that the projections are made in units of pixel distances so that no projection falls between the pixels. This can cause several projections from the previous frame to fall on the same pixel. The ambiguity arises when a decision has to be made for choosing the correct

projection. However, if the multiple projections are due to this factor alone, the problem is not severe. In this case, all the projection vectors will be close to each other and will give similar prediction values.

The problem becomes severe when the noise component or the nonlinearity of motion causes multiple hits on the same pixel. In this case, a sudden change of motion vector can occur. This can be explained better as a noise in the motion vector field. In the motion field it is unlikely that the motion of one pixel will be very different from those of the neighborhood pixels. This fact is mentioned by Hildreth (1984) as the smooth velocity field constraint. The constraint is that if we assume that most of the objects in the real world have smooth surfaces, then such objects in motion create a smoothly varying velocity field. To solve the first problem, we use following the considerations.

If the multiple vectors projected onto the same current pixel are stored and a decision is made as to which of these is most suitable, the algorithm may become too complex. Also, because of the uncertain number of vectors which are projected to the same pixel, reservation of memory for the vectors will be a difficult task. Instead, the decision can be made on a vector-by-vector basis. This means that the decision is made between two vectors: past and present. Whenever a vector is projected to a pixel which has already been projected, a decision must be made whether or not to update it.

The scheme works as follows. Consider Fig. 13. The differences between the new vector, $\vec{d}_1$, which hits at the pixel X, and the neighborhood vectors $\vec{d}_i$'s in the hit region are added and compared to the sum of the differences between the existing vector, $\vec{d}_2$, and the neighborhood vectors. The comparisons are done in components. If the sum of the differences for the x-component of the new vector is lower,

$$\sum_{i \in \{A,B,\ldots,H\}} |d_{1,x} - d_{i,x}| < \sum_{i \in \{A,B,\ldots,H\}} |d_{2,x} - d_{i,x}| \tag{4.16}$$

the existing vector, $\vec{d}_2$, is replaced with the new one, $\vec{d}_1$. Similarly, the same operation is performed on the y-component.

The second problem is that there can be regions where there are no projections at all. The latter problem can be approached with the following methods. First, the missing motion vector can be estimated with one of the vectors in the causal neighborhood region. In Fig. 13, one of the pixels A, B, C or D which gives the lowest prediction error is chosen and its motion vector is used for the present pixel. The second method is to average the motion vectors of the same neighborhood pixels as described above. Lastly, if a delay of one line to the coding can be tolerated, the window of the first method can be modified to include the pixels in the line next to the present pixel (future). This method causes one line of delay in the coding and can be described as delayed-decision coding. Before finding the present projection vector, the projected pixels in the non-causal region should be coded. The candidates of the reference pixels include all the pixels in the causal region and the projected pixels in the noncausal region.

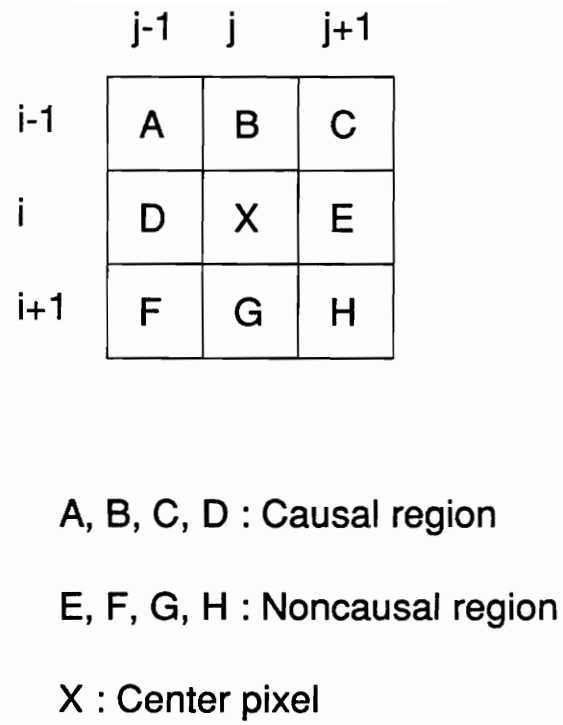


Figure 13. Neighborhood of the unhit pixel

Chapter 5

Combinations with Vector Quantization

5.1 Introduction

One of the main goals in image coding is to compress the image as much as possible without much degradation in quality. Using scalar quantization combined with conventional coding techniques, such as predictive or transform coding, high compression ratios cannot be obtained without severe distortion in the quality of the coded image. For low bit rate coding, vector quantization (VQ) has proved to be very effective. By effective we mean that for the same compression ratio, the subjective quality as well as mean square error (mse) is superior to the scalar quantization methods. VQ has been mainly used as a tool for compression of original image samples (Gray (1984), Nasrabadi and King (1988)).

Recently, VQ has been applied to other kinds of signal, such as prediction error signals (Murakami *et al.* (1984), Bage (1986), Furner *et al.* (1986), Nasrabadi *et al.* (1989a, 1989b)). The application of VQ to prediction error signals is not an easy task. The characteristics of the error map are different from those of the original image. The most noticeable difference is that the error image has two main components: smooth surface and pulsive components (Kaneko et al., 1987). The pulsive component is generated by the incorrect reconstruction of moving boundaries. Even if the motion estimator is very accurate, the absolute value of the error along the boundary will be higher than that of its neighborhood in the local sense. These pulsive components appear in the coded image as shot noise.

The first two methods in this chapter do not include techniques to compensate for this effect, and the results will be shown in the next chapter. The first method vector quantizes the error image straightforwardly without any modification while the second method tries to classify the block into several classes depending upon the type of edges it contains. Then a method which remedies this effect will be advanced for future study.

5.2 3-D Gradient Motion Estimation for Vector Quantization

5.2.1 Motion Segmentation

In an videoconferencing image sequence, a large part of the image is not moving. The image can be subdivided into moving and nonmoving parts. In a pixel-by-pixel motion estimation approach, the segmentation is performed for each pixel, while in a block-by-block approach, it can be done for each block. Even though we apply the pixel-by-pixel method here, because of the fact that VQ operates on a block, block segmentation is suitable. The following method is used in motion segmentation. For each block to be vector quantized, the difference between the block in the present frame and the corresponding one at the same location in the next frame is obtained. The average of the differences for each element in the block is obtained. This value is compared to a predetermined threshold and if the value is larger than the threshold, the block is designated as a moving block and if not, a nonmoving block. If the following equation is satisfied,

$$\frac{1}{N}\sum_{\substack{i=1\\j=1}}^{N}(I_{n+1}(i,j)-I_n(i,j)) \quad > \quad T_1 \tag{5.1}$$

the block is considered as moving. Otherwise, it is not moving. Here, I_n is the decoded image and I_{n+1} the real image. I represents the gray values, n the frame number, N the number of pixels in one block, and T_1 the threshold of determining motion. A sample motion segmentation map is shown in Fig. 14. Note that one element in this image corresponds to a block of size 4×4. The bright spots show the moving blocks and the rest is nonmoving. The size of this map is $(hsize/4) \times (vsize/4)$, where hsize is the horizontal length of the image and vsize is the vertical length of the image. Both hsize and vsize are divided by 4 because the block size used in this research is 4×4.

For the decoder to be able to know which block is moving, a motion segmentation bit must be sent to the decoder. This scheme is similar to the one used in the scalar motion compensation system in the previous chapter except that now each bit is for one block, not one pixel. If the motion detector detects motion, the codebook address of the vector quantized difference block is sent to the decoder. Otherwise, the decoder is informed that the block is compensable and nothing is sent to the decoder. This scheme adds one bit per block to the overhead. If the block size is large, the overhead due to this compensation bit will be negligible.

Figure 14. A sample block motion segmentation map

5.2.2 Pixel Motion Estimation

If 3-d gradient motion estimation is used, the motion vectors are projected from the previous frame to the present frame to be coded even before the present frame is input. The projections to the present frame are put into memory, so that whenever a pixel in the present frame is coded the prediction value from the previous frame can be obtained with the motion vector. There are times when no prediction value exists for the pixel to be coded. This is the case when no projections are made from the previous frame. In this case, the motion vector is calculated from the vectors found for the neighborhood pixels. The neighborhood method used in the pixel-by-pixel approach in Chapter 5 may be considered but the block-by-block nature of vector quantization prevents its usage.

A simple strategy is to average the motion vectors for the neighborhood pixels. For a 3×3 window, refer to Fig. 13. Note that the group of eight neighborhood pixels can be subdivided further into two: the causal and noncausal regions. The causal region is defined here as the region where the missing motion vectors have already been found. For ease of computation, it is convenient to scan the block from left to right and top to bottom. The causal and noncausal regions result from this scanning methodology. For the causal region, every neighborhood pixel is used for the computation of the missing motion vector at the center. But for the noncausal region, only the motion vectors of the projected pixels are used. The equation for the calculation of the missing motion vector can be expressed as

$$\mathbf{v}_{\text{missing}} = \frac{\sum_{\text{causal}} \mathbf{v}_i + \sum_{\substack{\text{noncausal} \\ \text{projected}}} \mathbf{v}_i}{n_{causal} + n_{np}} \tag{5.2}$$

where $\mathbf{v}_i$'s are the motion vectors, n_{causal} is the number of causal pixels, and n_{np} is the number of projected pixels in the noncausal region. The algorithm is explained in detail with the following pseudocode.

1. For the causal region (pixels, A, B, C and D in Fig. 7), calculate

$$SUMC = \sum_{A,B,C,D} v_i$$

2. For the noncausal region (pixels, E, F, G and H), count the projected pixels and let the sum of the projected pixels be SUM1

   If SUM1 > 0,

   Find $\text{SUMU} = \sum_{\substack{noncausal \\ projected}} v_i$

   Else

   SUMU = 0

   End if

3. Calculate $v_{missing} = \dfrac{SUMC + SUMU}{4 + SUM1}$

5.2.3 Block motion estimation

Block motion can be estimated from the pixel motion vectors. This method has not been popular until now because pixel motion estimation is relatively expensive to implement. One of the most widely used methods in block motion estimation is searching through the blocks in the previous frame and finding the block which matches

the present block best in the mean-square-error sense (Jain *et al.*, 1981) (Koga *et al.*, 1981) (Srinivasan *et al.*, 1984). Note that in this method, a complex search has to be done on a block-by-block basis. The whole process is serial: searching and coding.

In 3-d gradient motion estimation, the pixel motion vectors are determined before the new frame enters into the system. The 3-d gradient motion estimation is totally parallel to the coding procedure and does not need to be done during the block coding. Figure 15 shows the basic flow chart of this process. The pixel motion for the present block is already present when it is the block's turn to be coded. However, the block motion estimation should be done immediately before vector quantization because the block motion cannot be obtained while the pixel motion projections are still in progress. It follows that the pixel motion vector for the next frame must be obtained in parallel with the calculation of block motion vector and vector quantization of difference blocks for the present frame.

In the block, some pixels will have projections from the previous block and the others won't, as shown in Fig. 16. The block motion has to be calculated from these pixel-to-pixel projections in the block. The differences among the pixel motion vectors in the block may be large due to intrablock motion, inaccurate motion estimation, etc.. In deciding the motion vector for the whole block, it is important that the effect of the inaccurate motion vectors be small. One of the simpler techniques to achieve this is to filter the motion vectors in the block, then consolidate them. This operation is similar to the hierarchical structure of the quadtree used frequently in the field of computer vision. In this case, consolidation is done on the motion vectors, instead of the gray values. This process is explained in Fig. 16.

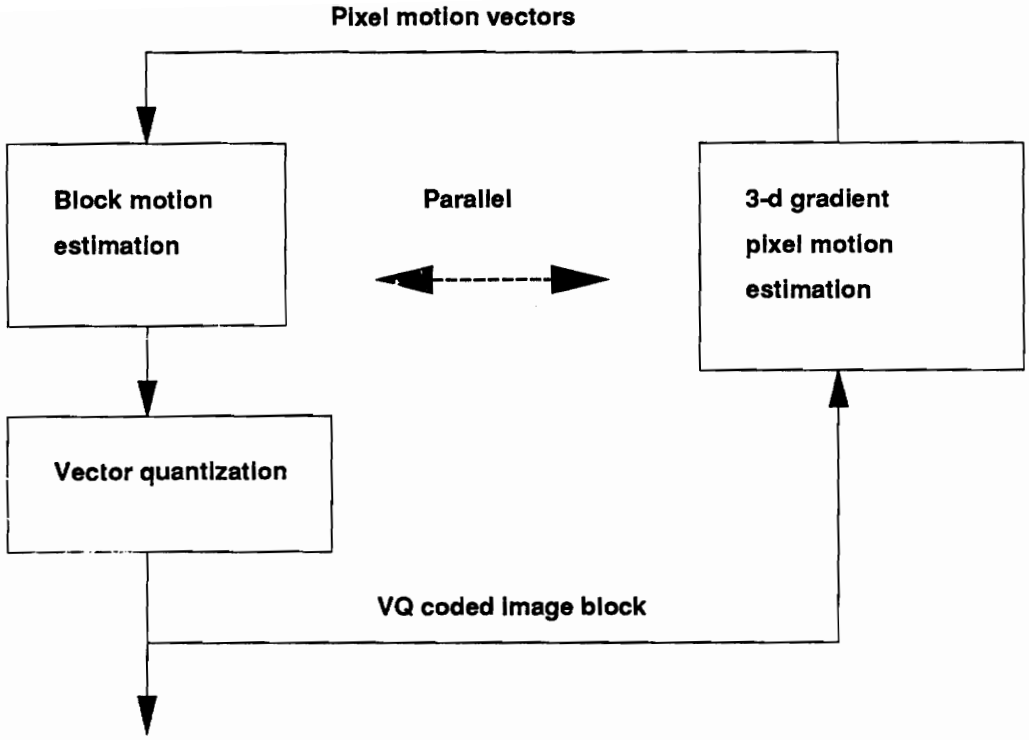


Figure 15. Flowchart showing the parallel operation of 3-d gradient motion estimation and block coding

A simple operation is averaging of the pixel motion vectors in the new block. This scheme may not give as accurate matching as many of the existing block motion estimation methods, but the main advantage is that the motion vectors for the block don't have to be transmitted, resulting in smaller overhead. All the information needed in the calculation of the block motion vector is available also at the receiver. The equation for averaging can be written as

$$\mathbf{v}_{\text{block}} = \frac{\sum_{\text{projected pixels}} \mathbf{v}_i}{n_{pp}} \tag{5.3}$$

where $\mathbf{v}_{block}$ is the block vector, $\mathbf{v}_i$ the pixel motion vectors and n_{pp} the number of projected pixels in a block. In the case of no projection on a block, the motion vector refers to the causal neighborhood block which is quantized with the code vector with lowest energy. Note that the code vector is a difference signal block and its energy indicates the degree of the code vector mismatch. The calculated block vectors may not be as accurate as the ones found with block searching methods with the transmission of motion vectors. However, the difference between the predicted block and the new block can be quantized with block quantization methods such as vector quantization to reproduce the original block more faithfully. The following pseudocode describes the algorithm more clearly.

1. Count the number of projections in a block, SUM1.
2. If SUM1 > 0

    Sum up the pixel motion vectors in the block (SUMP).

    Divide SUMP by SUM1

    Block motion vector is SUMP/SUM1

  Else

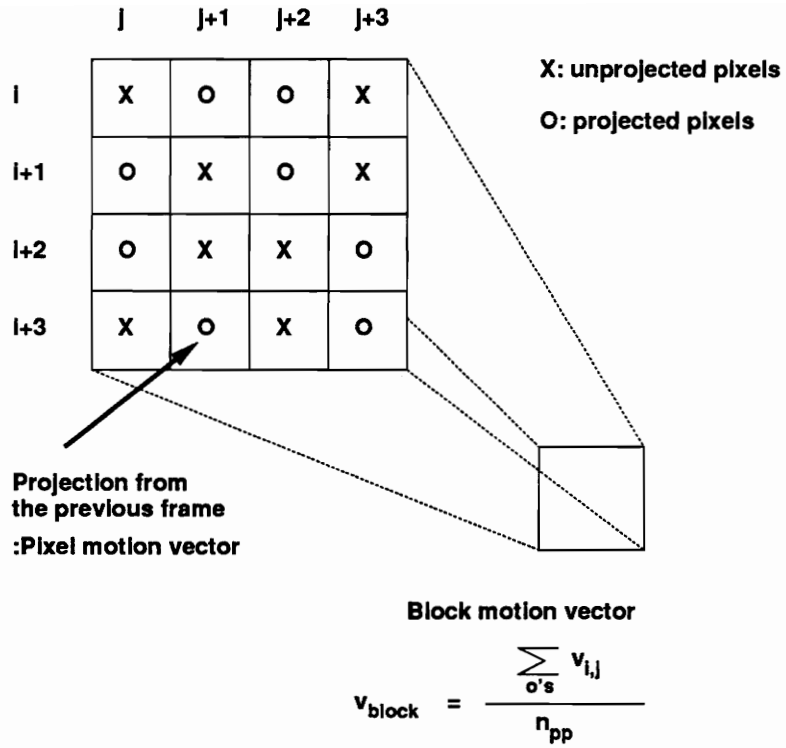


Figure 16. Consolidation of pixel motion vectors

Calculate the energy, E(I), in the code vector for each causal neighborhood block as defined in Fig. 7.

$$E(I) = \sqrt{\sum_{K=1}^{N} P(K)^2}$$

where N is the number of pixels in a block (16), and P(k) an element in the code vector

Find the block with the minimum energy among the neighborhood and its block motion vector, BVEC1.

Set the present block motion vector equal to BVEC1

End if

3. Vector quantize the difference between the present block and the predicted block indicated by the block motion vector

5.3 Vector Quantization of Difference Blocks

5.3.1 Method 1: Straightforward vector quantization

5.3.1.1 Initial codebook generation

For the generation of an initial codebook, many methods were considered. The first method to be considered is the product code mentioned in Section (2.4). If a product code is to be used, the scalar quantization has to be applied to each element in the vector. Suppose the dimensionality of the vector is k. It is usually convenient to make k equal to 2^q where q is an integer. The reason is that the address of the codebook is sent in a binary number with length q and the number of possible vectors is 2^q. Suppose the block size is $m \times n$ and each image sample is quantized to 8 bits. The number of the distinct

vectors is $8^{m \times n}$ which equals $2^{3(m \times n)}$. The problem is to make each vector in the initial codebook as widely spaced as possible. The reason is that the codevectors should represent as many real world vectors as possible. Assuming that we don't know what kind of image data is going to be coded, the distribution of data to each element of initial vector should be equal. Suppose m=n; it must then be true that

$$k = r^{n^2} \tag{5.4}$$

where r is the number of levels in each element. Then there is no problem of making a set of vectors in which all the elements are different. But in order for k to be 2^q, the relation

$$r = (2)^{\frac{q}{n^2}} \tag{5.5}$$

should be satisfied. Note that to increase r, the number of levels in each element, the ratio of q/n^2 must be a high number. Rather than increasing q, related to the codebook size, we can decrease the number n^2. This means we have to decrease the block size. However, in that case the total bit rate will increase. So the best alternative is to divide the vector into a few regions so that each region has the same level. One way is to divide the original vector by multiples of 4, not giving any preference to any edge directions or areas. The division should be done as much as possible to preserve the details in the image. The worst case is that the whole vector in the initial codebook is coded with just one set of gray levels. In this case, the whole vector is treated as one pixel so that the activities inside the vector will be ignored.

The next initial codebook generation method considered was the random code method mentioned by MacQueen (1967). In this scheme, the first N vectors in the training sequence, in this case the first few vectors in the error image sequence, are used as the vectors in the initial codebook. Here, N is the size of the codebook. Note that the initial codevectors have to be obtained from the error images resulting from lossless coding without VQ in the loop. This means that the error image which is the output of the 3-d edge motion estimation block without any quantization is used as the source for initial codevectors. It was suggested by Gray (1984) that in highly correlated data it is recommended to select several widely spaced vectors from the training sequence. Here "widely spaced" means that the differences within the initial codebook have to be sufficiently large. The procedure of initial codevector selection is given in the following pseudo-code.

1. Input the new candidate vector, $\mathbf{v}_k$.

   If there is no more vector, go to 3.

2. If Max (Abs($\mathbf{v}_k$ - $\mathbf{v}_i$)) > T) for i = 1, ..., L Then

      Include $\mathbf{v}_k$ as the new codevector. L = L + 1.

      Go to the step 1.

   Else

      Go to the step 1.

3. Stop

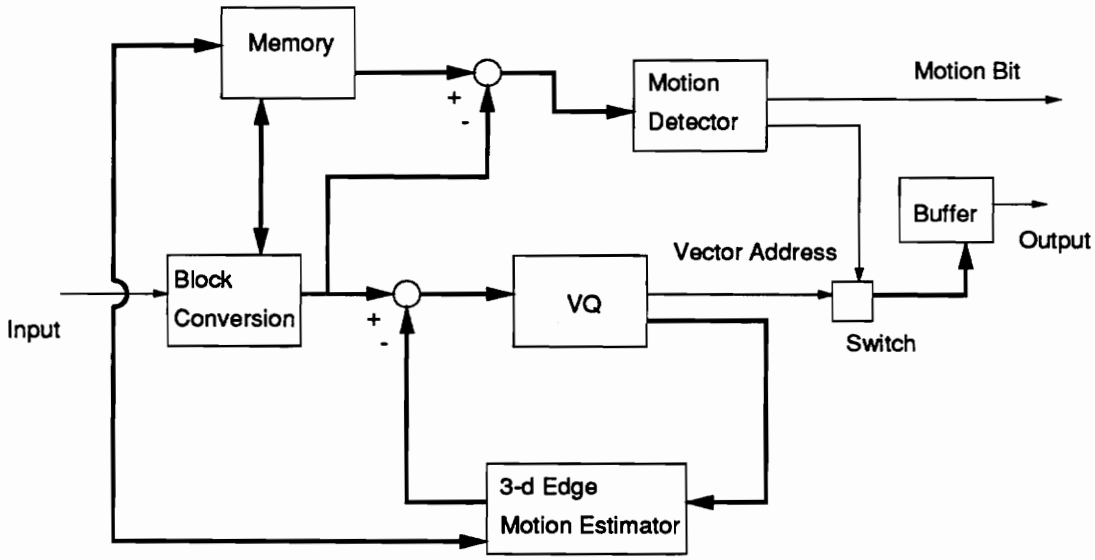
In this algorithm, the new vector, $\mathbf{v}_k$, is compared to the vectors already selected as the initial codevectors, $\mathbf{v}_i$. If the maximum of the differences between $\mathbf{v}_k$ and $\mathbf{v}_i$ is greater than a threshold, T, $\mathbf{v}_k$ is selected as the new codevector.

5.3.1.2 Vector quantization method

The first system developed is the one which vector quantizes the error blocks from the motion compensation part. This system is straightforward in its concept and was designed so that it can be used in benchmark comparisons with more advanced systems. As shown in the encoder block diagram of Fig. 17(a), the vector quantizer is in the forward loop in a prediction system. The thick arrows indicate that the signal is passed in vectors. The decoder is shown in Fig. 17(b). Note that there is no vector quantizer in the loop and the table lookup block has been added to fetch the difference vector from the codebook with the received vector address.

The prediction error values from the motion compensation scheme in the previous chapter are divided into blocks. Although there are various block shapes and sizes used in previous research on VQ, a suitable block size is 4×4. The square shape does not favor any type of image. It tends to reproduce vertical and horizontal edges with the same quality. Also the size of 4×4 is small enough to preserve detail for most images.

The code book for the VQ is obtained in three phases: initial code vector, open-loop and closed-loop code vector generation phases. The initial vector generation phase is described in detail in the previous subsection. The open-loop phase is the code book generation from the original difference signals. It means that the vector quantizer is not included in the feedback loop of Fig. 17(a). In this way, only the difference blocks between the original and the predicted block obtained with the motion compensation algorithm are used in the code book generation process. The code book thus obtained is fed back as the initial codebook to the new code book generation with the next frame. To generate the code book for a sequence of images, it is time-consuming and inefficient to generate the difference blocks for the whole sequence.



(a) encoder

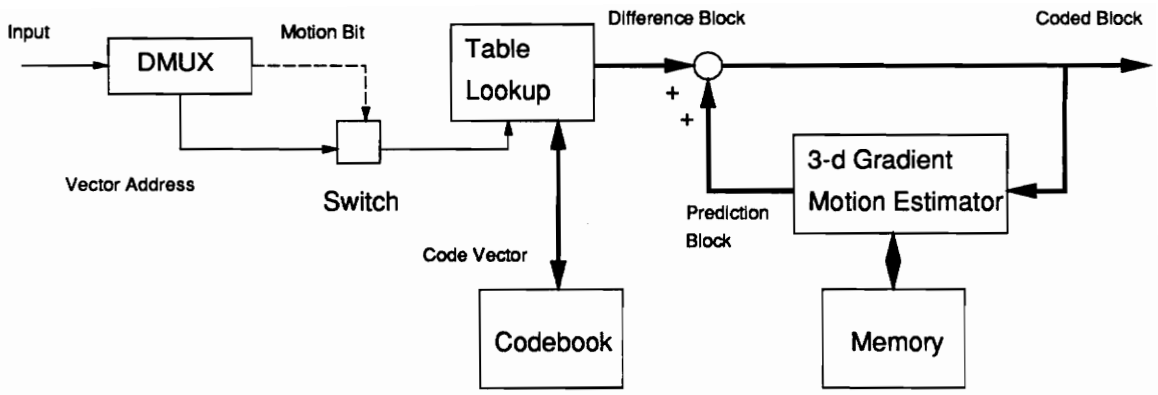
Figure 17. Block diagram of 3-d gradient motion-compensated VQ codec

The closed-loop phase is the one with the vector quantizer in the feedback loop in Fig. 17(a). The difference blocks between the original block and the predicted block is vector quantized with the previously generated code book, then added to the predicted block to reconstruct the original block. This process gives the code book the capability to cope with real-world difference blocks. In real situations, every image frame used in the prediction is corrupted with quantization noise and it is insufficient to use the code book trained with just the original difference blocks.

The code book after the closed-loop phase is used in the VQ process. It is important that the training sequence for the code book generation process be different from the testing sequence used to evaluate the vector quantizer. In real-world situations, the vector quantizer may encounter many different types of image sequences. In this dissertation, the scope is limited to videoconferencing image sequences. But even within video conferencing image sequences, there are many different types of motion and objects.

The next experiment was on the application of a mean/residual vector quantizer (Baker and Gray, 1982) to the prediction error images. The mean is removed from a block of the error image and the codebook is generated based on the mean-removed error images. The extracted mean then is scalar quantized and sent through the channel as an overhead. The statistics of the error images show that the mean of the block means is close to 0. This is due to the fact that the error image is bimodal in its nature and errors with both signs tend to coexist in a block.

The range of the error signal is larger than that of the gray values, spanning from -255 to 255 with 8-bit image sample quantization. To represent the error signal, 9 bits are needed. Normally, to reduce the overhead, the mean value is truncated to a smaller



(b) decoder

Figure 17 Block diagram of 3-d gradient motion-compensated VQ codec

number of bits. Note that the PDF of the prediction error signals tends to be Gaussian or Laplacian, usually highly peaked at the mean. Meanwhile, the statistics for the block mean of the error image should be Gaussian with the mean value of 0 if a sufficient number of means has been gathered. This allows us to use a tapered quantization scheme such as Max's (1960) ***minimum mean squared error*** (mmse) method. To obtain the statistics of the block means, such as their mean and standard deviation, the block means of the error images in open-loop mode (without VQ) are used. This is only an estimate of the statistics which occurs when VQ is used in the encoder loop.

5.3.2 Method 2: Classified vector quantization

When the straightforward VQ method is applied to the prediction error signal from the 3-d predictor, degradation of the edges is visible. The staircase effect (Ramamurthi and Gersho, 1986) which is common in a normal VQ method appears also when the prediction error signals are vector quantized. With a limited codebook size, it is impossible for the codebook to contain all possible edges. When the normal mean square error criterion is used in the selection of the codevectors from the codebook, it is likely that blocks with a high degree of activity translate to a codevector with a constant level. In a normal image coding case, this results in discontinuity of gray levels across the boundary between blocks.

In the quantization of prediction errors, noise will be introduced in a different way. It is natural that the interblock noise will not be as severe as in the case of vector quantizing image samples because of the fact that the noise which is added along the boundary will be dependent upon the recoverability of the error signal which may be a small percentage of the original sample. However, the error signal block consists of pulses

which are usually located along the edges of the original image. With a small number of codevectors, it is extremely difficult to match this type of signal and the distortion due to the fact that mismatch of the prediction error block becomes severe.

If the mismatches occur in the edge orientations or in the values on either side of the edge, the fidelity of the edges is not preserved in the coded image. It is a well-known fact that if edge integrity is maintained, the error along the edges does not degrade the subjective quality crucially (Netravali and Limb, 1980). This is due to the ***masking effect*** which allows the values around the edges be quantized coarsely. As mentioned before, the signal to be quantized here is a prediction error signal. The objective here is to preserve the edges of a prediction error image. It may seem that a different approach should be taken to code the error image. However, the most noticeable peaks in the error image arise along the edges of the original image. This is due to the fact that motion estimation does not always yield accurate edge locations. This point can be explained in detail with the following equations. If you assume that the gradient in one direction can be approximated by a difference between two consecutive sample values, the gradient is

$$\begin{aligned}\Delta e &= e(m_1, n_1) - e(m_2, n_2) \\ &= \{x(m_1, n_1) - x'(m_1, n_1)\} - \{x(m_2, n_2) - x'(m_2, n_2)\} \\ &= \{x(m_1, n_1) - x(m_2, n_2)\} - \{x'(m_1, n_1) - x'(m_2, n_2)\}\end{aligned} \tag{5.6}$$

where e is the prediction error value, Δe the difference of error values, x the original sample value, and x' the predicted value. From Eq. (5.6), the gradient in the original image can be represented as

$$\begin{aligned}\Delta x &= x(m_1, n_1) - x'(m_2, n_2) \\ &= \{x'(m_1, n_1) - x'(m_2, n_2)\} + Q(\Delta e)\end{aligned} \tag{5.7}$$

where Δx is the difference between original samples, and Q represents the quantizer. This means that the gradient of the original samples can be written as the sum of the gradient of the predicted values and the quantized gradient in the prediction error image. It is evident that if the prediction is not very accurate, the quantization of the gradient of the error image (which is the same as quantization on the error image) need to be precise.

A VQ method which retains the edges must be used. The classified VQ method (CVQ) (Ramamurthi and Gersho, 1986) can be modified to quantize the error signal block. The CVQ technique used in quantizing image samples can be applied directly to quantizing error signals. The classifier to be used in classifying image blocks can be designed similarly to the method in (Ramamurthi and Gersho, 1986). Several different classes can be defined. The ***shade*** block does not have any significant gradient. In experiments with error images, if only the moving blocks were quantized as in the previous section, there was no block belonging to this class. The ***midrange*** class has moderate gradient but no definite edges. If there are edges in the block but without any marked orientations, the class is called ***mixed***. The rest of the classes, depending upon the four major directions, are: horizontal, vertical and two diagonals. These classes are further divided depending upon the location and the polarity of the edges. The polarity is decided by whether the intensity change across the edge is from positive to negative or vice versa. Following the above argument, the blocks are classified into 31 classes.

The characteristics of the error signal are different from those of the image samples in many respects. One of the differences is the bimodality of the prediction errors. The prediction errors can have positive or negative values depending upon the prediction. An example of a prediction error image is shown in Fig. 18. Note the pulses and the walls which are the concatenation of the pulses. These sudden changes are due to incorrect motion estimation which is typical for motion estimated prediction error

images. The method Ramamurthi and Gersho (1986) used in the edge enhancement step cannot be directly applied to error images. In the procedure for obtaining the horizontal and vertical gradients, d_h and d_v, the normalization of gradients cannot be directly applied because of the bimodality of the prediction error signal. The gradient of the error signal cannot be normalized by dividing by the average values of two values with different signs. Also their argument is that the sensitivity of the eye is proportional to the normalized gradient. But such argument is irrelevant to the gradient of error images. There is no theoretically justified reason for the normalization. In experimentation, the straightforward gradients are obtained and thresholded by a value.

The generation of an initial codebook becomes more complicated than for the previous systems. The initial codebook should be classified and for each class a different set of codevectors must be chosen. Suppose we use the random codes method described in Section 5.3.1.1. Suppose we determine the size of each class in the codebook. If we have a limited number of training image frames at hand, it is quite difficult to obtain a reasonable number of initial codevectors for some classes. For some of the edge classes, their occurrence is not as frequent as for others. Aside from the fact that there are not enough widely spaced vectors, the number of vectors for the classes is too small for them to be used as the initial codevectors. These facts are illustrated in Table 2.

Table 2 is the result of the classification of the error images from the training image sequence without any quantization. The shade class represents the codevectors with no activity, and the midrange class exemplifies the codevectors with the amount of activity between the shade and edge classes. For the edge class, each class with different orientations and deviations was subdivided into two depending on whether the gray value is increasing or decreasing across the edge. Note that the number of vectors in a class ranges from 12 to 4786. Especially, for the diagonal edge classes, the number of vectors

Figure 18. Example of a prediction error image from 3-d gradient motion estimator

was too small. For classes with a small number of training vectors, it is difficult to decide on the number of initial codevectors for the class. Even if we use all the training vectors as the initial codevectors for a class with small number, it is questionable if they truly represent the codevectors.

One of the solutions to this problem is to use product codes for classes with a small number of training vectors. It was found that if there is an edge in an error block, there are strong correlations in the direction of the edge. This means that if the error block is classified as one of the strong edge classes, i.e. vertical, horizontal or diagonal, one set of levels can be assigned to the pixels on one side of the edge. For the initial codevectors, one quantization level represents either side of the edge. A problem arises when trying to assign the number of codevectors to each edge class. The number of codevectors must be chosen carefully so that the product code can be applied.

The procedure for determining initial codevectors for edge classes is as follows. First, we have to decide on the total number of codevectors, N_{cv}, in the codebook which is usually a power of 2. Then we assign a number of codevectors n_i, which is a square of an integer number, to each class of codevectors. Usually different values of n_i are allocated to two kinds of edge, i.e., diagonal or horizontal/vertical. Note that here the codevector is classified according to the direction of the edge, not considering whether the gray value is increasing or decreasing across the edge. This means that there are $\sqrt{n_i}$ quantization levels for each side of the edge. From the n_i cases, we have to subtract the number of the occurrences of the same levels, which makes the gray level equal for the whole block. There are $\sqrt{n_i}$ occurrences of this duplication of levels across the edge for one class of codevector. Hence, the total number of codevectors, p_i, for each edge class becomes

Table 1.. Number of training vectors in each class

1: shade, 2: midrange, 3: mixed edge, 4 - 9: horizontal edge, 10 - 15: vertical edge, 16 - 23: diagonal edge(45°), 24 - 31 : diagonal edge (135°) classes

Class	Number of vectors	Percentage (%)
1	0	0
2	4768	48.8
3	1342	13.7
4	367	3.7
5	343	3.5
6	239	2.4
7	416	4.2
8	426	4.3
9	258	2.6
10	161	1.6
11	179	1.8
12	104	1.0
13	150	1.5
14	194	1.9
15	111	1.1
16	56	0.5
17	78	0.7
18	64	0.6
19	14	0.1
20	48	0.4
21	80	0.8
22	62	0.6
23	19	0.1
24	42	0.4
25	55	0.5
26	39	0.3
27	12	0.1
28	32	0.3
29	59	0.6
30	42	0.4
31	15	0.1

$$p_i = n_i - \sqrt{(n_i)} \tag{5.8}$$

In Fig. 19, a simple case of two gray values on each side of the edge is demonstrated. If each class is further divided into two, depending on whether the gray value is increasing or decreasing across the edge, the total number of codevectors for each edge class becomes $p_i/2$. The total number of codevectors for the edge classes, n_e, is:

$$n_e = \frac{p_d}{2} \cdot n_d + \frac{p_{h/v}}{2} \cdot n_{h/v} \tag{5.9}$$

where p_d is the number of codevectors for each diagonal edge class, n_d the number of the diagonal classes considering the increase/decrease across the edge, $p_{h/v}$ the number of codevectors for each horizontal/vertical edge class, $n_{h/v}$ the number of the horizontal/vertical edge classes.

The total number of the codevectors, N_{cv}, can be expressed as:

$$N_{cv} = n_e + n_m \tag{5.10}$$

where n_m is the number of the codevectors for the midrange/mixed classes. Note that n_m is determined by the number of edge classes.

An example of initial codebook design will be described below. Suppose that the desired codebook size is 2^{10} (= 1024). We have to find out the number of codevectors for each edge class. One of the most annoying side-effects of vector quantization is the staircase-like appearance of diagonal edges. This leads us to allocate more vectors to the diagonal edge class. Allocate 64 (= 8^2) vectors for each diagonal edge class and allocate 36 (= 6^2) vectors for each horizontal/vertical edge class without considering decrea-

se/increase across the edge. Next, we have to delete the vectors with the same gray values on both sides of the edge. Hence 56 (= 64 - 8) vectors are assigned to the diagonal edge class and 30 (= 36 - 6) vectors are assigned to each horizontal/vertical class. There are eight diagonal edge classes and six horizontal/vertical edge classes. According to (5.9), the total number of codevectors for the edge class is 628. It follows that the remaining 396 vectors are allocated to the midrange/mixed class.

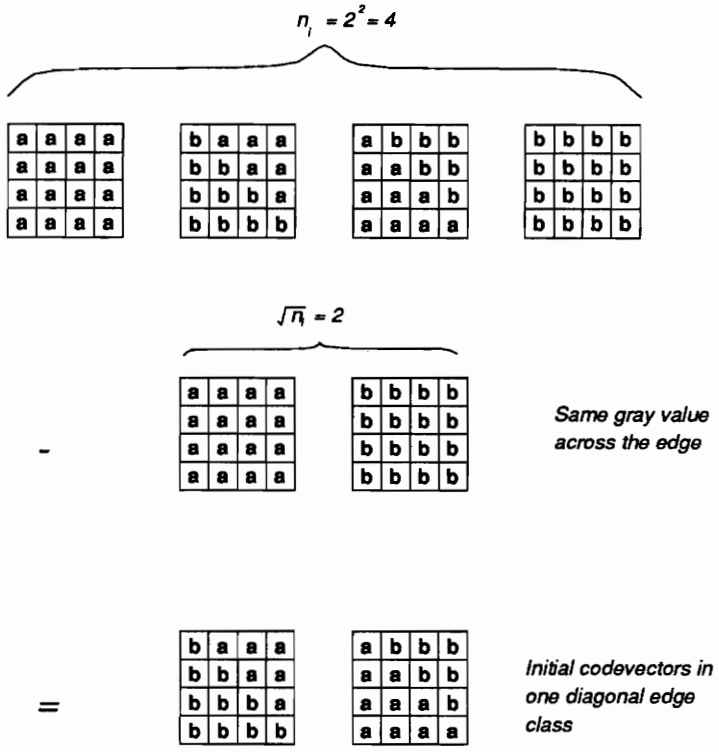


Figure 19. Example of counting codevectors for an edge class

5.4 Smoothing Filter for the Prediction Blocks

In the previous systems in this chapter, the difference block between the present block and the predicted block, found by the 3-d gradient estimator, is vector quantized. Vector Quantization (VQ) of difference signals is not a simple task. Even though VQ is one of the best quantization methods in applications to very low bit rate coding, it is very difficult for VQ to match the details of difference signals with a limited number of code vectors. If the quantization errors accumulate as more frames are coded, a gradual degradation of the image is the result. As mentioned in the previous section, what makes coding difference blocks more difficult is the pulsive component in the difference blocks as mentioned in (Nasrabadi et al., 1989). These pulsive components affect both the 3-d gradient motion estimation and the predicted values which are the coded image samples. The appearance of these components is very unpredictable and makes the code vector selection difficult. The blocks can be classified depending on the type of pulsive component but this may result in a large overhead to cover many types of such components.

These pulsive components come mainly from the inaccurate matching of moving edges. If there is a mismatch, then it is likely that two similar pulsive components will appear in different locations. It follows that it is desirable to suppress one of the components. If we suppress one of the components too much, in this case the predicted block, the benefit of VQ on the difference signal disappears. It simply becomes a normal VQ on the original image samples. Keeping this in mind, a smoothing filter can be applied to the predicted block to suppress the sharp components. Note that the predicted block means the block to which the present block corresponds to in the previous frame. In this way, also the noise which was introduced by VQ of the difference signals can be reduced.

As shown in Fig. 20, by applying the smoothing filter to the predicted block, the direct effect on the image by the filter can be avoided. Suppose the filter is used on the image after coding. The image may look better than the previous ones coded without the filter but the side effects of the filter, such as blurring of the image, can be severe. By filtering the predicted block just before subtracting from the present block, the effect of the smoothing on the coded image can be minimized. The addition of the code vectors compensates for the details which the predicted block has lost after filtering.

There are many types of filters which can be used for smoothing the predicted image as long as they have low-pass characteristics. Note that even a simple type of low pass filter can be very expensive to implement. The following technique can minimize the hardware needed in the implementation of the filter. Smoothing can be done with one of the mask components introduced in 3-d gradient operator (Lee et al., 1989). In this way, the filter can be integrated naturally into the system. The smoothing has been done already when the 3-d gradient was obtained for the motion estimation. In Fig. 5, one of the masks, G_{i2}, has been calculated for the 3-d gradient. The only thing for the new system to do is to fetch the component G_{i2} from the memory and to normalize them so that they are compatible with gray values. This simple filtering method gives more weight to the center pixel of the mask and half of the weight to the four neighborhood pixels. Refer to Fig. 7 for the coordinates of mask coefficients. The equation for the filter is

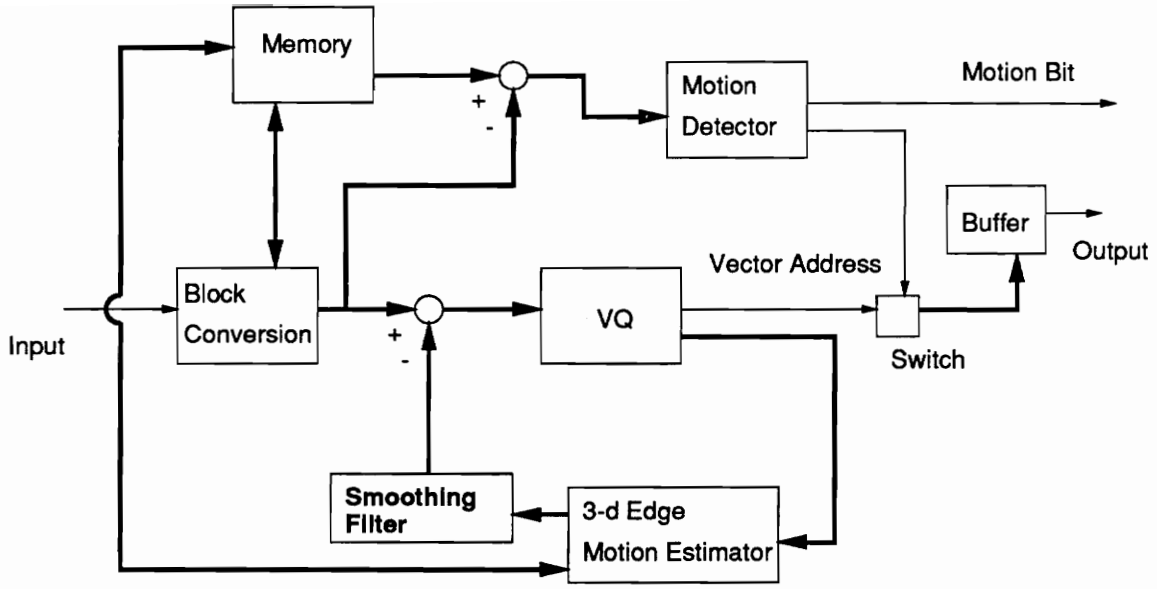


Figure 20. The block diagram of 3-d gradient motion compensated VQ encoder with prediction block filtering

$$\hat{x}(i,j) = \frac{x'(i-1,j) + x'(i,j-1) + 2x'(i,j) + x'(i,j+1) + x'(i+1,j)}{6} \tag{5.8}$$

where $\hat{x}(i,j)$ is the smoothed center pixel value and x' 's are the pixel values before smoothing.

When this scheme is applied to difference block VQ, it works similarly to mean-residual VQ which is used to remove the means from the block to concentrate on the shape of the signal and to reduce the codebook size. However, the difference is that the signal to be subtracted from the present block tends to follow the characteristics of the present block. Also the most important thing is that the overhead information doesn't have to be transmitted in this system while the mean is sent over the channel in the mean-removal VQ.

5.5 Conditional Refreshing of the Uncompensable Blocks

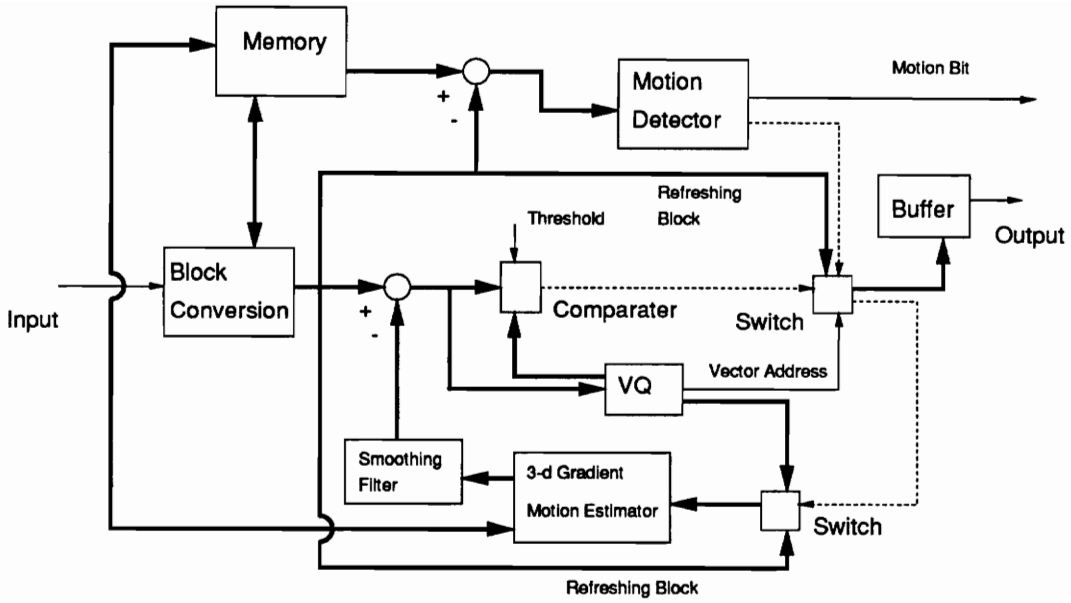
The problem with the techniques mentioned up to this point is that even if they have been proved to work well with the testing sequences at hand, there is no guarantee that their performance will remain the same forever after the testing sequence. The last technique employing the smoothing filter in the prediction stage gives some hope in restraining the coded images from much degradation, but the possibility of gradual degradation remains. Especially unexpected events such as a sudden movement or noise produced at the source can be fatal to the performance of the vector quantizer.

To eradicate the problem of gradual degradation, the technique of refreshing can be used. Refreshing can be done to either the whole frame or just a portion of it (blocks). Suppose the whole frame is refreshed with the original pixel data if the amount of noise exceeds a certain threshold. The bit rate will jump up to 8 bits/pixel at that point. This

sudden jump in the bit rate makes it impossible to be applied in practical systems. If the change in the bit rate is too large, it is very difficult to normalize the frame rate both in the transmitter and in the receiver. A large buffer may be needed to normalize the frame rate.

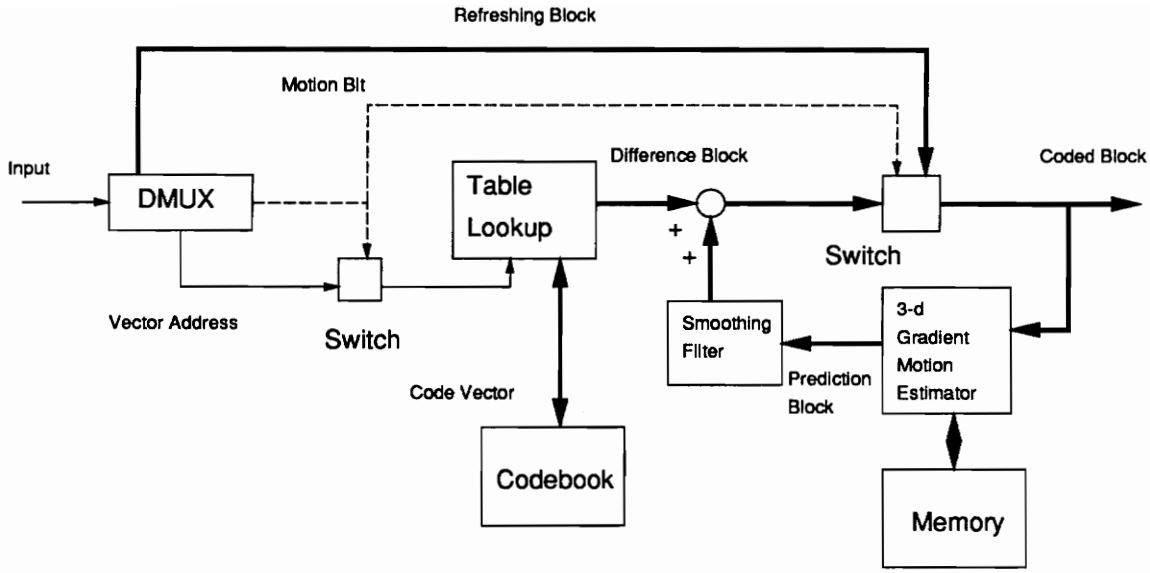
Instead of refreshing the whole frame, the blocks to be vector quantized in each frame can be refreshed with the original pixel values if the difference between the original and the vector quantized block is above a certain threshold. The refreshing can be done by sending the original pixel values or the quantized (pixel-by-pixel) pixel values with reasonable accuracy. If a quantization method is used for the refreshing, it makes the system more complex without much benefit. If the refreshing data is not accurate enough, it will only increase the frequency of refreshing because of the faster deterioration of the image quality. The system developed here uses the original pixel values for refreshing.

In Fig. 21(a), the block diagram of the encoder with conditional refreshing is shown. Also in Fig. 21(b) the decoder is shown. The system became more complex because of the addition of refreshing data. At both sides of the codec, a switching mechanism to switch between the refreshing data and vector address is added. The refreshing comes at the expense of an increase in the bit rate. This increase can be negligible in sequences with slow motion or quite high in cases of sudden motion or the appearance of new objects in the scene, etc.



(a) encoder

Figure 21. Block diagram of 3-d gradient motion-compensated VQ codec with conditional refreshing



(b) decoder

Figure 21 Block diagram of 3-d gradient motion-compensated VQ codec with conditional refreshing

The typical consequence of this replenishment is that the quality of the image will remain about the same. This means that the signal-to-noise ratio (SNR) will remain about the same. Although the subjective quality does not necessarily have to follow the SNR, the author has found experimentally that with the developed system the subjective quality follows the SNR most of the time, but not the other way.

In order to send the refreshing data to the receiver, the motion segmentation bit has to be modified to indicate the uncompensable state. Until now, the overhead consisted of the motion segmentation bit to indicate if the block was moving or not. The moving block has to be further classified into compensable and uncompensable. For the uncompensable block, the refreshing signal is sent. To indicate either compensable or uncompensable, one more bit has to be sent along with the motion bit. Whenever the motion bit is 1 (the block is moving), one more bit (0 when it is compensable and 1 when it is uncompensable) follows. Figure 22 shows a sample motion segmentation map when conditional refreshing is used. Note that the brightest spots indicate the uncompensable blocks and the darker ones show the compensable blocks which are coded with VQ. The black ones are the nonmoving blocks.

The amount of refreshing is controlled by the threshold of the difference between the original block and the reconstructed one. If this threshold is fixed, the variation of the bit rate may fluctuate widely depending on changes in the scene. In practical applications, the size of the buffer before the channel may be limited. In this case, nearly constant bit rate is desired. The threshold itself may be variable and can be controlled by a control signal from the buffer. If the buffer is nearly full, it can send a signal to increase the threshold and refresh the signal later when the buffer is ready to accept a larger volume of data. The pseudocode for the refreshing algorithm is shown below.

1. Input the difference block into the vector quantizer.

2. Find the differences between the difference block and the code vectors.

$$E(k) = \sum_{i=1}^{N} (DIFF(i) - CVEC(k))$$

where E(k) is the block difference, N the number of pixels in the block, DIFF the difference between the original block and the prediction block, and CVEC the code vectors in VQ codebook.

3. If the MINIMUM (E(K)) > T1

Send 1 after the motion bit and the refreshing block with original pixel values to the receiver

Else

Send 0 after the motion bit and the address of the code vector to the receiver

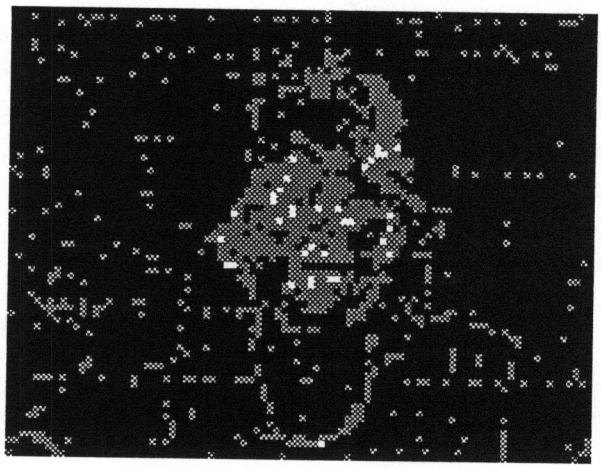


Figure 22. A sample block motion segmentation map with refreshing

Chapter 6

Experimental Results

6.1 Scalar Quantization

The performance of the developed motion estimator was compared to the pixel recursive approach developed by Netravali *et al.* (1979). The error signal from the predictor is scalar quantized. Netravali's method is one of the first in the pixel-recursive approach. It uses a sort of steepest-descent method to optimize the motion vector in a pixel-to-pixel fashion. This algorithm is fairly complex, which in turn makes it difficult to be implemented in hardware. The image sequence containing the normal video telephone pictures (Miss USA) is used in the comparison. Only the luminance components with a size of 287x360 were tested.

Visual inspection of the image frames from the two different algorithms did not show much difference in the quality when the error signal was scalar quantized. Figure 23 shows the original 20th frame of the test sequence, Fig. 24 the same frame coded with Netravali's algorithm, and Fig. 25 the frame coded with the first method of the 3-d gradient motion estimation in Section 4.2.1. The images coded with other modified algorithms didn't differ much by visual inspection conducted by the author. Also with the movie loop feature in the Perceptics image processor in the SDA laboratory, all the coded sequences were observed. The effects of the quantization noise were not noticeably different in these algorithms. The signal-to-noise ratios (SNR) of several different algorithms which are the inverses of mean square errors were compared. Although mean

square error is not always a good measure of image quality, it was adopted because it is the most widely used quality evaluation criterion and serves as an objective measurement for comparison with other methods. The equation used to find the SNR is as follows.

$$SNR = 10\log_{10}\frac{\frac{1}{M\times N}\sum_{i=1}^{M}\sum_{j=1}^{N}(x(i,j)-\hat{x}(i,j))^2}{(255)^2} \tag{6.1}$$

where $x(i,j)$ is the original sample value and $\hat{x}(i,j)$ is the coded sample value. The error variance is divided by the squared maximum gray value for the normalization. Also the data compressing capabilities were compared by the bit rates they produced. For ease of comparison, the same quantization scheme specified in Netravali's paper was applied to coding the uncompensable pixels in both methods. The signal to indicate which one of three areas (nonmoving, compensable moving and uncompensable moving) was run-length coded.

The output was quantized in the same way as Netravali's scheme (1979), for comparison. To acquire the entropy of the algorithm described here, six bits for coding uncompensable pixels and the run-length entropy of the compensability bits were considered. No further compression was done on the uncompensable data, although it can be. The entropy for the average run-lengths of three different areas (nonmoving, compensable moving and uncompensable moving areas) is obtained from the following equation:

$$H = \frac{H_n N_n + H_c N_c + H_u N_u}{r_n N_n + r_c N_c + r_u N_u} \quad bits/pixel \tag{6.2}$$

where H_n, H_c and H_u are the entropies of the nonmoving, compensable and uncompensable moving pixels, respectively. Also, r_n, r_c and r_u are the average run-lengths, and N_n, N_c and N_u are the numbers of each type. The overhead due to the uncompensable pixels must be added to Eq. (6.2). Six bits are needed to code the uncompensable error because the number of levels in the quantizer is 35. This part can be further compressed by other compression methods, such as Huffman coding. Hence the following equation was used to calculate the overall bit rate, BR.

$$BR = N + N_{conv} + 6 \cdot N_m \tag{6.3}$$

where N_{conv} is the number of bits needed to indicate which one of the two types of run is followed after a run. N_m is the total number of moving pixels.

Figure 26 gives a comparison of the bit rates. These bit rates were calculated for the 50 frames of the Miss USA sequence. In the figure, NETRAVALI means Netravali's method (1979). Method 1 means coding with searching and filling the missing vectors with pixels in the causal area. Method 2 is the same method except the missing vector was filled with average of the causal area. Method 3 is the second 3-d gradient estimation method in Section 4.2.2 with gradient plane. Method 4 is the searching method combined with the ability to select one vector in the case of multiple hits and a delayed decision for the missing vector. Finally, Method 5 is the method with gradient plane combined with delayed decision in filling the missing vector. We can observe that the bit rates of the systems with delayed decision in missing vectors are always lower than for Netravali's method, while some of the methods without the delayed decision closely match with Netravali's method.

In Fig. 27, the comparison of signal-to-noise ratio (SNR) shows that, at most of the frames, Netravali's method performs poorer than the others. The images coded by Method 3, the second estimation method, showed the best performance in the sense of SNR but with the highest bit rate. For all the algorithms developed here, the SNR's closely matched with each other on the average. We can conclude that for all the considered algorithms, the algorithms with delayed decision are superior to others including Netravali's pixel-recursive algorithm. The reason seems that the delayed decision works as a filter for the motion vectors and tends to reduce the sudden changes in motion which causes the increase in bit rate. Even without delayed decision coding which improves the performance of the system, the algorithms developed here perform very close to Netravali's method.

Here we have to note that the calculations involved in the algorithm discussed are simpler than those in the algorithm of Netravali *et al.* (1979). Netravali's algorithm involves complex recursive operations for optimization while ours involves no optimizing operations. The former has four multiplications, which need to be obtained with a fair amount of accuracy, while the algorithm here has only multiplications by 2, i.e., the left-shift operation in the accumulator register. As stated earlier, the divisions involved in this algorithm do not require high accuracy, allowing a division table to be used.

The operations in this algorithm can be run in parallel, which is not too complex to be realized in hardware. A motion vector can be calculated in parallel with the calculation of the prediction error. Also, the operations in the upper level of the resolution pyramid can be done in parallel with the lower level vector calculations. On the other hand, the calculations in pixel-recursive operations have to be done in the pixel-by-pixel basis. It means that no calculations can be done on the present pixel until the ones on the previous pixel are done.

Figure 23. Original image frame (20th frame of Miss USA sequence)

Figure 24. Image coded by Netravali's pixel recursive algorithm (20th frame of Miss USA sequence)

Figure 25. Image coded by 3-d gradient motion estimation algorithm (20th frame of Miss USA sequence)

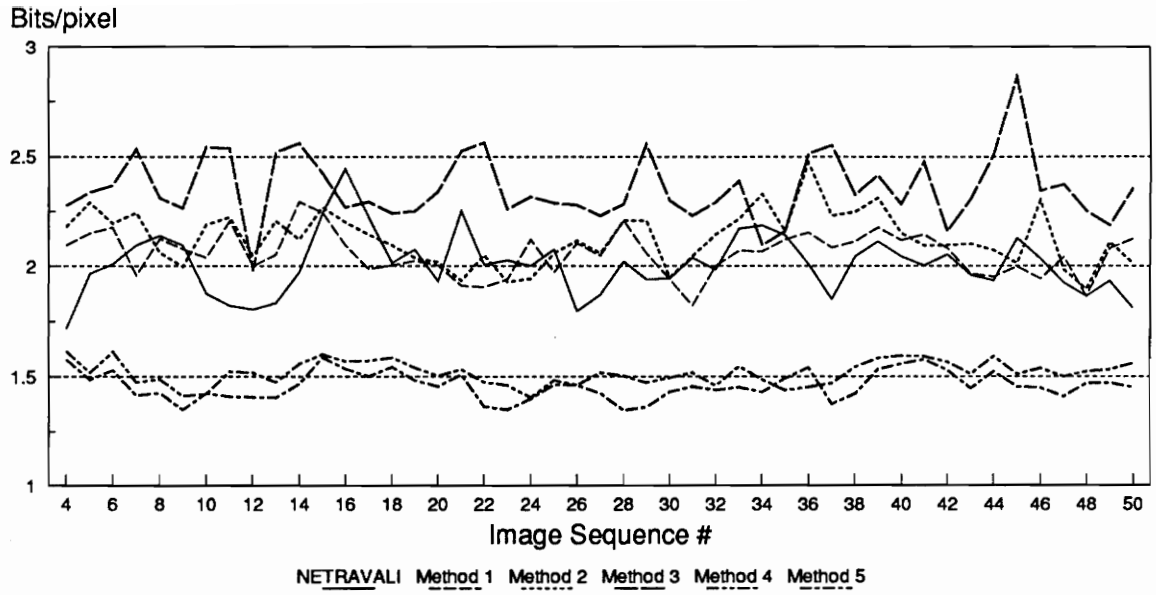


Figure 26. Comparison of bit rates

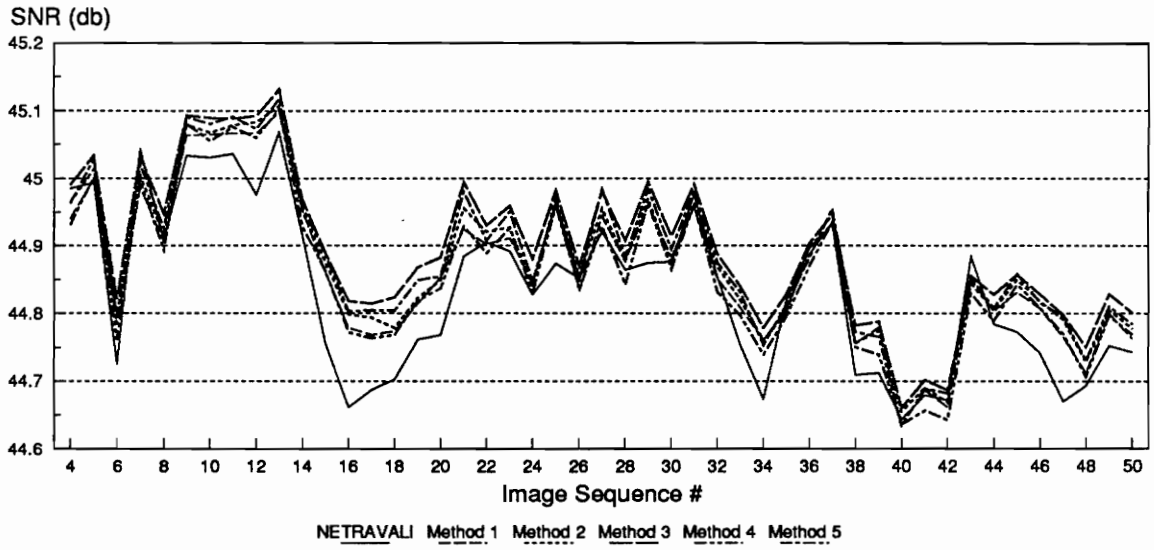


Figure 27. Comparison of signal-to-noise ratios (SNR)

6.2 Vector Quantization

The VQ methods were applied to the error images generated from the 3-d gradient motion estimator. For the training of the codebook, the CCIR image sequence was used. Figure 29 shows one frame of the training sequence. For simplicity, the 3-d gradient motion estimation technique will be referred to as GME. First, the pixel motion estimation combined with GME (GME-P), as described in Section 5.2.2, was applied to the Miss USA sequence. The initial codebook was generated following the steps of Section 5.3.1.1. The block size of each vector was fixed to be 4x4, which proved to be adequate for preserving the intelligibility of an image. The codebook size was chosen to be 256 so that the codeword length is 8. The bit rate for the image becomes 0.5 bits/pixel, assuming that no block is moving. The codebook is generated in two steps. The first step is to generate the codebook by not including the vector quantizer and generator in the coder loop. Then the second step is to include them in the loop. In the second step, the coder generates the difference signals which are more likely to be produced in actual image coding. In the second step, the codebook is trained by each frame with the codebook trained up until the previous frame as the initial codebook.

The result image coded with GME-P is shown in Fig. 30. Unpleasant noise is present everywhere in the image. The noise appears to have high frequency characteristics. The edges are not reproduced faithfully and noise is present even in the background of the subject. From now on, MCB (modified codebook) will refer to the codebook generated from the difference vectors of the training image sequence, instead of from the product quantization. Figure 32 shows the result with the GME-P method with MCB. By inspection, the result of GME-P with MCB seems to be a little bit better than the previous one, although the difference is negligible. The details in the image are more well preserved than the one without MCB. Most of the spot noise comes from the

incorrect codevectors. The vector quantizer tends to select the closest code vector to the difference vector from the codebook only in the mean-square-error sense, not taking account of the details, such as valleys and hills, in the block. These valleys and hills come from the pulsive components due to the mismatch of the edges in two frames. These noises tend to accumulate as more frames are coded, producing gradual degradation of the image quality. To demonstrate this fact, in Fig. 31, four frames from the sequence coded with GME-P at intervals of 10 frames are shown. The four frames from GME-P with MCB are shown in Fig. 33. The difference from Fig. 31 is not observable clearly, both of them showing about the same amount of gradual degradation through the sequence.

To quantize the mean value in mean/residual GME-P scheme, the mean and the standard deviation of the block means from the open-loop error image were obtained. 50 frames of the open-loop error images from the training image sequence were used in acquiring the statistics of the block means. The histogram closely resembled Gaussian distribution with the mean of -0.3786 and the standard deviation of 7.5626. The mean values were quantized into 15 levels, which requires 4 bits per sample. The reason for quantizing in 15 levels is that the already digitized level of 0 must correspond to 0 in the representation levels. The standard deviation was multiplied to the 15-level Lloyd-Max quantizer table (Jain, 1989). Figure 34 shows one frame coded with mean/residual GME-P method. Note that there is a little bit of improvement over GME-P in subjective quality for the amount of increase in the bit rate. We can observe from this that the separation of the mean is not critical in quantizing the difference images. The separation of the means alone does not produce the codevectors which matches more closely to the original vector. In Fig. 36, the result image coded with the mean/residual GME-P with MCB is shown. As in the case of GME-P, the modification of codebook has not changed

the image quality much. Also in Fig. 35 (a) through (d) show the similar degree of degradation as in GME-P. Even more noise can be observed in Fig. 37, four frames coded with GME-P with MCB, as the sequence number increases. But we have to note that more noise in the image does not necessarily relate to a decrease in SNR.

To implement the GME-P combined with classified VQ technique (GME-P-CVQ) method, an error block is classified into 31 classes. Table 2 shows the number of occurrences of each class when applied to the open-loop error images. Because only the moving blocks were considered in the classification, there were no shade blocks in the experiment. It is unlikely that a shade block will appear in a moving block most of which include edges. To obtain the threshold for the edge determination in a block, the experiment with various values of the threshold was done. The edge threshold T_e was determined so that the number of the mixed class is not large. Also the number of the midrange class shouldn't be too large. The suitable value of T_e was found to be 7.

In the experiment with GME-P-CVQ method, 10 bits were assigned to each block. Hence, a total of 1024 codevectors constitute the codebook. At first, ignoring the relationship between the levels across the edge, 64 codevectors were assigned to one of the diagonal edge classes and 36 codevectors to one of the horizontal/vertical edge classes. This allows use of the Lloyd-Max quantization table for 8 and 6 levels, respectively, for each set of classes. Then the cases for the same levels on both sides of the edge were deleted: 6 cases for H/V edges and 8 cases for diagonal edges. Hence the final distribution of the codevectors is as follows: 28 for diagonal edge class, 15 for H/V edge class, 396 for the midrange/mixed edge class. To use the Lloyd-Max quantization table, the variance of the error blocks must be calculated. The variance was 16.3814 and this value was multiplied to the table. Also the tables for the Laplacian density were used because the prediction error image closely resembles the Laplacian density. The image coded by

GME-P-CVQ is shown in Fig. 38. It shows that the edges of the image are reproduced a little bit more faithfully than others. But there is more spot noise than the images coded with the previous algorithms. Figure 39 (a) through (d) shows the four frames in sequence. Figure 39(a) shows an image with some details in edge preserved. However, the degradation through the sequence is far worse, compared to the systems considered so far. We can conclude at this time that the classification of vectors does not work well with the difference blocks between frames.

SNR's for the VQ combination systems can be calculated with Eq. (6.1) without any modification. The SNR's of the pixel motion estimation and VQ combination systems considered so far are shown in Fig. 40. Note that for all the algorithms, the SNR's decrease as more image frames are coded. This means that there is a degradation of image quality for all the systems considered here. Note that the VQ systems with MCB performs superior to others. This fact does not correspond well with the subjective comparisons. It was noted that in some cases, MCB increases the amount of spot noise, which can adversely affect the subjective quality. Figure 40 shows that the rate of decrease in SNR can be reduced with MCB. Also although GME-P-CVQ may be better in the subjective quality than others in the early part of the image sequence, improvement cannot be gained in SNR. Especially, the rate of deterioration in both objective and subjective measurement is greater than any other algorithm.

The bit rates of the sequence coded with GME-P, with or without MCB, or GME-B-CVQ can be calculated in the following way. One bit is needed for the motion segmentation to inform the decoder which block is moving. This one bit is divided by the block size, in this case 4x4, to convert into the pixel unit. Also for each moving block,

the codebook address must be sent to the decoder. It follows that the number of moving blocks is multiplied by the number of bits for VQ address and divided by the frame size. The following equation will make the above discussions clearer:

$$BR_1 = \frac{N_{mv} \cdot N_{b,vq}}{N_h \cdot N_v} + \frac{1}{N_{hb} \cdot N_{vb}} \tag{6.4}$$

where N_{mv} is the number of moving blocks, $N_{b,vq}$ the number of bits per code vector, N_h the width of a frame, N_v the height of a frame, N_{vb} the height of a block, and N_{hb} the width of a block. To calculate the bit rate for the mean/residual VQ systems, only the number of bits representing the block mean is added to the number of bits for code vector in the previous equation. The equation for getting the bit rate for mean/residual GME-P VQ systems is

$$BR_2 = \frac{N_{mv} \cdot (N_{b,vq} + N_{b,mean})}{N_h \cdot N_v} + \frac{1}{N_{hb} \cdot N_{vb}} \tag{6.5}$$

Note that only the term $N_{b,mean}$, the number of bits for block mean, has been added to Eq. (6.4).

The bit rates for the pixel motion estimation and VQ combination systems are compared in Fig. 41. It can be clearly seen that the bit rates of all the systems increase with image sequence number. It is almost certain that the bit rate will increase further in the later frames. This is caused by the continuously decreasing number of nonmoving blocks as more frames are coded because of the inaccurate coding, thus increasing the number of moving blocks which have to be coded with VQ. The difference between the bit rate of the mean/residual GME-P method and that of GME-P seems to be minor in the early part of the sequence but it increases toward the later part of the sequence. As

expected, modifying the initial codebook (MCB) does not change the bit rate noticeably. It can be concluded that introducing MCB has a positive effect on the GME-P system in the mean-square-error sense. Note that GME-P-CVQ has the highest rates, not improving the objective measurement, SNR, much.

Figure 28. One frame of training sequence (10th frame of CCIR sequence)

Figure 29. Image coded by GME-P (20th frame of Miss USA sequence)

(a) 10th frame

(b) 20th frame

(c) 30th frame

(d) 40th frame

Figure 30. Four frames coded by GME-P

Figure 31. Image coded by GME-P, initial code vectors from the training sequence (20th frame of Miss USA sequence)

(a) 10th frame

(b) 20th frame

(c) 30th frame

(d) 40th frame

Figure 32. Four frames coded by GME-P, initial code vectors from the training sequence

Figure 33. Image coded by mean/residual GME-P (20th frame of Miss USA sequence)

(a) 10th frame

(b) 20th frame

(c) 30th frame

(d) 40th frame

Figure 34. Four frames coded by mean/residual GME-P

Figure 35. Image coded by mean/residual GME-P, initial code vector from the training sequence (20th frame of Miss USA sequence)

(a) 10th frame

(b) 20th frame

(c) 30th frame

(d) 40th frame

Figure 36. Four frames coded by mean/residual GME-P, initial code vector from the training sequence

Figure 37. Image coded by GME-P-CVQ (20th frame of Miss USA sequence)

(a) 10th frame

(b) 20th frame

(c) 30th frame

(d) 40th frame

Figure 38. Four frames coded by GME-P-CVQ

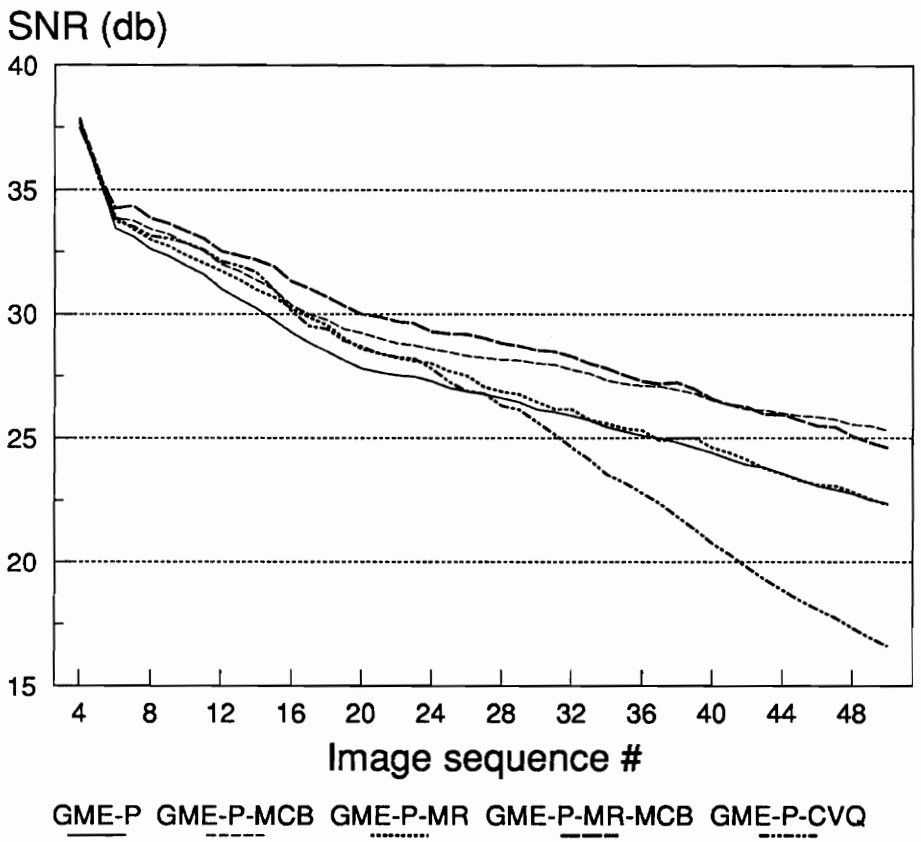


Figure 39. Comparison of SNR's in VQ systems

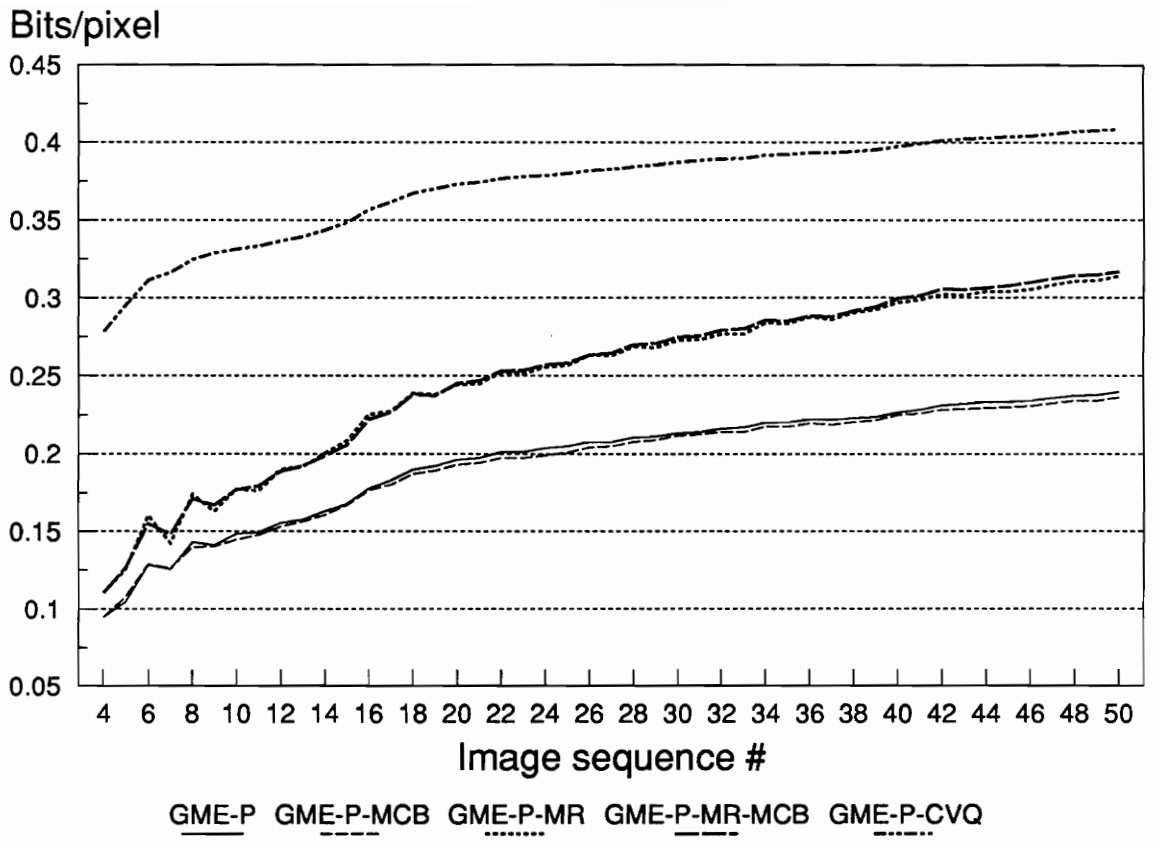


Figure 40. Comparison of bit rates in VQ systems

The block motion estimation have been applied to the 3-d motion estimation technique. From now on, this combination will be called GME-B. At this point, 3-d stochastic gradient motion estimation (SGME) is applied to the block motion estimation combined with VQ (SGME-B) for the comparisons with GME-B. The coefficients for the stochastic gradient are found for the 3x3x3 mask cube. The masks of the previous method (Lee and Nadler, 1989) smooth the image assuming that the original image has some noise component in it. To make the comparisons of the previous and the stochastic methods, the initial smoothing level of the stochastic method was set to be equal to the one of the previous method. The initial value of SNR was set to 29. As in Table 1, the SNR value ranges from 29 downward to 5, where it is assumed that the quality of the image is below the acceptance level.

Refer to Fig. 41. The SNR values of image sequence coded with GME-P and GME-B methods were compared. As can be seen from the figure, SNR value of GME-P is less than that of GME-B for all the frames in the sequence, showing a faster degradation of frames. This result shows that GME-P is estimating the motion less accurately. Also in Fig. 42 the bit rate of GME-P is considerably higher than that of GME-B, steadily increasing to 0.25 bits/pixel after 50 frames. It was concluded that the pixel motion estimation can be dropped from further consideration. Next, the performance of SGME combined with block motion estimation (SGME-B) was compared with GME-B. The comparison of SNR's and bit rates of both systems indicated that SGME-B performs better than GME-B for the whole sequence, although the difference is very small. The valleys present in SNR, sudden degradations in image quality, are smoothed by SGME method. In the bit rate comparisons, the rate of increase can be decreased by a nominal, but not negligible, amount.

The next performance evaluations were conducted on the systems using the prediction smoothing algorithm (PS). Figure 41 shows the comparisons of SNR's and note that PS has nearly eliminated the image degradation problem. Although there has been a degradation in the first part of the sequence, the value of SNR seems to have been stabilized to about 37 dB. The difference between the systems with and without PS is about 7 dB at 50th frame. The performance of PS is good also in terms of the bit rate as shown in Fig. 42. The bit rate fluctuates around 0.15 bits/pixel in the last part of 50 frames. It is increasing in the last part of the sequence but the last part of the sequence involves a large motion which is detrimental to any coding algorithm. The SGME method has improved the performance of PS technique in both SNR and the bit rate only slightly.

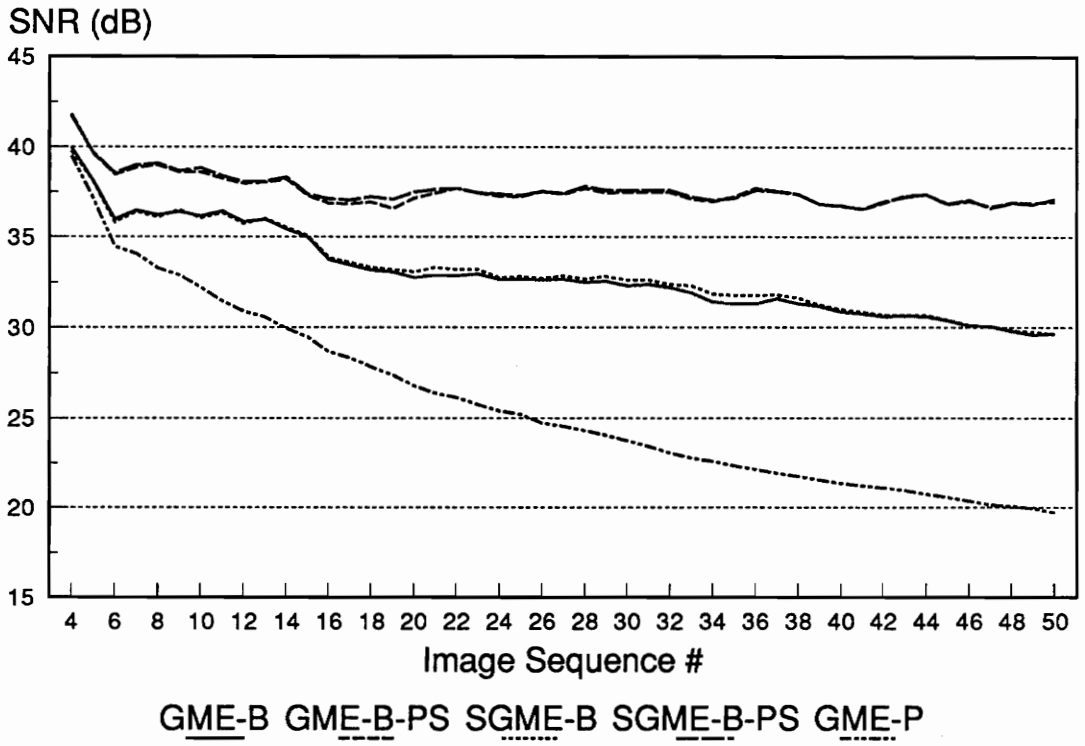


Figure 41. Comparison of the pixel and block motion estimation methods in SNR

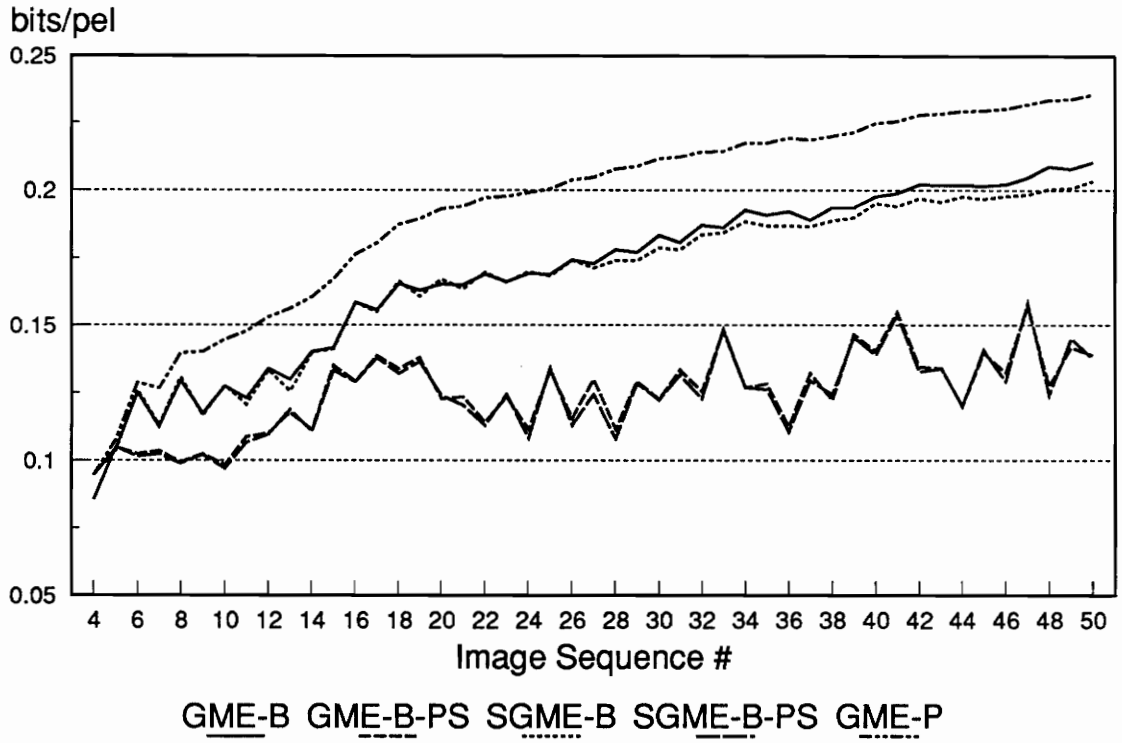


Figure 42. Comparison of the pixel and block motion estimation methods in the bit rates

The comparison of the SNR's including the conditional refreshing technique as mentioned in Section (5.5) is shown in Fig. 43. This figure shows how SNR can be stabilized through refreshing. Even with a large amount of motion between frames, SNR is nearly constant. Note that the valleys present in SNR for GME-B-PS are almost eliminated in SNR for GME-B-PS with refreshing. It shows that whenever the motion estimation does not perform accurately enough, the refreshing takes over.

For the system, GME-B-PS with refreshing, the bit rate calculation becomes a little bit more complex than the previous ones. If the motion segmentator determines that a block is moving, it is further decided if the block is to be refreshed with the refreshing algorithm in the last chapter. Suppose the number of the moving blocks is N_{mv}. Then (6.6) is satisfied.

$$N_{mv} = N_{mv,vq} + N_{mv,ref} \tag{6.6}$$

where $N_{mv,vq}$ is the number of moving blocks to be vector quantized, $N_{mv,ref}$ the number of moving blocks to be refreshed. The number of bits for the blocks to be vector quantized is the number of bits to represent a code word, $N_{b,vq}$, multiplied by $N_{mv,vq}$. The number of bits for blocks to be refreshed is the product of the number of bits for the refreshing data $N_{b,ref}$ and the size of the block $N_{vb} \cdot N_{hb}$, where N_{vb} is the height and N_{hb} the width of the block.

Next, the number of the motion segmentation bits has to be calculated. For the nonmoving part, only one bit is needed so the number of bits to be sent is N_{nm}. The moving block needs two bits per block. The reason is that in addition to indicate it is moving, the transmitter also needs to notify the receiver whether the moving block is vector quantized or refreshed. The first bit of these two bits is to indicate it is moving

and the second bit to indicate if it is to be refreshed or not. For example, if a block is determined to be not moving, 0 is sent to the receiver as a motion segmentation bit. If it is determined to be moving and vector quantized, then 10 is sent. Otherwise, if it is moving and to be refreshed, 11 is sent. Therefore, the sum of $N_{mv,vq}$ and $N_{mv,ref}$ is multiplied by 2. The sum of all the numbers considered so far is divided by the number of pixels in a frame, $N_h \cdot N_v$, to obtain the bit rate in terms of bits/pixel. Refer to Fig. 44 for the structure of the data for each type of block.

The equation for the bit rate can be expressed as

$$BR_3 = \frac{N_{mv,vq} \cdot N_{b,vq} + N_{mv,ref} \cdot (N_{b,ref} \cdot N_{vb} \cdot N_{hb}) + N_{nm} + 2 \cdot (N_{mv,vq} + N_{mv,ref})}{N_h \cdot N_v} \qquad (6.7)$$

The bit rate comparison shown in Fig. 45 shows how the bit rate varies, depending upon the amount of refreshing. Obviously, the overall bit rate has been increased by about 0.05 bits/pixel and there are more fluctuations with refreshing. But with a slight increase in the bit rate, virtually no degradation in coded image sequence can be obtained. This is further proved with coding 50 more frames of Miss USA images.

The SNR for the 100 frames coded with GME-B-PS (refreshing) is shown in Fig. 46 and the corresponding bit rate is shown in Fig. 47. As expected, SNR and the bit rate fluctuate, depending upon the amount of motion between frames. The conditional refreshing on the blocks rather than on the frames keeps the bit rate from surging too abruptly. SNR and the bit rate stabilizes quickly after the portion where a large motion is involved from about 60th to 88th frame. The large motion comes from the subject in the image sequence moving her head sideways.

The last 50 frames of Miss USA sequence involves a larger amount of motion compared to the first 50 frames. It was observed that the subject is moving her head sideways in the last 50 frames. The bit rate shown in Fig. 47 shows a substantial increase in this period. One might ask a question that what would happen if the first part of the image sequence consists of a large amount of motion. To test this, the order of the test sequence was reversed and GME-B-PS (refreshing) algorithm was applied. The SNR in Fig. 48 shows that there is a sudden drop in the first few frames and stabilize to approximately 36 dB in the last part. As shown Fig. 49, the bit rate increases up to 0.22 bits/pixel and fluctuates around 0.18 bits/pixel in the last part. These results demonstrates that if the first part of the sequence involves a large amount of motion, it takes some time for the motion estimator to perform as well as the opposite case.

The last method, GME-B-PS with refreshing, was tested on a sign-language sequence, called Telesign, which was digitized at Image Processing Technologies Inc., Blacksburg, Virginia. This experiment is to prove that this algorithm can perform well with an image sequence involving a lot of motion for a sufficient period of time. In Fig. 50, SNR is shown for the 200 frames. This sequence involves a very busy motion of hands, inevitable in sign language, which makes it extremely difficult to code efficiently. Although SNR has fallen from that of the previous sequence by about 1 to 2 dB, the last method is performing well through the whole sequence without any sign of degradation at the end. The bit rate for Telesign sequence is shown in Fig. 51. The fluctuation in the bit rate is noticeable compared to the previous sequence. The bit rate can go up as high as about 0.7 bits/pel and down to about 0.2 bits/pel. This fluctuation comes from the sign language, which can cause a very abrupt motion between frames.

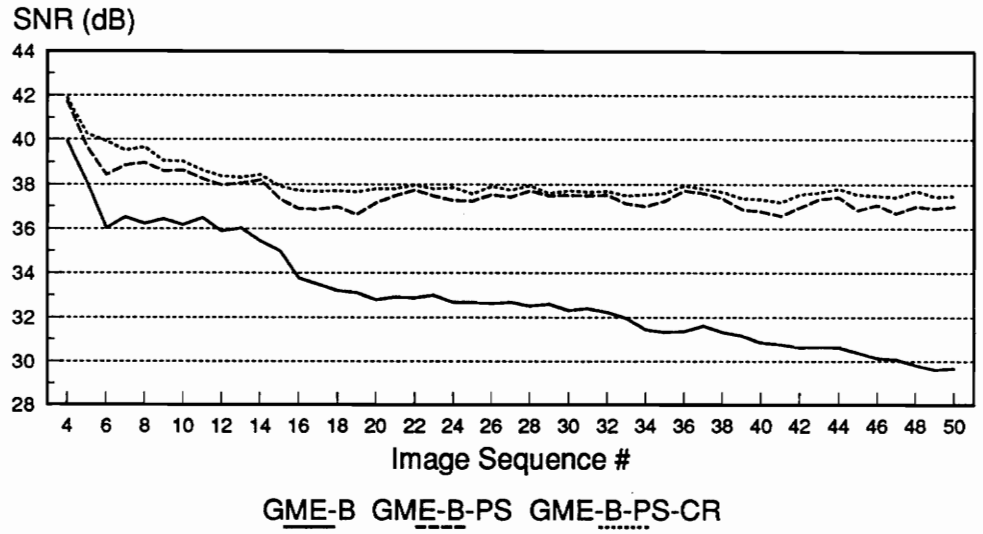


Figure 43. Comparison of SNR's with and without conditional refreshing

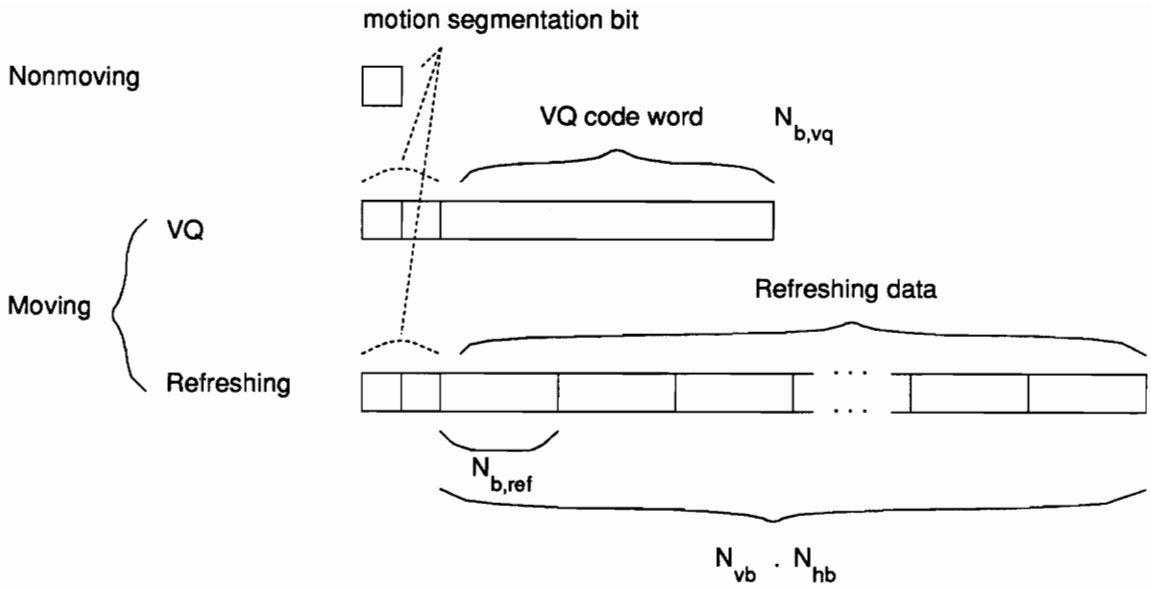


Figure 44. Structure of the data for GME-B-PS with refreshing

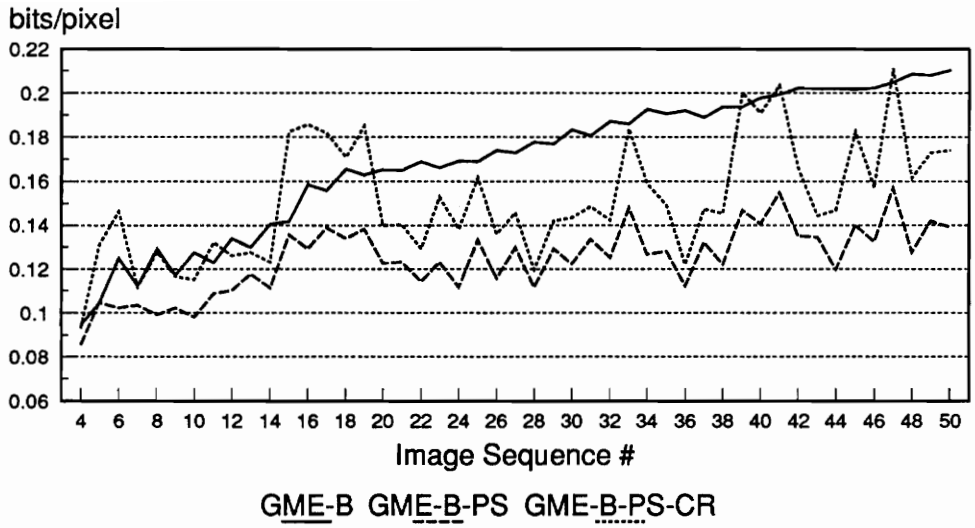


Figure 45. Comparison of bit rates with and without conditional refreshing

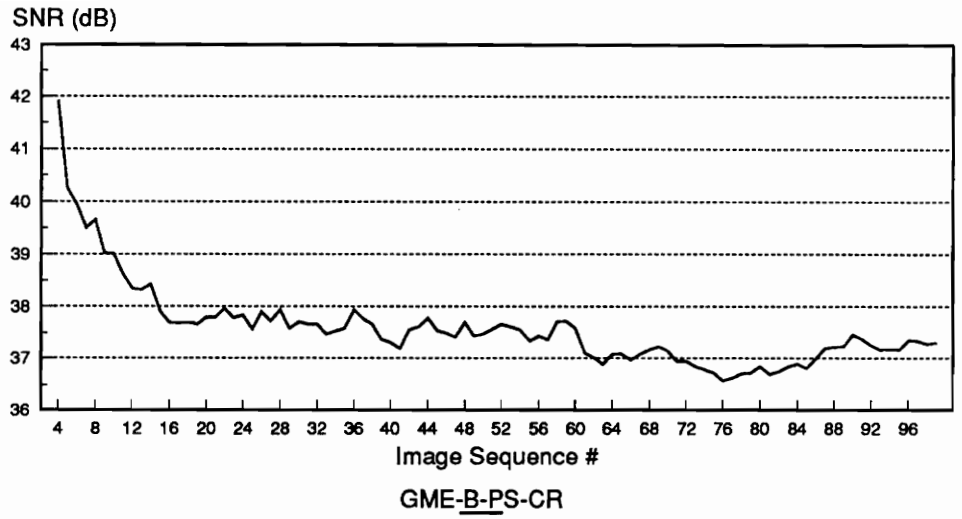


Figure 46. SNR for 100 frames coded with GME-B-PS, refreshing

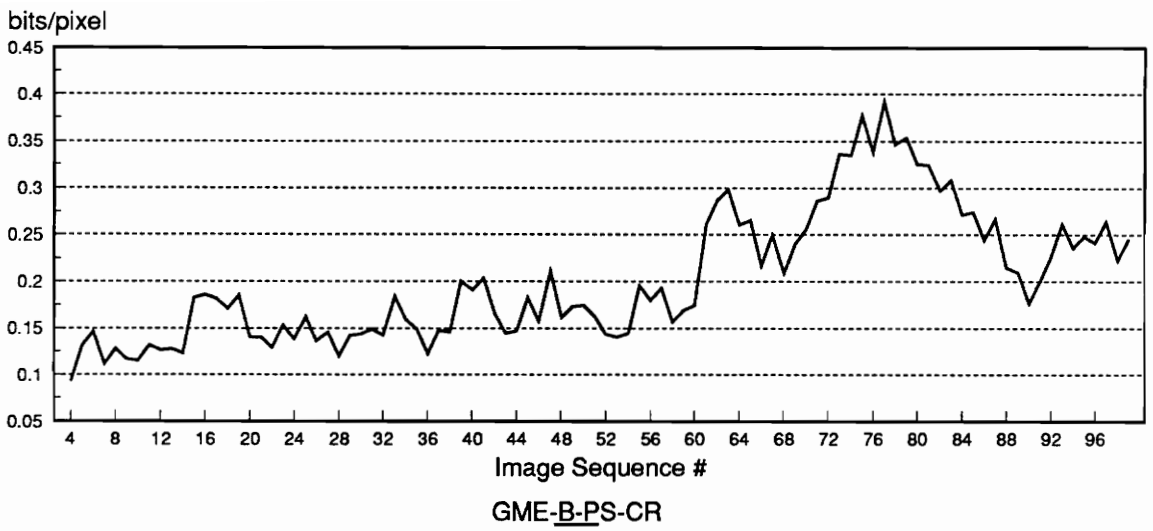


Figure 47. Bit rate for 100 frames coded with GME-B-PS, refreshing

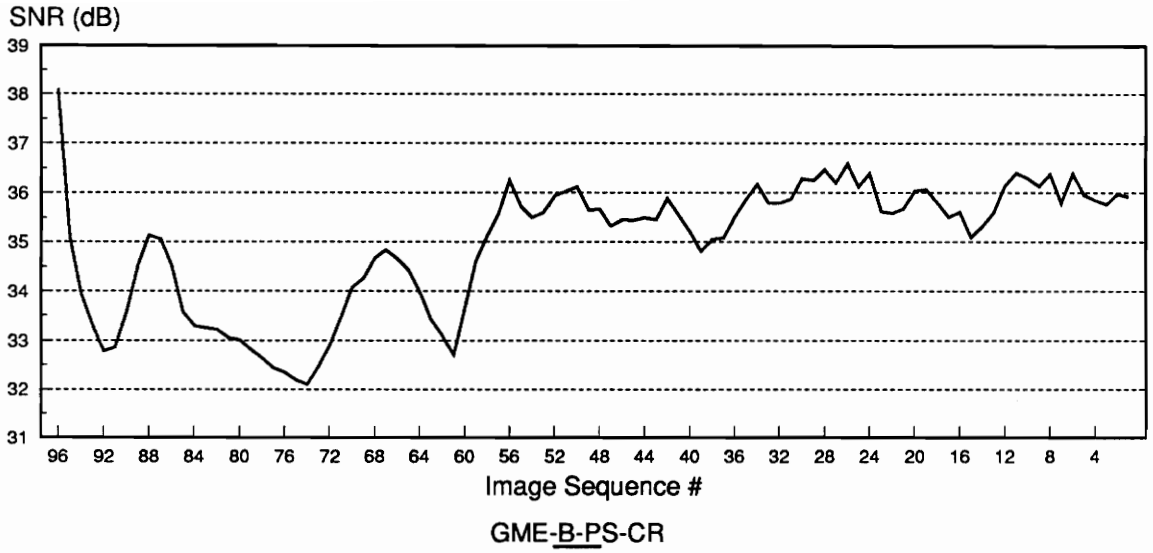


Figure 48. SNR for reversed 100 frames coded with GME-B-PS, refreshing

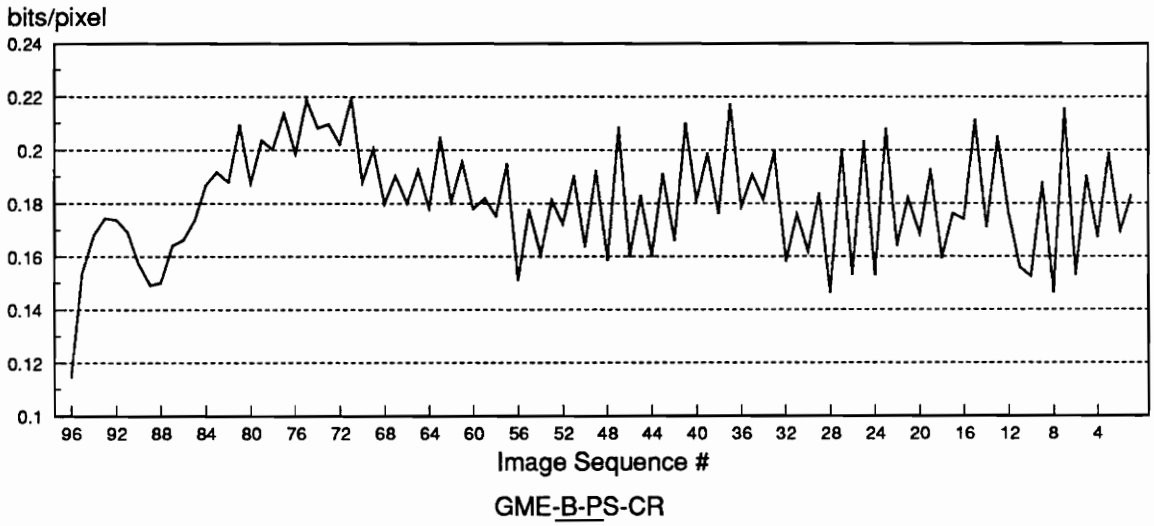


Figure 49. Bit rate for reversed 100 frames coded with GME-B-PS, refreshing

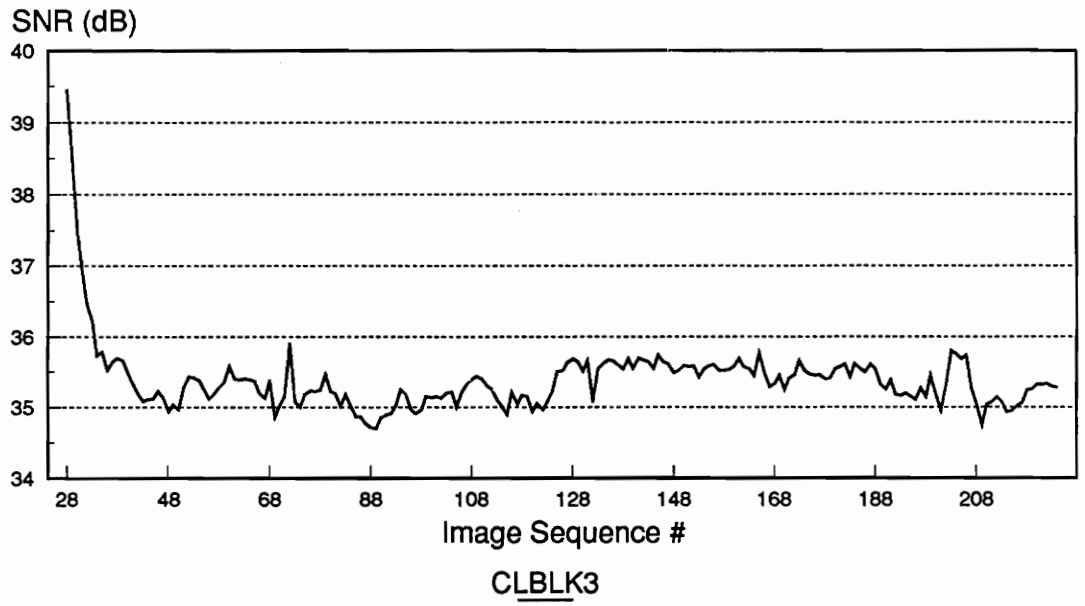


Figure 50. SNR for Telesign sequence (200 frames) coded with GME-B-PS, refreshing

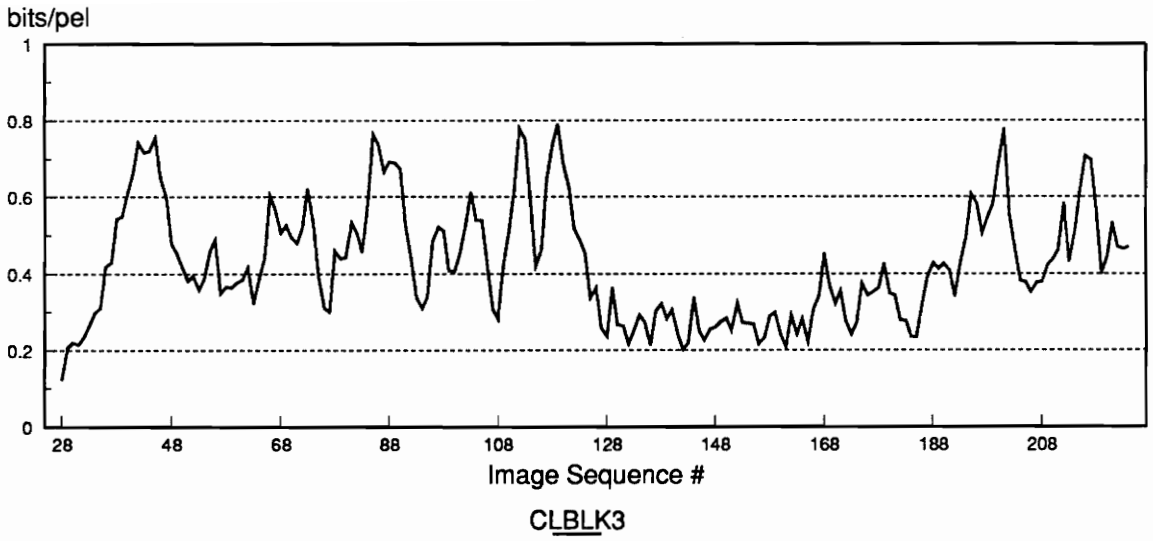


Figure 51. Bit rate for Telesign sequence (200 frames) coded with GME-B-PS, refreshing

The following discussion will show the result images coded with the block motion estimation method (GME-B). The frames coded with GME-P showed a gradual degradation of image quality, which also appeared as a steady increase in the amount of noise. Better image quality can be observed from Fig. 52 which is a picture coded with GME-B. Although the spot noise is still visible, it is not affecting the subjective quality as it did in GME-P. Also refer to Fig. 53 (a) through (d) for the sequence of four frames with an interval of 10 frames. The degradation is not so severe as in the cases of previous systems, although it still exists. This shows that the block motion estimation works better than the pixel motion estimation in the vector quantization. The mismatch of the code vector doesn't affect the quality of the image as much as in the case of the pixel motion estimation. In the pixel motion estimation, the difference block shows much higher activity than in the block motion estimation. The reason is that the pixel motion errors can cause sharp isolated peaks in the difference block.

The image in Fig. 54 is coded by SGME-B, which delivers about the same quality as Fig. 52. However, there is less spot noise in Fig. 54. This makes a more faithful reconstruction of the details in the image: the eyes and the mouth. Also four frames coded with SGME-B are shown in Fig. 55 (a) through (d). The degree of degradation is small and it is about the same compared with the images coded with GME-B. We have seen that the SNR for SGME-B is a little bit higher than the one for GME-B. To find out what makes this small difference in the SNR comparison, the images coded with the two techniques were closely inspected. With the authors' observation with a high quality monitor, there were fewer false edges than Fig. 52 which is coded with GME-B. On the whole, SGME-B performs slightly better than GME-B. The question may arise whether this slight improvement justifies the preference of SGME-B over GME-B. GME-B has a big advantage over SME-B in its simpler implementation. At this point, GME-B seems

more feasible for the practical applications.

Next, the prediction smoothing (PS) filter was applied to the prediction blocks. The image in Fig. 56 is coded in combination with GME-B (GME-B-PS) and the one in Fig. 58 in combination with SGME-B (SGME-B-PS). The two images do not show any noticeable difference. The improvement of the images due to this filtering are significant. The high frequency noise emanating from the mismatch of the VQ code vectors was nearly eliminated with the filter. The peaks in the difference block observable with the previous systems were smoothed. As one of the side effects, there are some influences from the smoothing filter, such as the smearing of the image texture, the slight blurring of edges. The most important point is that the quality of the images coded later does not degrade as much as the ones coded with previous methods. The filtering has nearly stopped the degradation in the coded image sequence. The quality of two pictures are again about equal as shown in the SNR comparisons. The two sets of four frames coded with GME-B-PS and SGME-B-PS are shown in Fig. 57 and Fig. 59 respectively. The degradation which was present in the previous systems has nearly vanished. With the 50 frames of test sequence, the degradation seems to have been disappeared completely. Again, the difference in the degree of degradation between GME-B-PS and SGME-B-PS is meager. SGME-B-PS works a little bit better in reducing the spot noise.

When the conditional refreshing is applied to the system of GME-B-PS, the coded image looks like Fig. 60. As expected, the image has much better subjective quality than the previously coded ones. The threshold for refreshing seems to be sufficiently low so that the good image quality is kept. The noise still present in the picture in GME-B-PS due to the mismatch of VQ code vectors is reduced in Fig. 60. Also we can see that the same image quality is preserved in Fig. 61 (a) through (d). There seems to be no degradation in subjective image quality whatsoever. A similar result is obtained with the

Telesign sequence. One original frame (50th) of Telesign sequence is shown in Fig. 62. Also one frame of result coded with GME-B-PS, refreshing is shown in Fig. 63. Because of the busy motions of hands, the center of the subject is not reproduced as well as the other less busy areas. To ensure there is no degradation through the sequence, four frames in intervals of 40 frames are shown in Fig. 64. On the whole, the image has not degraded much, taking into account of the fact that the motion has been increased tremendously compared to the previous test sequence.

Figure 52. Image coded with GME-B

(a) 10th frame

(b) 20th frame

(c) 30th frame

(d) 40th frame

Figure 53. Four frames coded by GME-B

Figure 54. Image coded with SGME-B

(a) 10th frame

(b) 20th frame

(c) 30th frame

(d) 40th frame

Figure 55. Four frames coded by SGME-B

Figure 56. Image coded with GME-B-PS

(a) 10th frame

(b) 20th frame

(c) 30th frame

(d) 40th frame

Figure 57. Four frames coded by GME-B-PS

Figure 58. Image coded with SGME-B-PS

(a) 10th frame

(b) 20th frame

(c) 30th frame

(d) 40th frame

Figure 59. Four frames coded by SGME-B-PS

Figure 60. Image coded with GME-B-PS with refreshing

(a) 10th frame

(b) 20th frame

(c) 30th frame

(d) 40th frame

Figure 61. Four frames coded by GME-B-PS with refreshing

Figure 62. Original frame of Telesign sequence (20th frame)

Figure 63. Telesign image coded with GME-B-PS with refreshing (20th frame)

(a) 40th frame

(b) 80th frame

(c) 120th frame

(d) 160th frame

Figure 64. Four frames coded by GME-B-PS with refreshing (Telesign sequence)

Chapter 7

Conclusions

A number of new motion estimation techniques have been developed using the 3-dimensional gradient operator for the usage in image sequence coding systems. The 3-d gradient motion estimation technique was combined with both scalar quantization for high bit rate coding and vector quantization for low bit rate coding. Their performances in coding image sequence have been demonstrated in subjective comparisons and signal-to-noise ratio (SNR) vs. bit rate comparisons.

The following are the contributions made by this dissertation in image sequence coding research.

1. New motion estimation algorithms using the 3-dimensional gradient operator have been developed. When these techniques were tested on the scalar quantization, they yielded similar performances, both in SNR and bit rate, with the pixel-recursive motion estimation, without using complex calculations. The methods are more suitable for hardware realization in VLSI because of the replicable nature of the modules in the algorithm.

2. A stochastic 3-d gradient motion estimator was developed. This technique is more advanced than the 3-d gradient motion estimation in that the coefficients in the gradient operator varies according to the noise content in the image. The performance of this method combined with vector quantization was tested and it performed better than the simplified 3-d gradient estimation method in both the objective and subjective comparisons. However, the improvement in performance is minor while this system is more complex than the previous method. In the further investigations with

vector quantization, this method was dropped and the 3-d gradient estimation was used. The performance of this method can be enhanced with further research in the future.

3. A new method for obtaining block motion was developed. The block motion was estimated through averaging the pixel motion vectors projected to the block to be coded. This method does not need any overhead for the motion vector since the block motion can be estimated also within the receiver. The performance of the system employing this technique proved to be superior to one without it. The strong point of this block motion estimation is parallel operation with the 3-d gradient pixel motion estimator, which reduces the time to estimate the block motion during the intervals between block quantizations.

4. A new method for smoothing the prediction block was developed. The filtering is not applied to the result block going out of the system loop. Instead, it is applied to the output from the 3-d gradient motion estimator so that the side effects of the filtering, such as blurring of the edges and textures, can be minimized. This method minimizes the complexity by using the same weights for the smoothing filter as is used in the 3-d gradient motion estimator. This method has nearly eliminated the gradual degradation of image frames encountered in the systems where the difference blocks are vector quantized. The bit rate fluctuated around 0.15 bits/pixel with SNR of about 37 dB in the last part of 50 frames.

5. A new method of refreshing with the original pixel values was developed to ensure the same quality of image, as measured by SNR, when coding an unlimited number of frames. This scheme compares the energy in the difference between the difference block and the code vector in vector quantizer and sends a refreshing block if

this difference is larger than a predetermined threshold. With the Miss USA sequence, the overall bit rate has been increased by about 0.03 bits/pixel compared to the method without refreshing and the degradation was suppressed, showing nearly constant SNR at 37 dB. This technique was verified to perform well with 100 frames from the Miss USA sequence and 200 frames from the Telesign sequence.

Chapter 8

Bibliography

Ahmed, N., Natarajan, T. and Rao, K. R., "Discrete Cosine Transform," *IEEE Transactions on Computers*, vol., pp. 90-93, 1974.

Bage, M. J., "Interframe predictive coding of images using hybrid vector quantization," *IEEE Trans. Commun.*, vol. COM-34, pp. 411-415, Apr. 1986.

Baker, R. L. and Gray, R. M., "Image compression using non-adaptive spatial vector quantization," *Conf. Rec. Sixteenth Asilomar Conf. Circuits, Syst., Comput.*, pp. 55-61, Oct. 1982.

Bergman, H. C., "Displacement estimation based on the correlation of image segments," *IEEE Proc. Int. Conf. on Electronic Image Processing*, pp. 215-219, Jul 1982.

Cafforio, C. and Rocca, F., "Methods for Measuring Small Displacements of Television Images," *IEEE Transactions on Information Theory*, vol. IT-22, pp. 573-579, 1976.

Cafforio, C. and Rocca, F., "The Differential Method for Image Motion Estimation," *Image Sequence Processing and Dynamic Scene Analysis*, vol. F2, pp. 105-124, 1983.

Chen, W. and Hein, D., "Recursive temporal filtering and frame rate reduction for image coding," *IEEE Journal on Selected Areas in Commun.*, vol. SAC-5, no. 7, pp. 1155-1165, Aug. 1987.

Connor, D. J., Pease, R. F. W., Scholes, W. G., "Television coding using two-dimensional spatial prediction," *Bell System Tech. J.*, vol. 50, pp. 1049-1061, Mar. 1971.

Connor, D. J. and Limb, J. O., "Properties of frame-difference signals generated by moving images," *IEEE Trans. Commun.*, vol. COM-22, pp. 1564-1575, Oct. 1974.

Dubois, E., Prasada, B. and Sabri, M. S., "Image Sequence Coding" in *Image Sequence Analysis*, New York: Springer-Verlag, 1981.

Dukhovich, I. J. and O'Neal, J. B., "A Three-Dimensional Spatial Non-Linear Predictor for Television," *IEEE Transactions on Communications*, vol. COM-26, pp. 578-583, 1978.

Dukhovich, I. J. and O'Neal, J. B., "A Three-Dimensional Spatial Non-Linear Prediction for Television," *EASCON-77 Record*, vol., pp. 17.5.a-i, 1977.

Frei, W. and Chen, C., "Fast Boundary Detection: A Generalization and a New Algorithm," *IEEE Trans. Comput.*, vol. 26, pp. 988-998, 1977.

Furner, R. R., Christiansen, R. W. and Chabries, D. M., "Motion Compensated Vector Quantization," *Proc. IEEE ICASSP*, pp. 989-992, 1986.

Gersho, A. and Ramamurthi, B., "Image Coding Using Vector Quantization," *Proc. IEEE ICASSP*, vol. 1, pp. 428 - 431, 1982.

Graham, R. E., "Predictive Quantizing of Television Signals," *IRE WESCON Conv. Rec.*, vol. 2, pt. 4, pp. 147-157

Gray, R. M., "Vector Quantization," *IEEE ASSP Magazine*, pp. 4-29, Apr. 1984.

Habibi, A., "Comparison of Nth-Order DPCM Encoder with Linear Transformations and Block Quantization Techniques," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 948-956, Dec 1971.

Heuckel, M. F., "A Local Operator Which Recognizes Edges and Lines," *J. Ass. Comput. Mach.*, vol. 20, pp. 634-647, 1973.

Heuckel, M. F., "An Operator Which Locates Edges in Digitized Pictures," *J. Ass. Comput. Mach.*, vol. 18, pp. 113-125, 1971.

Hildreth, E. C., "Computations Underlying the Measurement of Visual Motion," *Artificial Intelligence 23*, pp. 309-354, 1984.

Horn, B. K. P. and Schunk, B. G., "Determining Optical Flow," *Artificial Intelligence 17*, pp. 185-203, 1981.

Huang, T. S. and Tsai, R. Y., "Image sequence analysis:Motion Estimation," in *Image Sequence Analysis*, Berlin, Germany: Springer-Verlag, pp. 1-18, 1981.

Iinuma, K., Koga, T., Nima, K. and Iijima, Y., "A Motion-Compensated Interframe Codec," *Proc. SPIE Image Coding*, vol. 594, pp. 194-201, 1985.

Jain, A. K., "Advances in Mathematical Models for Image Processing," *Proceedings of IEEE*, vol. 69, pp. 502-528, 1981.

Jain, A. K., *Fundamentals of Digital Image Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1989.

Jain, A. K. and Ranganath, S., "Image Segmentation and Edge Extraction Based on 2-d Stochastic Models," *Proc. IEEE ICASSP*, pp. 1520-1523, 1982.

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

Jayant, N. S. and Noll, P., *Digital Coding of Waveforms*, Englewood Cliffs, NJ: Prentice-Hall, 1984.

Kaneko, M., Hatori, Y., Koike, A., "Improvements of transform coding algorithm for motion-compensated interframe prediction errors-DCT/SQ coding," *IEEE J. Selected Areas in Commun.*, vol. SAC-5, no.7, pp. 1068-1078, Aug. 1987.

Koga, T., Hirano, A., Iinuma, K., Iijima, Y. and Ishiguro, T., "A 1.5 Mb/s Interframe Codec with Motion-Compensation," *Conf. Rec. ICC*, pp. D8.7.1 - 5, 1983.

Koga, T., Iinuma, K., Hirano, A., Iijima, Y. and Ishiguro, T., "Motion-Compensated Interframe Coding for Video Conferencing," *Proceedings of the National Telecommunications Conference*, New Orleans, LA, Dec, 1981, pp. G5.3.1 - G5.3.5, 1981.

Labit, C. and Benveniste, A., "Motion Estimation in a Sequence of Television Pictures," *Image Sequence Processing and Dynamic Scene Analysis: NATO ASI Series*, T. S. Huang, vol. F2, pp. 292-306, 1983.

Labit, C. and Benveniste, A., "Motion of Edges and Motion Estimation in a Sequence of T.V. Pictures," *Proc. IEEE ICASSP*, pp. 460-463, 1982.

Lee, C. and Nadler, M., "Predictive image coding with Pseudo-Laplacian edge detector," *IEEE Jounal on Selected Areas in Commun.*, pp. 1190-1206, Aug. 1987.

Lee, C. and Nadler, M., "Interframe image coding with 3-dimensional edge detection," *Proc. IEE 3rd Int. Conf. Image Processing and its Applications*, Jul. 1989.

Limb, J. O., Pease, R. F. K., and Walsh, K. A., "Combining intraframe and frame-to-frame coding for television," *Bell Syst. Tech. J.*, vol. 53, no.6, pp. 1137-1173, Aug. 1974.

Limb, J. O. and Murphy, J. A., "Measuring the speed of moving objects from television signals," *IEEE Trans. Commun.*, vol. COM-23, no.4, pp. 474-478, Apr. 1975.

Linde, Y., Buzo, A. and Gray, R. M. , "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, vol. COM-28, pp. 84 - 95, 1980.

Liu, H. K., "Two- and Three-Dimensional Boundary Detection," *Computer Graphics and Image Processing*, vol. 6, pp. 123-134, 1977.

MacQueen, J., "Some methods for classification and analysis of multivariate observations," *Proc. of the fifth Berkeley symposium on Math. Stat. and Prob. 1*, pp. 281-296, 1967.

Max, J., "Quantizing for minimum distortion," *IRE Trans. Inform. Theory*, vol. IT-6, pp. 7-12, 1960.

Murakami, T., Asai, K., Itoh, A. and Yamazaki, R., "Interframe vector coding of color video signals," presented at *Proc. Int. Picture Coding Symp.*, Jul. 1984.

Musmann, H. G., Pirsch, P. and Grallert, H. J., "Advances in Picture Coding," *Proc. IEEE*, vol. 73, pp. 523 - 548, 1985.

Nagel, H. H., "Overview on image sequence analysis," in *Image Sequence Processing and Dynamic Scene Analysis*, T. S. Huang, Ed., Berlin, Germanu: Springer-Verlag, pp. 2-39, 1983.

Nasrabadi, N. M., Lin, S. E. and Feng, Y., "Interframe Hierarchical Vector Quantization," *Proc. IEEE ICASSP*, pp. 1739-1742, 1989.

Nasrabadi, N. M., Lin, S. E. and Feng, Y., "Interframe Hierarchical Vector Quantization," *Optical Engineering*, vol. 28, pp. 717-725, Jul 1989.

Nasrabadi, N. M. and King, R. A., "Image Coding Using Vector Quantization: A Review," *IEEE Transactions on Communications*, vol. 36, pp. 957 - 971, 1988.

Netravali, A. N. and Limb, J. O., "Picture Coding: A Review", *Proc. IEEE*, vol. 68, pp. 366-406, Mar 1980.

Netravali, A. N. and Robbins, J. D., "Motion-Compensated Television Coding: Part 1," *The Bell Syst. Tech. J.*, vol. 58, pp. 631-670, 1979.

O'Neal, J. B., "Predictive quantizing differential pulse code modulation for the transmission of television signals," *Bell Syst. Tech. J.*, vol. 45, pp. 689-722, May-June 1966.

Pratt, W. K., *Digital Image Processing:* John Wiley and Sons, 1978.

Ramamurthi, B. and Gersho, A., "Classified Vector Quantization of Images," *IEEE Transactions on Communications*, vol. COM-34, pp. 1105 - 1115, 1986.

Roach, J. W. and Aggarwal, J. K., "Determining the movement of objects from a sequence of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, no.6, pp. 554-562, 1980.

Robbins, J. D. and Netravali, A. N., "Recursive Motion Compensation: A Review," *Image Sequence Processing and Dynamic Scene Analysis*, vol. F2, pp. 76-103, 1983.

Robert, P., Cafforio, C. and Rocca, F. , "Time/Space Recursions for Differential Motion Estimation," *Proc. SPIE Image Coding*, vol. 94, pp. 175-185, 1985.

Schalkoff, R. J. and McVey, E. S., "A model and tracking algorithm for a class of video targets," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-4, no.1, pp. 2-10, Jan. 1982.

Shannon, C. E., "A mathematical theory of communication," *Bell Systems Technical Journal 27*, pp. 379-423, 623-656, 1948.

Spiegel, M. R., *Mathematical Handbook of Formulas and Tables*, McGraw-Hill, 1968.

Srinivasan, R. and Rao, K. R., "Predictive Coding Based on Efficient Motion Estimation," *Proc. ICC*, pp. 516-520, 1984.

Topa, L. C. and Schalkoff, R. J., "Edge Detection and Thinning in Time-Varying Image Sequences Using Spatio-Temporal Templates," *Pattern Recognition*, vol. 22, no. 2, pp. 143-154, 1989

Tsai, R. Y. and Huang, T. S., "Estimating three-dimensional motion parameters of a rigid planar patch," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-29, no. 6, pp. 1147-1152, Dec. 1981.

Walker, D. R. and Rao, K. R., "New Techniques in Pel-Recursive Motion Compensation," *Proc. Int. Conf. Commun.*, vol. 2, pp. 703 - 706, 1984.

Zschunke, W., "DPCM Picture Coding With Adaptive Prediction," *IEEE Trans. Commun.*, vol. COM-25, no. 11, pp. 1295-1302, Nov. 1977.

Zucker, S. W. and Hummel, R. A., "A Three-Dimensional Edge Operator," *IEEE Trans. PAMI*, vol. PAMI-3, no. 3, May 1981.

Vita

Choon Lee was born in Seoul, Korea, on October 18, 1957. He received the B.S. degree in electronic engineering from Yonsei University, Seoul, Korea in 1981 and the M.S. degree in electrical engineering from Virginia Polytechnic and State University, Blacksburg, in 1986. He will receive Ph. D. degree at Virginia Polytechnic and State University also in electrical engineering in December 1990. His research interests include image compression, digital image processing, pattern recognition, digital signal processing and artificial intelligence.