OBJECT BASED VIDEO CODING

MD. AHSAN SHAMIM

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

OBJECT BASED VIDEO CODING

by

© Md. Ahsan Shamim

A thesis submitted to the School of Graduate

Studies in partial fulfillment of the

requirements for the degree of

Master of Engineering

Department of Electrical and Computer Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

August 2000

St. John's Newfoundland

ABSTRACT

In this thesis, I propose a method to segment moving objects in image sequences, so that each can be represented by its boundary and a compact motion description. I also propose a new binary tree coding technique to code motion boundaries in an efficient way. Assuming that a scene consists of a small number of moving objects, each pair of frames is processed as follows: First a small number of "movement classes" is recursively identified, each represented by two or more motion parameters. Second a spatially segmented version of the *reference frame* (i.e. the later of the two frames) is used to classify segments into movement classes. Third segments using various similarity heuristics are merged together. Finally the motion boundaries and vectors are efficiently coded. Experimental results of some standard test sequences have shown that the proposed algorithm results in good quality motion segmentation with a small number of motion vectors. The coding of motion boundaries uses a modified binary tree coding algorithm, which shows better performance than conventional quadtree, binary tree and READ coding algorithms applied on practical motion boundary images.

ACKNOWLEDGEMENTS

The author would like to express his profound gratitude to his thesis advisor Dr. J. A. Robinson for his valuable technical suggestions and parental guidance during the study period. His constant moral support, responsibility, above all his dedication for work had inspired the author that had raised his devotion for work. The author is very much lucky to get him as his supervisor.

The author is also indebted to his parents for their affection and sacrifice that has always made him cautious about their pride.

Heartiest thanks to the author's colleagues of the Multimedia Communications Laboratory especially Mr. Li-Te Cheng for their valuable technical and moral support.

Finally, the financial support by Dr. J. A. Robinson, the School of Graduate Studies, the Faculty of Engineering and Applied Science at Memorial University of Newfoundland, which has provided the author such a scope that has brought him more than a very allurable degree, are greatly acknowledged.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Motion estimation between frames in a video sequence is one of the key features of present day research in video compression. In general, a still image is a graphical representation of objects at a particular state and a video sequence is a collection of images representing sequential states of object positions. The change in object location in a video sequence results mainly from two causes: global motion due to camera movement and local motion due to intrinsic movements of objects in the scene. If the spatio-temporal intensity is assumed to vary due only to the local motions then the adjacent frames in a sequence are found to have some temporal redundancy. This temporal redundancy results in inefficient video compression if frames are coded independently. In order to obtain efficient video compression, successive frames should be coded by exploiting this redundancy. This leads to the idea of exploiting temporal redundancy between frames by identifying objects in motion.

When the corresponding object motion between adjacent frames in a video sequence is estimated, one frame is selected as a reference, termed the *reference frame*. A vector denoting the displacement of an object in the reference frame with respect to that in the current frame is determined. This vector is known as a *motion vector*. During reconstruction, the reference frame is used to predict the current frame using the motion

vector information. The object of interest in the reference frame that is referenced by a motion vector is copied into the reconstructed frame. This technique is known as *motion compensation* and the process of compressing video using estimated motion is known as *interframe coding*.

Many practical applications depend on the measurement of motion in a sequence of images, including dynamic scene analysis and understanding, image registration and stabilization, visual navigation and obstacle avoidance, and video data compression. To support a range of applications for communication, access, and manipulation of digital audio-visual data, the **Moving Picture Experts Group (MPEG)** was established in 1988 [1]. In August 1993, the MPEG group, responsible for coding of moving pictures and audio, released the MPEG-1 standard for the coding of moving pictures associated with audio at up to about 1.5 Mb/s. In 1990, MPEG started the MPEG-2 standardization phase as an improvement over the existing MPEG-1. The MPEG-1 standard was mainly targeted at CD-ROM applications whereas the MPEG-2 standard addresses substantially higher quality for audio and video with video bit rates between 2 Mb/s and 30 Mb/s, primarily focusing on the requirements for digital TV and HDTV applications.

The rapid convergence of telecommunications industries, computers, and TV/film industries forced the MPEG group to officially initiate a new standardization phase in 1994. The new standard headed towards algorithms for audio-visual coding in multimedia applications, allowing for interactivity, high compression, and/or universal accessibility and portability of audio and video content. Bit rates targeted for the video standard are

between 5 Kb/s and 64 Kb/s for mobile applications and up to 2 Mb/s for TV/film applications. This standard is called MPEG-4. Seven new key video-coding functionalities have been defined which support the MPEG-4 focus and which provide the main requirements for the work in the MPEG video group.

These functionalities are given as follows: [1]

Content based manipulation and bit stream editing: Content-based manipulation and bit-stream editing are important for audio-visual applications such as home movie production and editing, digital effects and interactive home shopping. MPEG-4 supports this functionality by providing MPEG-4 Syntactic Description Language (MSDL) and MPEG-4 coding schemes.

Hybridization of natural and synthetic data: Combining synthetic scenes or objects with natural scenes or objects is very interesting for games and other audio-visual applications. MPEG-4 supports this combined data coding more efficiently.

Improved temporal random access: This provides an efficient method to randomly access, within a limited time and with a fine resolution at a very low bit rate, video frames or arbitrarily shaped image content from a video sequence.

Improved coding efficiency: The use of fixed block partitioning on a fixed grid result in blocking artifacts and unnatural object motion at very low bit rates. In addition, it is

unable to support any of the content-based functionalities such as object scalability. The optimized syntax of these coders to improve coding efficiency makes them highly susceptible to channel noise. MPEG-4 video with object based partitioning will provide subjectively better visual quality at comparable bit rates compared to existing standards such as H.261.

Coding of multiple concurrent data streams: Multimedia applications such as virtual reality, 3D movies, and multimedia presentations require efficiently coding multiple views and soundtracks of a scene. MPEG-4 will provide the ability to code multiple views of a scene efficiently.

Robustness in error-prone environments: Wireless communication requires error robustness for low bit-rate applications under severe error conditions. MPEG-4 provides an error robustness capability to allow access to the multimedia applications over a variety of wireless and wired networks and storage media.

Content based scalability: Browsing multimedia objects from a database and selection of quality of objects in a decoding process require content-based scalability. MPEG-4 will provide the ability to achieve scalability with fine granularity in content, quality, and complexity.

The common theme through these functionalities is object-based processing which focuses research and development on schemes for automatically identifying objects and

coding them. There has been considerable research in object-based coding for very low bit-rate video compression. These techniques achieve efficient compression by separating coherently moving objects from stationary background and compactly representing their shape, motion and the content [2]. Some of the advantages of these approaches include a more natural and accurate rendition of the moving object and efficient utilization of the available bit rate by focusing on the moving objects. In addition to these compression advantages, if suitably constructed, the object based coding techniques can also support content based functionalities such as the ability to selectively code, decode, and manipulate specific objects in a video stream. The ability to scale the bit stream at an object level can also be supported by these techniques. To elaborate, the decoder can use only part of the original coded bit stream to selectively decode a subset of the coded objects at varying quality and resolution for each individual object.

The increasing availability of potentially interesting material makes retrieval of relevant information harder. Multimedia databases on the market today allow only limited capability, domain-limited searching for pictures and video, using characteristics like color, texture and information about the shape of objects in the pictures and video. On the contrary, object based coding is associated with the indexing of the available audio-visual (AV) information, which will facilitate an effective search of multimedia data [3]. If image/video can be stored in the form of individual objects, retrieval of multimedia information is as simple as that of textual information. Therefore, the need of tools for acquiring the objects for these purposes is stringent. But tasks of automatically

segmenting image sequences into semantic meaningful objects prove to be very challenging.

Interest is growing to support the capability to interact with a movie by retrieving any kind of information related to object itself. This process is often referred to as *hypermedia*, merging the hypertext concept with that of mixed information sources. Information must be linked to the video object, and the user can access it both sequentially and directly by means of queries. Also, the link must be bi-directional, in that the viewer can either select the object and retrieve the information or select the type of information and go to the corresponding object. Another advantage of this scheme is that it is not necessary to decode the full sequence if the user wants to access only part of it. This supports object-based filtering. These practical ideas are at the root of the new MPEG-7 standard.

The objectives of MPEG-7 are as follows: [4]

- Allow fast and efficient searching for multimedia material of user's interest.
- Specify a standard set of schemes to describe various types of multimedia information.
- Specify encoding of description schemes and descriptors.
- Descriptors are associated with the content itself.
- Descriptors must be meaningful in the context of application.

- Descriptors are independent of the encoding of the content.
- Any type of audiovisual material can be retrieved by any type of query material.

MPEG-4 is an international standard that provides core technologies for efficient object-based compression of multimedia contents for transmission, storage, and manipulation, whereas MPEG-7 addresses content description technologies for efficient and effective multimedia retrieval and browsing. It extends the limited search capabilities of multimedia (pictures, graphics, 3D-models, audio, speech, video) in MPEG-4. MPEG-7's standardized format for content description will benefit intelligent multimedia retrieval systems for industry profiles such as content management, consumer electronics, multimedia systems, surveillance, and intellectual property management.

There are many features in images of a video sequence such as color, intensity, regular patterns (texture), and motion information etc., that affect video compression. In general, when an object experiences a motion the whole object (or the grouping of a few regions) undergoes a similar kind of motion irrespective of color, intensity and texture contents of the object. So, the motion information of an image is much more homogeneous than other features to be exploited in video compression. The MPEG standard is based on predicting successive frames from a reference frame using motion information.

1.2 Objectives

Considering current developments in MPEG-4 and MPEG-7 standards, the objectives of this thesis are as follows:

- To develop a content based motion estimation/motion segmentation technique that is robust, and fast. The innovation of this thesis is that the technique will work in a top-down fashion, not by agglomerating lots of local motion estimates, but by successively splitting global estimates into regional estimates.

- To find a suitable coding technique to pass the motion information efficiently. The criteria include increased coding efficiency for the boundary component of data relative to other two-level schemes.

1.3 Organization of the Thesis

Chapter 2 contains a review of previous approaches to motion estimation, motion segmentation and their coding techniques. As the field is quite broad, those which provided the foundation for the new approach, in both rationale and design detail, are emphasized.

Chapter 3 contains a detailed description of the new algorithm, starting with the new content- based motion estimation and motion segmentation technique. With two frames of

a video sequence, a small number of "movement classes", each represented by two motion parameters are recursively identified and assigned to the spatially segmented version of the *reference frame* (i.e. the later of the two frames). These motion estimation and segmentation details cover the first part of this chapter. Then an efficient coding scheme is developed to pass the motion information and to predict the later of the two frames from the first one. This coding detail is explained in the second part of this chapter.

Chapter 4 contains detailed results from testing the various options available under the new motion-estimation and motion segmentation technique. The results are compared and analyzed against the conventional block matching motion estimation technique. Comparison criteria are threefold: physical object extraction in movement, the required number of transmitted motion vectors and the quality of the predicated image. The object extraction, an important issue in MPEG-4 and MPEG-7, is examined by analyzing motion vector boundaries and the quality of the predicted image is assessed both visually and quantitatively, with respect to PSNR *(Peak Signal to Noise Ratio)*.

Chapter 5 contains the detailed results from testing the various options available under the new boundary coding scheme and its application to the motion information transmission. Results are compared and analyzed against TIFF (Tagged Image File Format) based G4 fax and conventional hierarchical quadtree and binary tree coding schemes. Comparisons are based on the relative performance in terms of required number of transmitted bits to pass equal information. This chapter also includes results on the

application of the new approach to the motion information transmission at variable rates and the effectiveness of codewords in terms of the information content. The coding scheme is also applied to some textual images to find its effectiveness.

Chapter 6 summarizes the work and suggests some avenues for further development.

CHAPTER 2

REVIEW OF LITERATURE

Segmentation of a video into individual moving objects is an active research topic in digital video processing. It provides the basis for measurement and annotation as well as object based interaction and compression. Many techniques and experimental results have been published on object based motion estimation and coding. Here a brief description of the literature related to the objectives of this work is presented. The literature survey can be divided into three main areas:

- General motion estimation algorithms.
- Content based motion estimation and segmentation.
- Coding of motion information.

2.1 General Motion Estimation Algorithms

Observation of practical world images suggests that moving pictures have a *pixel conservation* property that the pixels on one frame may be translated to form the pixel pattern on a subsequent frame. Of course, changes in the position of the object may expose parts of the object to the camera that were previously unseen, and changes in the camera position may cause large-scale changes in the image. Moreover, textures in images also cause ambiguous motion estimation. These factors have made motion estimation in scenes containing multiple moving objects still a difficult problem in

computer vision. A number of different approaches have been used for estimating motion such as differential methods, region-based matching and phase-based techniques. While the first two categories estimate motion from the original pixel intensities, frequency-based techniques transform the image data into a new domain, often the frequency or Fourier domain, and estimate motions from the transformed coefficients.

2.1.1 Differential Methods

Differential methods of the motion estimation techniques rely on the following two assumptions: 1) the illumination is uniform along the motion trajectory and 2) the occlusion problem is neglected. The first assumption means that changes in intensity with time are assumed to be due only to the motion of two-dimensional intensity patterns, neglecting the problem of illumination change over time. The occlusion problem of the second assumption refers to the possibility of *uncovered background*. For the area of an uncovered background in the *reference frame*, no optical flow can be found. Although these assumptions do not always hold for real world video sequences, they are still used by many motion estimation techniques. In general, there are two types of *gradient based techniques* in motion estimation [5].

Raw Gradient Scheme

Raw gradient-based schemes compute visual motion directly from the ratios of temporal to spatial image irradiant gradients. Visual motion computed this way is usually called *optical flow*. The idea was first used in a 1D signal by **Limb** and **Murphy** [6] and introduced in 2D imagery by **Horn** and **Schunk** [7]. The advantage of a raw gradient-

based scheme is that it can be applied uniformly across the image, although where the spatial or temporal gradients are small the results will be sensitive to noise. A more obvious disadvantage of the method is that only the component of image motion along the gradient direction can be derived. This is known as the *aperture problem* in motion estimation.

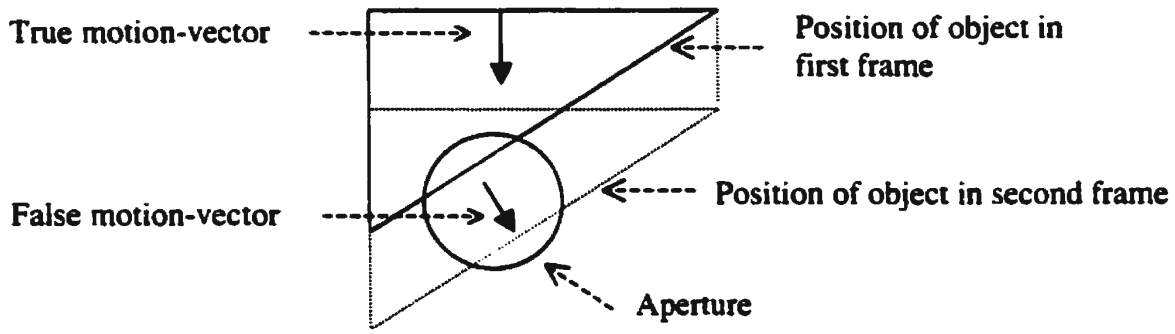


Figure 2.1: The aperture problem.

Edge Gradient Scheme

There is a wide consensus that the information rich areas in an image lie at its illumination discontinuities, the image edge points. **Marr** and **Ullman** [8] proposed that the human visual system computes motion by temporal filtering of edge signals found at the zero-crossings of the signal formed by convoluting the image intensity with the **Marr-Hildreth** operator [9]. Their work was extended by **Buxton** and **Buxton** [10], **Duncan** and **Chou** [11]. Advantages of zero-crossings are that they correspond to points where the gradient is locally maximum, thereby reduces error. Secondly, they are tied more closely to physical features. If the zero-crossings move, it is more likely to be the consequence of movement of the underlying physical surface. But the disadvantage of

edge gradient schemes is that they suffer from the aperture problem in just the same way that raw gradient schemes do.

2.1.2 Region Based Matching

Out of many region-based motion estimation algorithms, the block-matching technique is the most popular. A block–matching motion estimation technique proposed by **Jain et al.** [12], provides a simple and elegant way to identify and express motion, and hence, is commonly adopted in many video compression standards (e.g. ITU-T H.261/H.263, and MPEG-1, 2). By dividing each frame into rectangular blocks, motion vectors are found corresponding to the location that gives the minimum *Frame Distortion Measure (FDM)*, i.e. Mean Absolute Error (MAE) of the block, within a search region. A full search (FS) algorithm searches all possible locations inside the search window of the reference frame to provide an optimal solution. There are also some faster but acceptable search-algorithms. Some of these techniques are discussed below.

Three Step Search

The three-step search algorithm was proposed by **Koga et al.** [13] in 1981. The algorithm is based on a coarse-to-fine approach with logarithmic decreases in step size. The initial step size is half of the maximum motion displacement d (i.e. $\lceil d/2 \rceil$ where $\lceil . \rceil$ is the upper integer truncation function). For a value of d the number of checked points is equal to $\{1+8\lceil \log_2(d+1) \rceil\}$.

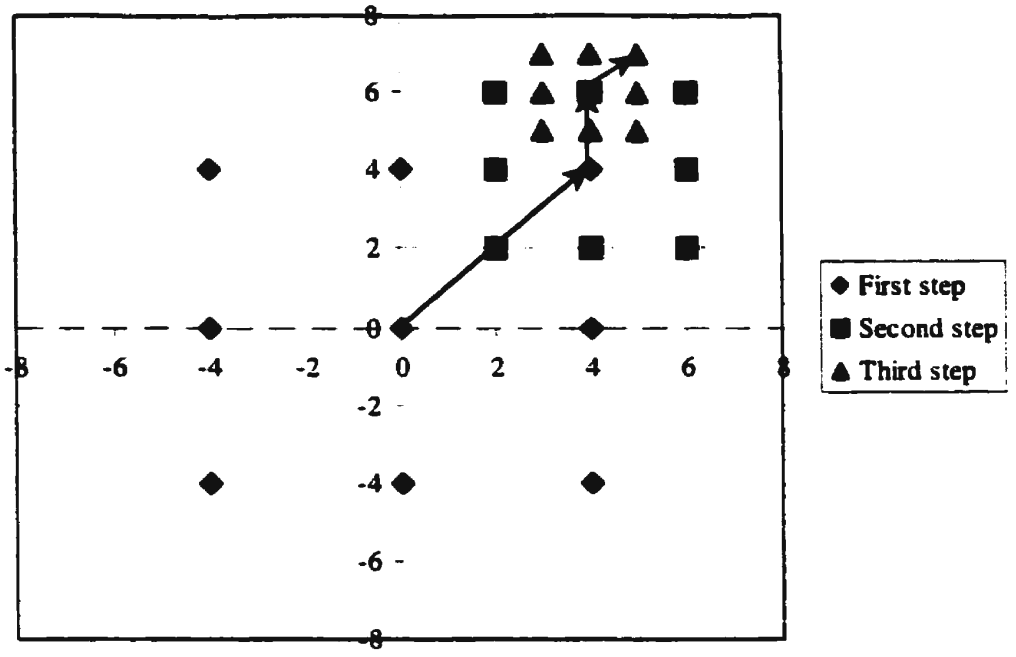


Figure 2.2: Three-step search algorithm.

Gradient Descent Search

The Block-based gradient descent search algorithm was proposed by **L. K. Liu** and **E. Feig** [14] in 1996. This algorithm uses a center-biased search pattern of nine checking points in each step with step size of one. It does not restrict the number of searching steps but it stops when the minimum checking point of the current step is the center one or the search window boundary has been reached. This approach performs better in searching small motions.

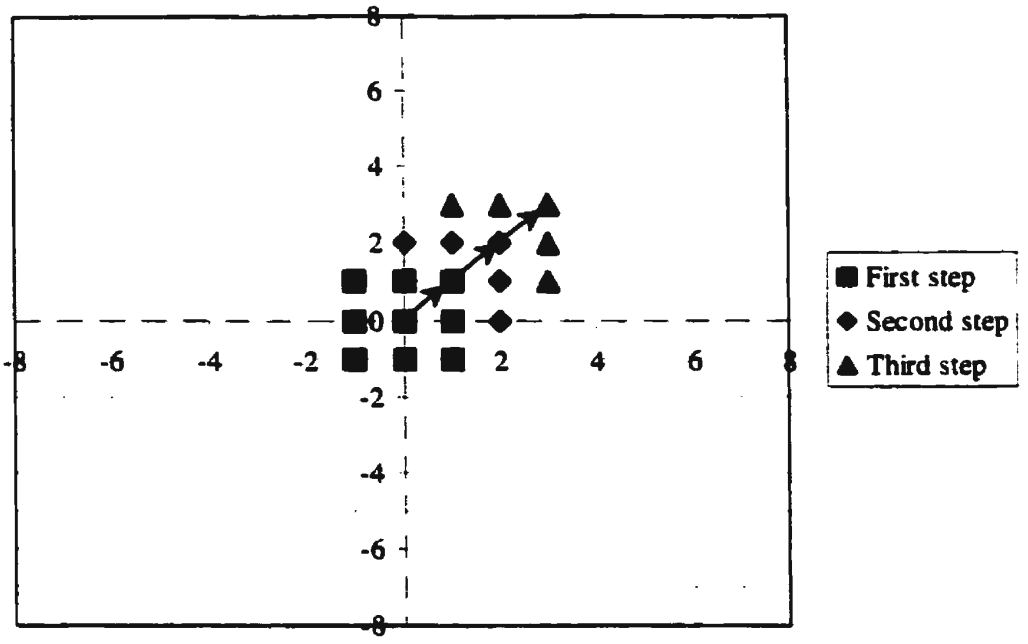


Figure 2.3: Gradient descent search algorithm

Hierarchical Block Matching Algorithm

The hierarchical block matching algorithm was proposed by **M. Bierling** [15] in 1988. The basic idea of hierarchical block matching is to perform motion estimation at each level successively, starting with the lowest resolution. The estimate of the motion vector at a lower resolution level is then passed onto the next higher resolution level as an initial estimate. The motion estimation at the higher level refines the motion vector of the lower one. At higher levels, a relatively smaller search window can be used as it starts with a good initial estimate.

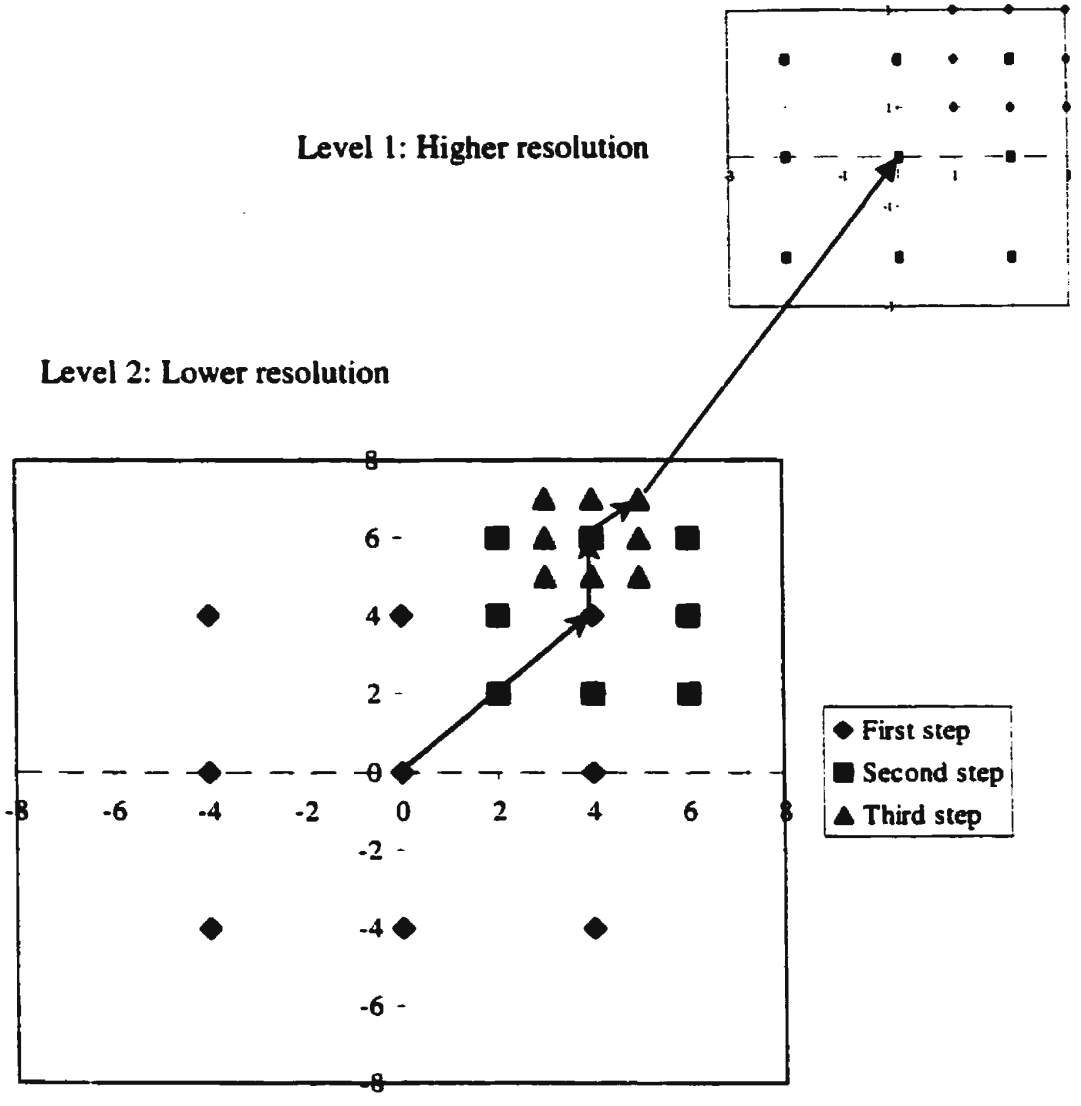


Figure 2.4: Hierarchical block matching algorithm

2D-Logarithmic Search

2D-logarithmic search was proposed by **Jain et al.** [12] in 1981. It uses a (+) cross search pattern in each step. The initial step size is $\lceil d/4 \rceil$. The step size is reduced by half only when the minimum *Block Distortion Measure* point of previous step is the center one or

the current minimum point reaches the search window boundary. Otherwise, the step size remains the same. When the step size is reduced to 1, all the eight checking points adjacent to the center checking point of that step are searched. The algorithm is faster than the steepest descent technique if the BDM converges to the desired point. But the algorithm has higher probability of being divergent than the steepest descent technique.

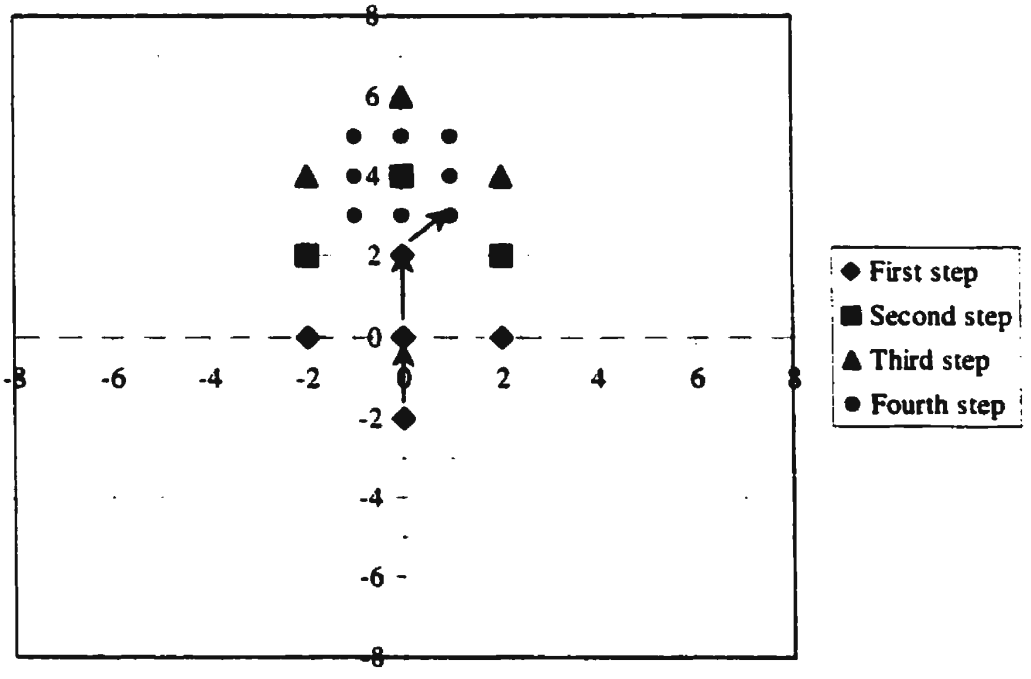


Figure 2.5: 2-D logarithmic search algorithm.

Cross Search

The cross search algorithm was proposed by **Ghanbari** [16] in 1990. It is also a logarithmic step search algorithm using a (X) cross searching patterns in each step. The initial step size is half of d. As the step size decreased to one, a (+) cross search pattern is used if the minimum block distortion measure point of the previous step is either the

center, upper-left or lower right checking point. Otherwise a (X) cross search pattern is used.

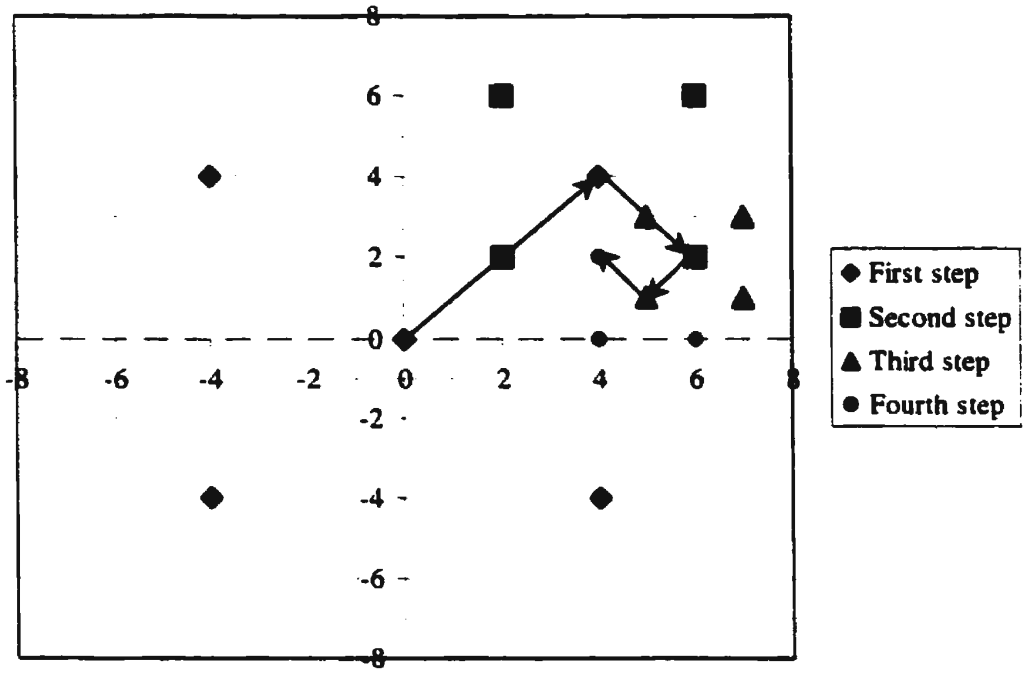


Figure 2.6: Cross search algorithm.

Quadtree Methods

Though most of the earlier block-matching works were based on fixed block-size, **Chan et al.** [17] proposed the use of variable-sized blocks for motion estimation so that, where appropriate, large areas of uniform motion could be represented by relatively few blocks, thus minimizing the required number of motion vectors. They use a "top-down" approach in which initially large blocks are matched, and if for the best match of any block, the resulting error is above a prescribed threshold, then that block is split into four smaller blocks. This process is repeated until the maximum number of blocks, or locally

minimum errors, are obtained. Finally a process of re-merging small blocks to form large blocks is performed to remove blocks that do not contribute to improving image quality.

Rhee et al. [18] proposed a quadtree-based algorithm for variable-size block matching motion estimation. The scheme allows the dimension of blocks to adapt to local activity within the image, and the total number of blocks in any frame can be varied. This permits adaptive bit allocation between the representation of displacement and residual data, and also the variation of the overall bit-rate on a frame-by-frame basis. For fixed bit-rates, large block errors result in increased information loss and produce lower image quality. Thus, minimizing block error is an important goal of motion estimation. Ideally, to achieve good video compression ratio and image quality, both the number of blocks and the block error have to be minimized because the motion vectors are encoded along with the block error. Unfortunately, this is a conflicting requirement, particularly with fixed-size block matching (FSBM), where the size of all blocks is the same. In FSBM, increasing the block size is the only way to reduce the number of motion vectors. However, its success depends on an appropriate selection of blocks. This poses an interesting optimization problem.

Though **Chan et al.** reported a significant improvement in quality over FSBM techniques for relatively low bit-rate coding, the algorithm proposed by **Rhee et al.** finds the optimal covering quadtree of a specific number of variable-sized blocks which results in the minimum motion compensation error. In terms of motion estimation accuracy, it provides

the best achievable performance of a quadtree structured VSBM technique for any prescribed number of blocks.

2.1.3 Phase Based Technique

Whilst the vast majority of estimation algorithms operate in the spatial domain, frequency domain techniques have also been developed from time to time. These make use of a basic property of the Fourier transform that the magnitude spectrum of a signal displaced in time is the same as that of the undisplaced signal but the phase spectrum undergoes a shift that is linearly proportional to the spatial displacement. Thus, assuming reasonably uniform motion from frame to frame, the Fourier transform of a given frame difference signal should be the same as its previous counterpart when the two-dimensional phase shift term is taken into account. Over the moving area, therefore, motion compensation takes the form of a phase shift adjustment, and taking the average shift allows compensation by a non- integral number of elements. It is suggested by **Haskell** [19] that this be combined with predictive coding of the frame difference signal, which upon inverse transformation provides a motion-compensated prediction for the present frame difference.

Magarey et al. [20] has proposed a phase-based matching technique using complex-valued coefficients, which are produced by a complex-valued filter pair. Here the filtering consists of first and second quadrant information of an image. The advantages of the CDWT lie in its hierarchical structure of motion estimation and its orientational selectivity. But this hierarchical structure of the CDWT algorithm is best suited to

estimating continuous motion fields, such as those resulting from camera motion relative to a distant or flat static scene. Motion estimated at finer levels in the pyramid suffers from the aperture problem like other motion estimation techniques. To avoid the aperture problem, motion is estimated initially at a course level and passed to the next finer level as a reference. This hierarchical method has problems at motion discontinuities, because of the implicit assumption of motion field smoothness inherent in a coarse-to-fine approach. If motion estimated in a coarse level is passed to the next finer level uncorrected then motion boundaries will be wrongly located.

2.2 Content Based Motion Estimation

Content based motion estimation is the key element of current MPEG-7 standard. Algorithms for region-based motion estimation can be divided into two classes. In the first class, the current image of a video sequence is segmented into regions, and the motion of each region is estimated with respect to the previous image [21-22]. In the second class, the previous image is segmented and the motion of each region in the previous image is estimated with respect to the current image [23-24].

The work by **Kunt et al.** [25] is one of the pioneers in the field of content based video compression. They suggested that describing images in terms of physical entities such as contours or regions is more meaningful than using other entities such as intensity and the color of the image. This should lead to more compact representations and hence to higher compressions.

Musmann et al. [26] proposed an object-oriented analysis-synthesis coder, which encodes objects instead of fixed sized blocks of picture elements. They described the objects by three parameter sets defining the motion, shape and color of an object. From several experiments they concluded that object based coding is able to synthesize images which look more natural than the images predicted by block oriented coding. Moreover, they suggested that the knowledge about the object boundaries could be used to improve the coding gain of the intraframe transform coder.

Shi et al. [27] has proposed a motion segmentation algorithm that aims to break a scene into its most prominent moving groups. A weighted graph is constructed on the image sequence by connecting pixels that are in the spatio-temporal neighborhood of each other. At each pixel, motion profile vectors are defined that capture the probability distribution of the image velocity. Here, the distance between motion profiles is used to assign a weight on the graph edges. The partition of the image is obtained by the eigen-vector calculation of the weight matrix. The main drawback of their scheme is that the size of the weight matrix is proportional to the square of d, where d is the number of pixels in the image. So it is quite time consuming to get the eigen vector for a large matrix. In addition, the partition was found to be quite sensitive to the parameters they have used.

Nguyen et al. [28] proposed a motion segmentation technique. An important step in bottom-up motion segmentation algorithm is the merging of regions exhibiting similar motion. In general the similarity between motion of two regions is defined as the euclidean distance between the corresponding vectors of motion parameters obtained by the least-square estimation. This measure is found sensitive to noise. Unlike other schemes, their scheme makes the decision on equality of motion parameters in two regions via a statistical test. Though they have developed a

robust motion segmentation technique, the basic motion estimation algorithm is based on the optical flow-measurement. This has the inherent drawbacks of motion estimation using optical flow techniques.

In video sequences, object motion causes regions to be covered or uncovered. Uncovered regions may seriously decrease the accuracy of motion estimation and hence increase the displaced frame difference (DFD). **Wang** [29] proposed a technique to improve the accuracy of region-based motion estimation by considering uncovered regions in the image segmentation and motion estimation procedures. His technique involves only the first class of motion estimation algorithms, in which the current image is segmented. The improved algorithm consists of three steps: initial segmentation and motion estimation, uncovered regions detection, and improvement of segmentation and estimation.

2.3 Coding of Motion Information

In general, object-based coding schemes must spend most of their bits on coding boundaries. So the aim of this work is to code the two-level motion boundary or object contour images in an efficient way. There are many techniques used in two-level coding. Some of these are briefly discussed below [30]:

2.3.1 Scan-Based Schemes

Run-length Coding (RLC): The simplest of two-level coding schemes is the run-length coding technique. Here the distances between adjacent transitions in the picture level are transmitted. **Figure 2.7** shows an example of RLC to transmit runs of 3, 4, 1, 8 and 4 pixels using a fixed

eight bit codeword. Runs more than maximum possibie run-length with a specific number of bits are coded using a make-up codeword plus a terminator.

Bit stream: 00000011/00000100/00000001/00001000/00000100

Figure 2.7: Run-length coding scheme.

Relative Address Coding (RAC): The principle of relative address coding is illustrated in **Figure 2.8**. Transitions on the current line most often occur close to related transitions on the previous line. By coding the displacements between the current line's transitions and their neighbours on the previous line, this closeness is exploited. The result is that most transitions are represented by codewords denoting displacements, or relative addresses of 0, ±1 or ±2.

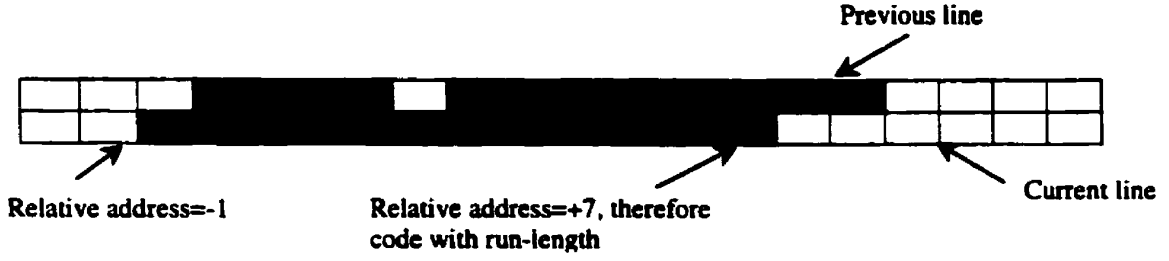


Figure 2.8: Relative address coding scheme.

Relative Element Address Designate (READ) Coding: READ coding is more efficient than the straightforward RAC. Here each transition is coded according to the relative positions of four surrounding transitions. **Figure 2.9** illustrates the READ coding scheme. B is the current transition now to be coded and A is the reference point immediately before B. Usually A will be at a transition as shown in the diagram, but it is not always. C is the next coding element to the right of B. D is the first transition on the previous line to the right of A and of the same sense as B, and E is the next transition after D on the previous line.

There are three modes in a READ coding scheme. The first is the pass mode. This is detected if E occurs before B; that is a run on the previous line has no corresponding run on the current line. This is coded with a fixed codeword. The current line reference point (A) is then reset to the pixel directly below E (this is the case where the reference point is not a transition). Secondly, the vertical mode is detected if the distance between B and D is less than or equal to three (in either direction). This mode is coded using the relative address of B with respect to D. If the distance between B and D is greater than three, the run-lengths AB and BC are transmitted using Huffman codewords (horizontal mode).

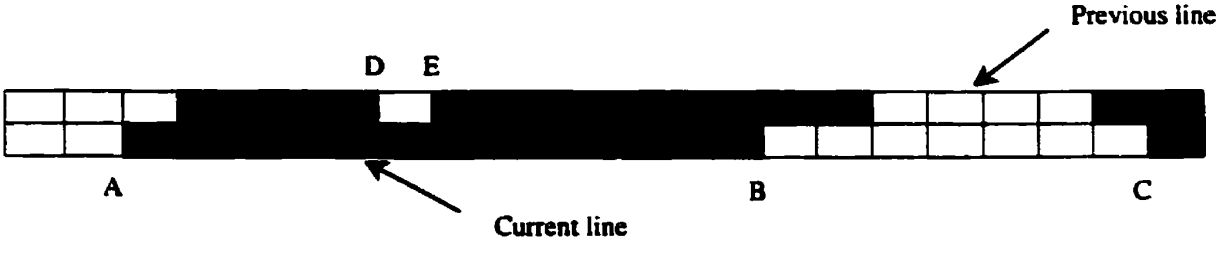


Figure 2.9: READ coding scheme.

2.3.2 Contour Schemes

Chain Coding: Contour coding is sometimes very useful to represent object or motion vector boundaries. Chain coding [31] is a popular contour coding scheme. In this scheme the start point of the contour is identified by an absolute address, the contour is traced by defining the position of the adjacent pixel and by transmitting the direction of the current pixel relative to the reference pixel. **Figure 2.10** shows a *chain coding* scheme.

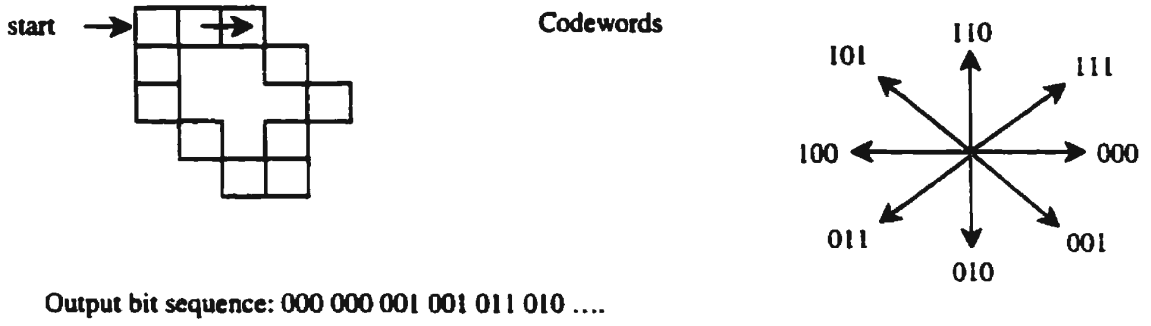
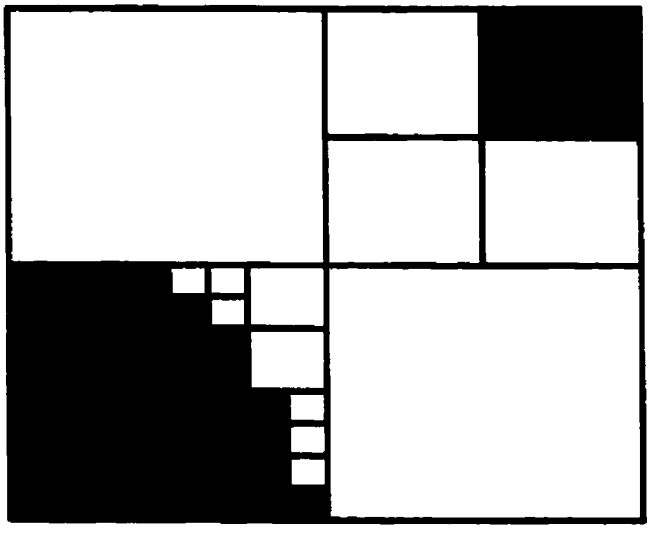


Figure 2.10: Chain coding scheme.

Koplowitz et al. [32] have considered a chain coded representation of digital contours and proposed a multi-resolution representation scheme using a pyramidal structure to obtain decreasing levels of resolution from the fine resolution data. The disadvantage of this scheme is that number of bits to transmit the information is proportional to the number of points of interest where the proportionality constant corresponds to the number of bits required for each direction representation.

2.3.3 Block Oriented Schemes

Quadtree Coding: Block location coding (or hierarchical coding) is a region based coding scheme, which is very efficient to code a chunk of similar adjacent pixels. Quadtree coding is a simple and an efficient block-oriented coding scheme. In general, a square block of size N×N is progressively quartered until each sub-block is codable. **Figure 2.11** illustrates a general quadtree coding technique.



Subdivision codebook:

0	subdivide
10	all black
11	all white

For 2×2 block which needs to be subdivided:
1: black and 0: white

Output bit sequence:

First pass:	0
Second pass:	110011
Third pass:	11101111 100100
Fourth pass:	0111011 100100
Fifth pass:	0010 1010 1011

Figure 2.11: Quadtree coding scheme.

Binary Tree Coding

In this area of image compression and transmission, a representation of pictures by binary trees was explored by **Knowlton** [33]. In his scheme a binary tree which describes the image is transmitted top-down (using a prefix notation). If the image is uniformly black or white, only one symbol is transmitted-the graylevel of the image. If not, the transmitter

sends a meta-symbol, which indicates that the picture is not uniform and that the code that follows represents two subareas of the picture. A single cut that divides the whole picture into two equal parts creates the two subareas. The coding proceeds in this fashion recursively until the transmitted code exhausts all the uniform sub- and sub-subareas of the picture.

Cohen et al [34] proposed an improved binary tree coding scheme that outperformed the Knowlton's scheme. In their scheme they have followed the same technique suggested by Knowlton. But they optimized the total number of cuts finding the effective direction of cut of the block at each processing stage. If the block has at least one black pixel it is said to have activity. The idea is to choose cut directions to minimize the number of blocks with activity. This scheme required the direction of cut and the structural information of the block to be transmitted.

A detailed explanation of the quadtree coding scheme and the binary tree coding scheme [34] is given in **Chapter 3** in order to compare their coding algorithm relative to the new coding technique developed during my thesis. Also **Chapter 4** shows some results of their application on contour images and performance of these two conventional techniques is compared against the new approach, which is also a block-oriented scheme. So it is suitable to discuss these two techniques with the description of the new approach.

2.4 Summary of Literature Review

2.4.1 Motion Estimation Algorithm

Between different motion estimation algorithms, block-matching techniques are found to be simple and relatively insensitive to noise. Both differential methods and phase-based techniques depend on the features of an individual pixel. If a pixel represent noise then these two methods will fail to estimate the correct motion vector for that pixel.

Traditional block matching techniques suffer mainly from two limitations. Firstly: there is always a dilemma in the selection of the block size. If the block size is small then there is a higher probability of estimated motion for a block being sensitive to noise. On the other hand, a larger block size may result in a block consisting of segments from two separate objects moving with different motions. So a top-down region-matching algorithm can avoid the selection dilemma of the block size, where the non-representative pixels corresponding to the calculated motion vector at the current stage will form the region to be considered in the next stage. As the region at the first stage corresponds to the whole image so the full search algorithm, which is the most reliable way to find the global minimum point, will be very time consuming. The steepest descent search algorithm is found suitable for this and a few following stages. But the reliability of the steepest descent technique reduces as the number of pixels to represent a object becomes less. Instead of getting global minima this search technique may find local minima. In that case a full search algorithm will be appropriate.

2.4.2 Motion Segmentation

Traditional motion segmentation techniques follow two main steps. At first the motion information at each pixel of the desired frame in a video sequence is estimated. Then a merging algorithm is followed to agglomerate pixels with similar motion. These techniques in general require a stopping condition to find different moving objects, which is in most of the cases the assignment of the total number of objects with different motions. The disadvantage of this technique is that wrong motion estimation of some pixels in an object may result in the merging of two differently moving objects instead of the desired ones. So a different motion segmentation technique that does not depend on the assignment of total number of moving objects and assign the same motion vector to a rigidly moving object will be more practical and robust.

2.4.3 Motion Boundary Coding

Chain coding techniques depend on fixed length codewords. The desired codeword corresponds to the position of the next adjacent uncoded pixel relative to the current coded pixel. The total number of transmitted bits in a chain-coding scheme is proportional to the number of active pixels in an image, which becomes inefficient for a large number of pixels. On the other hand, in *Run Length Coding* scheme, if there is a possibility of large run-lengths then each run-length needs to be coded by a large number of bits and the technique is inefficient in the case of smaller run-lengths. So a region based technique is suitable to code motion boundaries. The basic idea behind a hierarchical coding scheme is to segment a picture into the largest possible uniform areas and to transmit a hierarchical representation of these areas. In the traditional binary tree method the total number of cuts

is optimized finding the effective direction of cut at one stage of processing. The deficiency of these schemes is that direction information is very costly in terms of information transmitted. Moreover the optimization of the cut position was not considered previously. Instead a fixed cut position, the middle of the blocks, was used. Although there is no such choice of cut direction in a quadtree scheme, again there is a fixed cut position and the optimization of the cut position was not considered here too.

In consideration of some limitations of present day content based motion estimation and segmentation techniques and their coding, a new algorithm is developed in this thesis that eliminates some of the existing problems. One interesting point of this new technique is that the order of calculated motion vectors represents the relative layer of moving objects, which is very useful information in predicting images from motion information and the reference frame. Moreover, a new binary tree algorithm is suggested to code motion boundaries. The algorithm is explained in detail in the next chapter.

CHAPTER 3

THEORETICAL ANALYSIS

This chapter consists of the theoretical model of our content based motion estimation and segmentation technique. It also explains a new binary tree scheme together with conventional quadtree and binary tree coding algorithms.

3.1 General Description

With two frames in a video sequence to estimate motion, we first consider the whole image as a single moving object, and, by gradient descent, find a best-estimate translational motion estimate. This generally represents background or camera motion, and parts of the image that have this motion are in the first "movement class". The matching errors that result from displacing the second frame by this motion and subtracting the first frame are thresholded. Following a morphological operation to remove noise, the pixels with matching error greater than the threshold are judged as outside the first movement class, and are considered as a non-contiguous block for a second round of gradient-descent motion estimation. The best estimate motion parameters for this new block define the second "movement class", and, again, the difference image is thresholded and the super-threshold points are used to form a new block. The process repeats until most of the picture is sub-threshold for one of the motion vectors. In general, this technique finds the motion vectors of large contiguous regions before those of smaller regions. After a certain stage of motion vector calculation only small, unrelated

objects remain. At this stage gradient descent techniques become unreliable. So, a full search algorithm is applied if the cross-correlation coefficient between two test frames with remaining objects falls below some threshold (i. e. 0.1) at an estimated motion vector.

Figure 3.1 shows the proposed motion estimation algorithm.

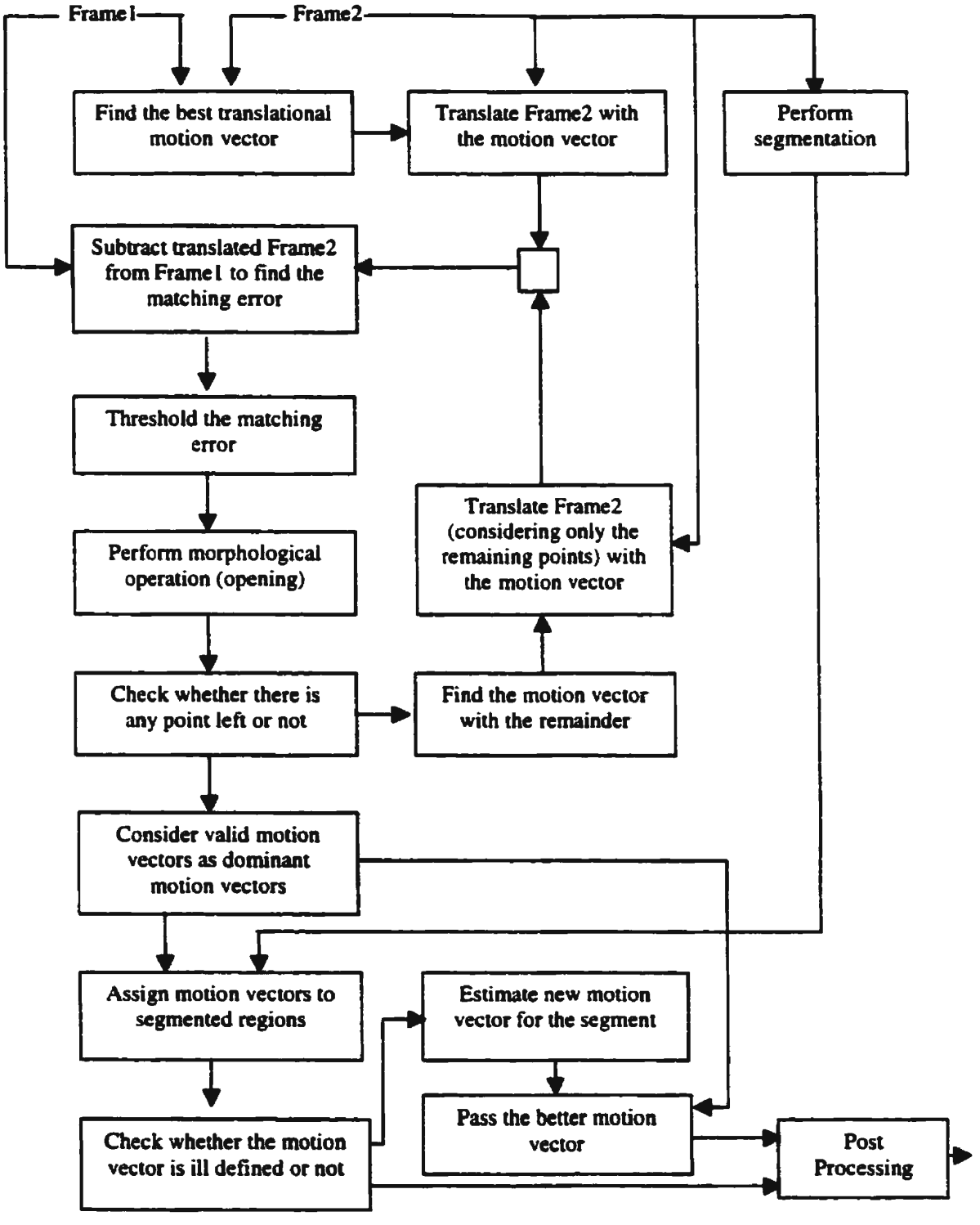


Figure 3.1: Proposed motion estimation algorithm.

3.2 Translational Motion Vector Estimation

With two frames in a video sequence, the best translational motion vector is found using a region matching algorithm. Here, the block is the objects of interest in the frame, in our case it is the pixels remaining after each stage of motion vector calculation. The search algorithm is the steepest-descent technique, which is computationally non-expensive and reliable for a large number of contiguous points. After a few dominant motion vector calculations, the contiguity of points corresponding to objects diminishes. So the steepest descent technique becomes unreliable. In that case, Full-search algorithm is used with a search range of ±32. For switchover from steepest descent algorithm to full-search algorithm, a threshold in cross-correlation co-efficient between images is considered (i.e. 0.1). The best translational motion vector corresponds to the search location giving minimum Frame Distortion Measure (FDM). The FDM between two frames Frame1 and Frame2 at a search location (m, n), m and n corresponding to the vertical and horizontal shift of the block respectively, is given by **Equation 3.1**.

$$FDM_{(m,n)} = \sum_i \sum_j \left| Frame1[i+m][j+n] - Frame2[i][j] \right| \tag{3.1}$$

Here, i and j correspond to the vertical and horizontal position of a pixel of interest. The summation is performed over all i and j that will cover all pixels of interest within the block.

3.3 Threshold Selection

Figure 3.2(a) and **(b)** shows two frames of the *mobile and calendar* video sequence. With these two frames for motion estimation, the first dominant motion vector corresponds to the camera motion. When the second frame is translated with this vector and subtracted from the first frame, it forms a difference image, which is shown in **Figure 3.2(c)**. Ideally, the difference image should remove stationary objects such as the background, the stationary part of the toy and the ground in front. But the difference image shows only a small number of removed points, represented by white marks, corresponding to the stationary objects.

The difference image consists of the remaining points that are part of the stationary objects. The value of the difference image at a particular pixel is termed as the matching error corresponding to that pixel due to the calculated motion vectors. Most of the matching errors corresponding to objects following the calculated vector lie in the lower range. If these points are not removed they may affect subsequent motion vector calculation. In order to remove these points, matching errors are thresholded. **Figure 3.2(d)** and **(e)**, shows matching errors thresholded at two different values. If the threshold is too low, **Figure 3.2(d)**, most of the pixels corresponding to objects following the calculated motion vector remain. This will cause some wrong motion vector calculation in the subsequent stages. On the other hand, if the threshold is too high, **Figure 3.2(e)**, most of the pixels corresponding to objects following the calculated motion vector are removed. In addition some pixels corresponding to objects following other vectors are

also removed. This will cause the lack of sufficient pixels for those motion vector calculations at later stages. So there is always a dilemma of threshold selection. This makes the choice of threshold important in classifying matching errors.

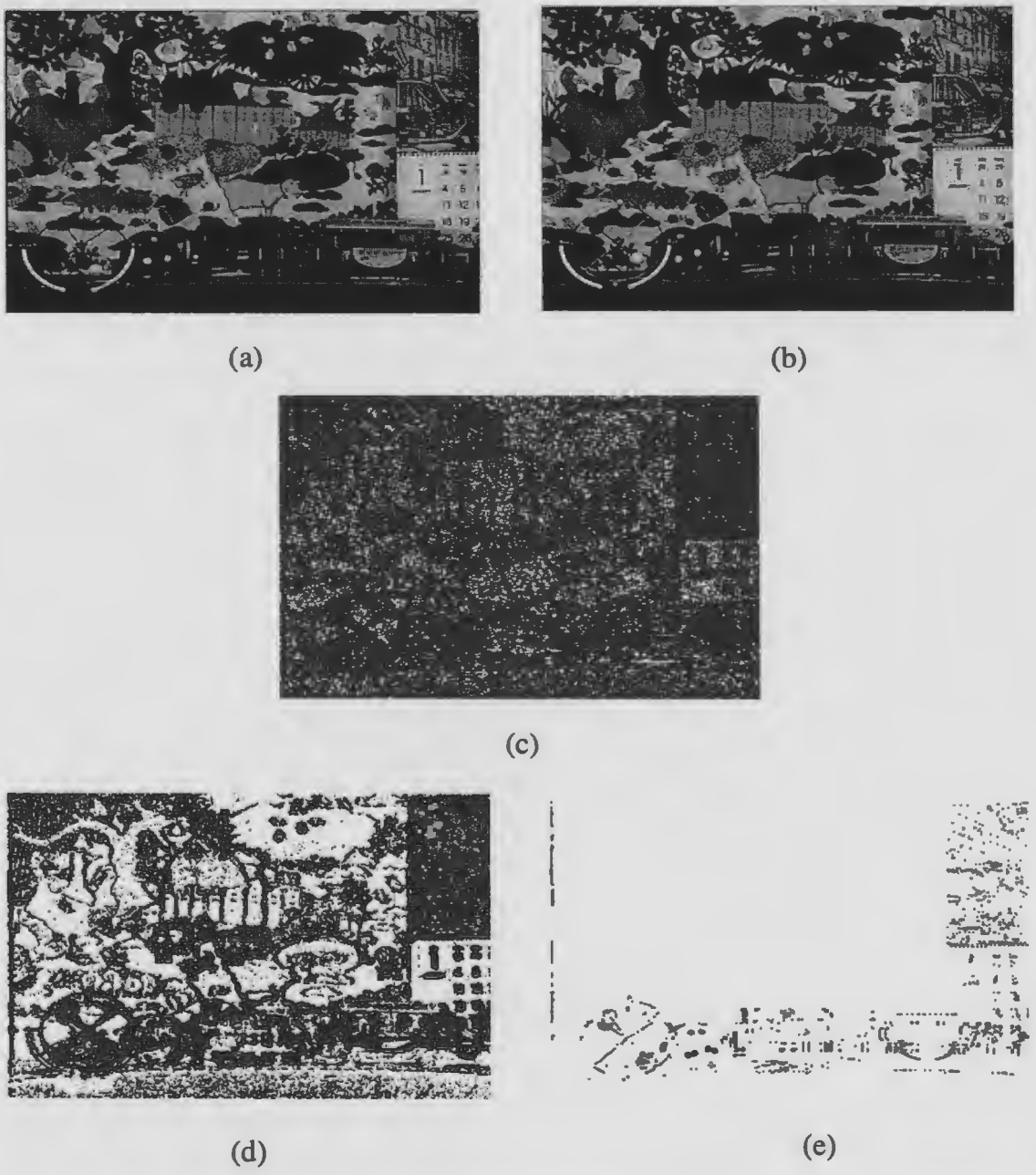


Figure 3.2: (a) Frame91 (b) Frame94 of the *mobile and calendar* sequence. (c) The difference image after first dominant motion vector calculation. (d) Matching errors thresholded by a low value. (e) Matching errors thresholded by a high value.

The appropriate threshold is calculated by finding the cross-correlation profile between the reference frame and the displaced frame using pixels with matching errors greater than thresholds from 0 to 255. The cross-correlation coefficient between two images X and Y is given by:

$$r = \frac{n\sum xy - \sum x \sum y}{\sqrt{\left[n\sum x^2 - (\sum x)^2\right]\left[n\sum y^2 - (\sum y)^2\right]}} \tag{3.2}$$

where **n** is the total number of pixels to be considered, **x** and **y** correspond to the intensity value at a certain point of images X and Y respectively, and the summations are over all pixels in the region being matched.

The cross-correlation profile has several curved segments that decrease exponentially. In general, points in each segment correspond to related objects. Those points having matching errors in the first segment are assumed to follow the calculated motion vector. We therefore calculate the threshold from the point of maximum slope between the first and second segments. **Figure 3.3(a)** and **(b)** shows the threshold calculation technique of two general types of profiles found in this case.

Figure 3.3(a) shows the natural correlation profile for true motion vectors. Here, the range from *0* to t_l corresponds to the same correlation level. Then the correlation coefficient decreases until the threshold reaches t_m. Near t_m the profile becomes flat as the effect of other objects being removed after a certain threshold becomes prominent. In

order to avoid the removal of other objects moving with different motion vectors, the maximum slope of the profile between *0* and t_m is found and the desired threshold, t_2 is obtained as follows:

$$t_2=t_1+(P-M)/\delta_{max} \tag{3.3}$$

where δ_{max} is the maximum gradient of the profile between *0* and t_m.

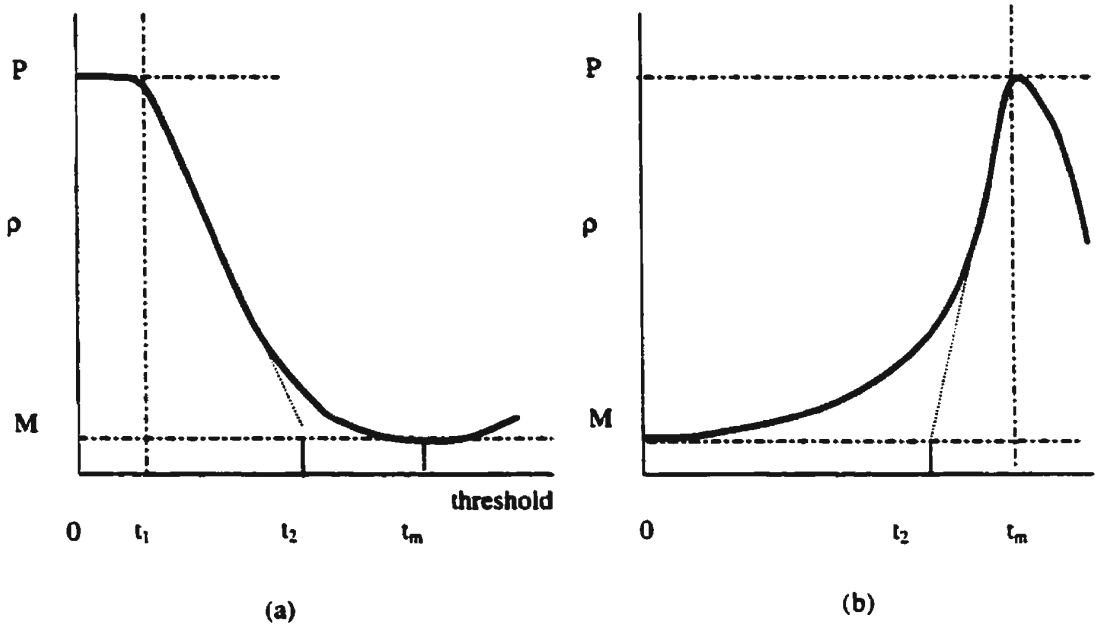


Figure 3.3: Cross-correlation coefficient (ρ) vs. Threshold profile. (a) for true motion vector (b) for false motion vector.

Figure 3.3(b) shows the correlation profile for false motion vectors. Generally, these types of profile result from false responses of the remainder of objects whose motion vectors have already been found. The motion vector for this type of profile is not considered and threshold t_2 is obtained as follows:

$$t_2 = t_m - (P - M)/\delta_{max} \quad (3.4)$$

This threshold prevents the unwanted effect of objects on the next level motion-vector calculation. The process of calculating motion vectors continues until the profile becomes flat which represents that the remaining part of the picture will not contribute any more valid motion vectors.

Figure 3.4(a) shows the thresholded image of the difference image shown in **Figure 3.2(c)** where the threshold is calculated using **Equation 3.3**.

(a) (b)

Figure 3.4: (a) Thresholded difference image after the first stage of motion vector calculation, (b) Remaining image segment after "opening" operation on the thresholded image.

3.4 Morphological Operation

In order to remove noise from one frame to another, *opening* is done on the thresholded image by a 3×3 square operator. In order to define opening some basic terms are defined as follows: [35]

Translation: The translation of a set of points A by $x=(x_1, x_2)$, denoted by $(A)_x$, is defined as

$$(A)_x = \{c \mid c = a + x, \quad for \quad a \in A\} \tag{3.5}$$

Reflection: The reflection of B, denoted by $\hat{B}$, is defined as

$$\hat{B} = \{x \mid x = -b, \quad for \quad b \in B\} \tag{3.6}$$

Dilation: The dilation of set A by the structuring element B, which is another set of points, denoted by $A \oplus B$ is defined as

$$A \oplus B = \{x \mid (\hat{B})_x \cap A \neq \varnothing\} \tag{3.7}$$

Thus the dilation process consists of obtaining the reflection of B about its origin and then shifting this reflection by x. It is the set of all displacements such that $\hat{B}$ and A overlap by at least one nonzero element.

Erosion: The erosion of set *A* by the structuring element *B* denoted by A ⊖ B is defined as

$$A \ominus B = \{x \mid (B)_x \subseteq A\} \tag{3.8}$$

Thus the erosion of A by B is the set of all points such that *B*, translated by *x*, is contained in *A*.

The opening of set *A* by structuring element *B*, denoted $A \circ B$, is defined as

$$A \circ B = (A \ominus B) \oplus B \tag{3.9}$$

This is the erosion of *A* by *B*, followed by a dilation of the result by *B*.

Figure 3.5 illustrates the opening operation [35] of a set *A* with a square-structuring element *B*.

The main attribute of *opening* is that it smoothes the contour of an image, breaks narrow isthmuses, and eliminates thin protrusions. It seems logical to apply *opening* as practical objects in general have smooth surfaces. **Figure 3.4(b)** shows the effect of an *opening* operation on a thresholded image in estimating motions between two frames in the *mobile and the calendar* sequence. The structure element, *B* for this case is a (3×3) square operator. It removes the noisy parts of the thresholded image, which are usually one pixel thick and this prevents those pixels from affecting the subsequent motion vector calculation.

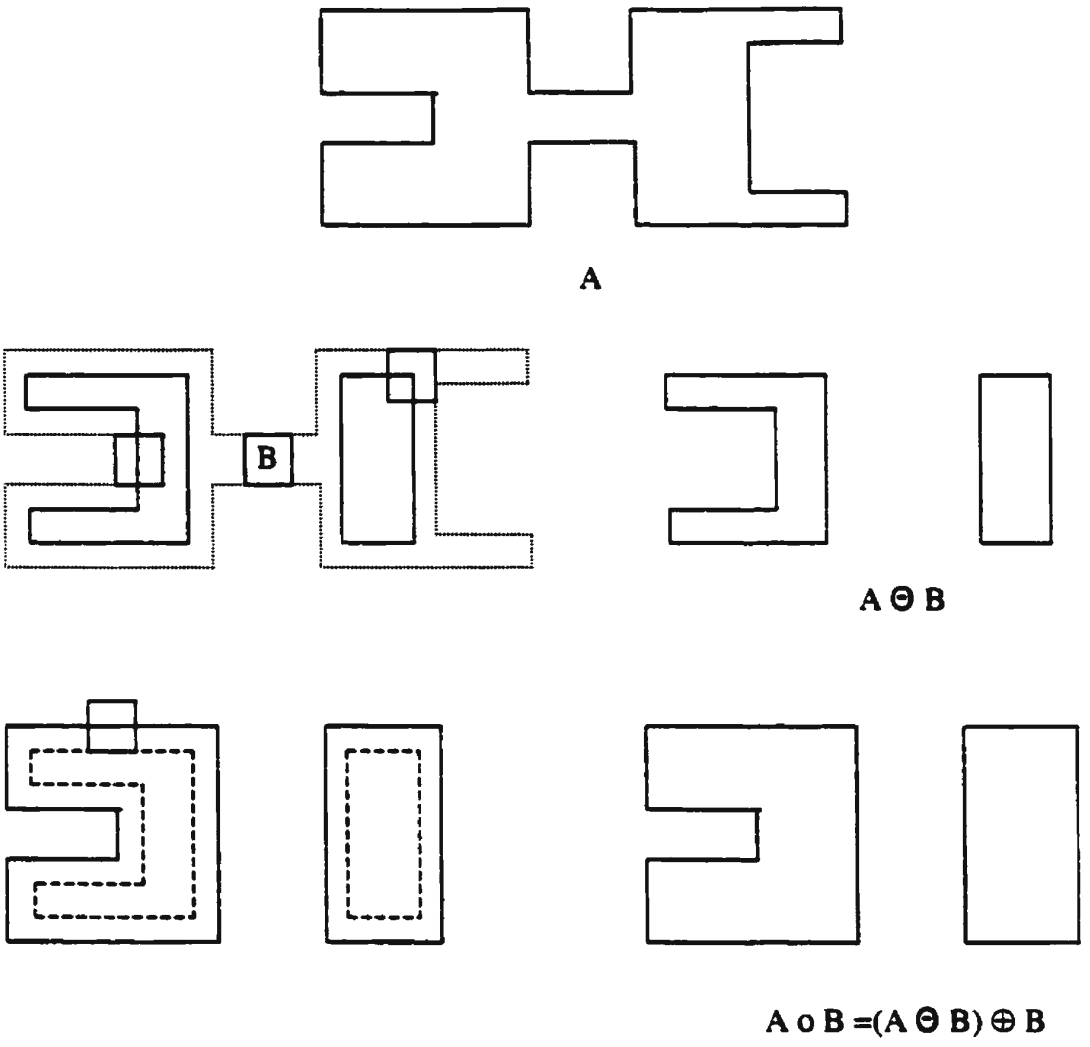


Figure 3.5: Illustration of opening operation.

3.5 Segmentation and Assignment of Motion Vectors

Due to the aperture problem and the impact of noise, local motion at a pixel is not well defined. So assignment of motion-vectors to individual pixels may result in wrong motion estimation of those pixels. In order to avoid this a group of similar pixels is considered to follow the same motion. This results in more robust motion vector assignment. In general,

pixels with uniform intensity correspond to the same rigid object and the pixels corresponding to the object undergo similar motion. So, having obtained the movement classes, we spatially segment frames in two steps, using a region-growing algorithm. Region growing is a procedure that groups pixels or sub-regions into larger regions. The procedure starts with a set of "seed" points and from these grows regions by appending to each seed point those neighboring pixels that have similar properties (such as gray level, texture, and color). Each seed grows by pixel addition and forms a cluster. In our segmentation technique, clusters having difference in mean intensity less than a threshold are merged together. To form a sufficiently large number of clusters, the threshold is set very low. On the other hand, the threshold should not be small enough to generate clusters with isolated pixels. **Figure 3.6** shows the segmentation of a frame in the *mobile and calendar* sequence.

In the first step, the segmentation is done over the second frame and then each segment is tested for each of the possible motion vectors. The motion vector giving the minimum matching error for the segment is considered to be the motion vector for that region. In order to tackle the problem of uncovered background, we perform segmentation on the first frame in the second step, which will act as the reference frame for motion compensation, and assign motion vectors to the segments according to the same procedure as discussed in the first step. The motion information in the first frame is then translated to correspond to the second frame. If this translated motion information does not match for any point in a segment of the second frame, the most probable translated motion vector in the segment, except the previousiy assigned motion vector for that

segment in the second frame, is then examined. For comparison, some critical pixels are not considered on the assumption that these pixels are exposed due to the movement of objects. At first one of the two motion vectors being tested is assigned to the segment and matching errors are found for both motion vectors without considering critical pixels. Then the same procedure is repeated with the other vector assigned to the segment. The vector giving the minimum cumulative matching error is kept for further processing. If cumulative matching errors for the vectors are same, the motion vector assigned in the first step is retained for the segment.

In order to consider the most likely motion vector, any segment giving matching error per pixel greater than some threshold is reconsidered. The threshold is set at t_m (**Figure 3.2(a)**), the worst possible threshold for a pixel in terms of matching error to move with that motion vector, at the stage when that motion vector was found. Then the motion vector for each reconsidered segment is calculated using gradient descent technique. To reduce the chance of assigning erroneous motion vectors that may result from the initial wrong search direction, the matching error of the test segment with the calculated motion vector at the later stage is compared to that with the previously assigned motion vector. Then the motion vector giving minimum matching error is kept on the assumption that the object shape is changed due to the motion.

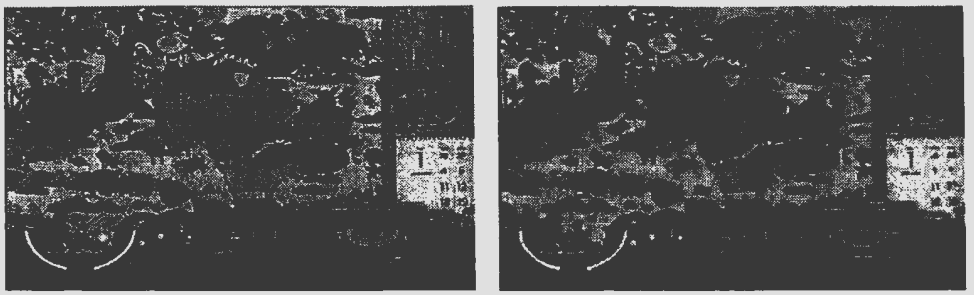


Figure 3.6: (a) Frame94 of the *mobile and calendar* sequence (b) Segmented image with intensity threshold = 10 and minimum number of pixels in a cluster = 6.

3.6 Post Processing Strategies

After the initial stage of motion vector assignment, we apply two steps of post processing. First: if there is any segment a pixel wide and smaller than a minimum object size (4×4), the segment is assumed to have the wrong motion vector and it is assumed that it has resulted from illumination change along an edge. In this case, we merge the segment to the surrounding which gives minimum matching error. Second: in order to improve the performance of chain coding without reducing the quality of the predicted image, we check smaller objects with their surrounding motion vector. If the matching error with its present vector is less than 1.1(10% tolerance) times that of its surrounding most likely motion vector, then the segment is merged with the surrounding. With these post-processing steps, the quality of the predicted image, the estimated frame constructed from the first frame of a sequence and the estimated motion vectors, does not degrade too much as compared to the original frame, but it simplifies the motion segmentation which facilitates efficient coding.

3.7 Prediction from the Motion Information

The prediction of the second frame from the reference frame using the motion information depends on finding critical regions. Critical regions are newly exposed groups of pixels and occur near motion boundaries. Generally, critical regions result from two situations. Let us consider an image that consists of objects composed in two layers. With both objects in motion, upper layer object movement will expose regions of the lower layer object depending on the relative movement of objects. In the second situation, lower layer object movement will also expose regions of lower layer object, which were previously covered by the upper layer object. The critical region depends on the relative movement of layers. **Figure 3.7** illustrates the critical region calculation for a general case of objects in motion.

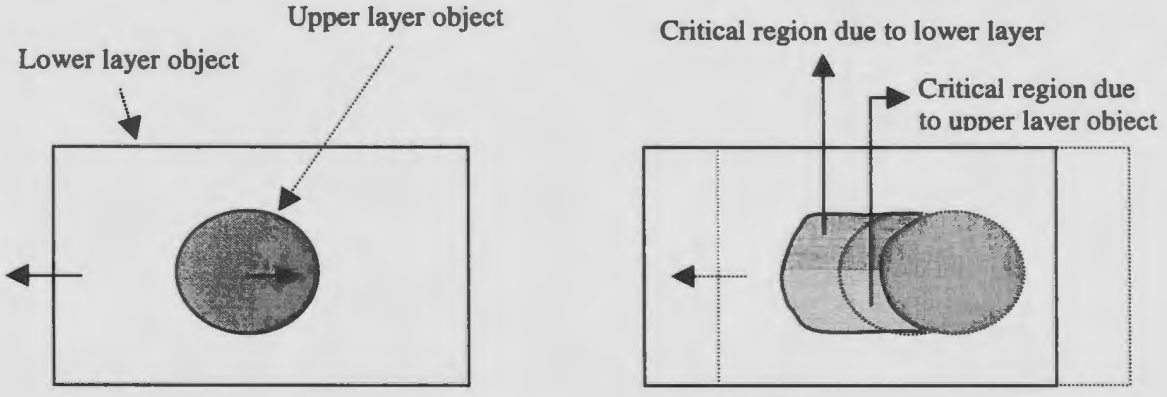


Figure 3.7: Critical region due to object movement in a two-layer image.

In our motion estimation technique, motion vectors that are calculated in later stages correspond to upper-layer objects. So the motion information is sufficient to calculate critical regions depending on the relative motion between objects.

When estimating the second frame from the first frame, pixels except in critical regions are estimated by translating pixels of the first frame using the motion information defined for pixels in the second frame. Critical regions are taken care of in a separate manner. Generally, critical regions are exposed pixels corresponding to lower layer objects. These are estimated by scanning through the image. When we scan through the image and find a critical pixel, then the relative motion of the moving object and the object where the critical region creates a hole is measured. If the relative motion is horizontal the intensity value to the left of the pixel is preserved. Then we continue scanning through columns and the intensity value to the right of the pixel where the continuation of critical region breaks is also preserved. With two values for these two pixels, the layer information of these two pixels is obtained and the critical region is substituted by the intensity value of the pixel corresponding to the lower level object. If both pixels correspond to the same motion vector the average of these two values is used for substitution. If the relative motion is vertical, the top and bottom pixel values are considered instead of left and right pixel values and scanning is done through rows. Otherwise, the average of the top and left pixel values is used instead of only the left pixel value and the average of the top and right pixel values is used instead of only the right pixel value as in the case of relative horizontal movement. The scanning in this case is done through columns. Once a critical region is taken care of we look for the next critical region and the process stops when all critical regions are covered.

3.8 Contour Coding

The conventional quadtree and binary tree contour-coding schemes introduced in **Chapter 2** are explained in detail in section 3.8.1 and 3.8.2 respectively. Then the new algorithm developed during my research in section 3.8.3 will be explained. A graphical comparison of these three coding schemes is shown at the end of this chapter.

3.8.1 Quadtree Coding

With a boundary image, there is little probability of a block being *all black* (i. e. all boundary). So a different quadtree scheme from the general scheme described in the previous section is proved to be efficient. This scheme divides the block to be coded into four sub-blocks. Then one of two possible codewords is assigned to each sub-block. Codeword **All_White (W)** is passed if the block has a uniform white region, otherwise **Split (B)** is passed. All sub-blocks coded as "B" are reconsidered for further analysis. The process continues until all sub-blocks to be analyzed are exhausted. Codeword mapping to binary for quadtree is as follows: B→1, W→0. **Figure 3.8(a)** illustrates the quadtree coding scheme.

3.8.2 Binary Tree Coding

The basic idea behind hierarchical coding schemes is to segment a picture into the largest possible uniform areas and to transmit a hierarchical representation of these areas. In the scheme proposed by **Cohen et al.**, they optimized the total number of cuts finding the effective cut direction for a block at a certain processing stage. If the block has at least a single black pixel it is said to have *activity*. The idea is to choose cut directions to

minimize the number of blocks with activity. This scheme required direction of cut and the structural information of the block to be passed to represent a block. For an m×n block, where both m and n are greater than 1, codeword **Split (B)** is passed as a structural representation, if the block has activity, otherwise **All_White (W)** is passed. In order to split a block, the direction information is passed before the structural codeword of sub-blocks. If the effective cut is in the same direction as the previous cut direction, codeword "U" is transmitted, otherwise "C" is transmitted. For single pixel thick blocks such as (m×1) and (1×m) blocks with m>1, an additional information **All_Black (A)** is required which efficiently codes a group of pixels. In these cases direction information is not required as there is only one possible cut direction. If the block is one pixel wide, then its height will be divided. On the other hand, if the block is one pixel high, then its width will be divided. In order to code a single pixel, "B" corresponds to black pixel and "W" corresponds to white pixel. Codeword mapping to binary for binary tree is as follows:

For (m×n) blocks:

UB→10, CB→11, W→0.

For (m×1) and (1×m) blocks (m>1):

B^{*}→0, W^{*}→10, A^{*}→11

For (1×1) block:

B^{**}→0, W^{**}→1.

Here the asterisks (*, **) define different classes of blocks. **Figure 3.8(b)** illustrates the binary tree coding scheme.

3.8.3 Proposed Coding Scheme

I now describe the new algorithm. The algorithm is a binary tree scheme. In contrast to earlier methods it adapts the cut position of a block to exploit inter-pixel redundancy and optimize the coding redundancy. Moreover, no single pixel is required to be coded, if it is inherently coded in the higher stages of the hierarchy. The efficiency of the hierarchical coding scheme depends on the assignment of a codeword to the largest possible uniform region. We have considered three types of blocks. First: (m×n) blocks where **m** and **n** are both greater than 1. Second: (m×1) and (1×m) blocks where **m>2** and Third: (1×2) and (2×1) blocks. Initially, all blocks are considered to be in "Primary Stage" and a specific cut direction is specified. Once the cut direction is fixed, we divide the block in the middle. If there is activity in both blocks, we send a codeword meaning split into two equal blocks. Both blocks will then be analyzed separately in the later stage. If there is activity in one block only, we put a codeword meaning second block has activity and a different codeword meaning first block has activity. Then we continue to subdivide the block with activity and check whether the block adjacent to that without activity has activity or not. If it has no activity then we put a continuation mark "1" and continue the checking process until there is activity in the adjacent sub-block. When there is an end of continuation before the maximum possible continuation steps, we put a terminating mark "0". Otherwise, maximum possible continuation will indicate the terminating mark. Here, the block with activity is analyzed further in later stages. If there is a continuation mark with blocks having activity in one sub-block only or there is activity in both the sub-blocks then the cut direction of the resultant sub-blocks for the next processing stage will

be opposite to the current cut direction. Otherwise, it will be the same as the current direction. If the cut direction is the same and the block has resulted from the larger block in the previous stage which had activity in one block only, then we know the activity location in one block at later stages. These blocks at this stage are termed as "Advanced Blocks". This utilizes the inter-pixel redundancy and codes subsequent blocks more efficiently. To pass equal information, "Advanced Blocks" require less bits than blocks in "Primary Stage". For (1×m) and (m×1) blocks the cut direction is always the same, always cutting the longer axis. The coding algorithm is the same as that of (m×n) block. But there is an additional codeword for the whole black block. (1×2) and (2×1) blocks are regarded in the same way as (1×m) and (m×1) blocks. Again, blocks at advanced stages are more efficiently coded than those at primary stages.

Codewords and their mapping to binary for the modified binary tree are as follows:

For (m×n) blocks,

	Primary Stage	Advanced Stage
Activity in both blocks (Split Equally)	A →0	D →0
Activity in the first block	B →10	E →1
Activity in the second block	C →11	F →1

For (m×1) and (1×m) blocks,

	Primary Stage	Advanced Stage
Activity in both blocks (Split Equally)	G →00	K →0
Activity in the first block	H →01	L →10
Activity in the second block	I →10	M →10
All_black block	J →11	N →11

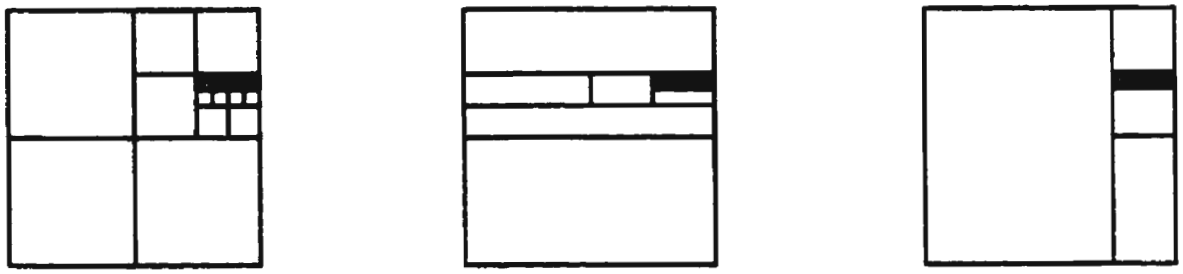
For (2×1) and (1×2) blocks,

	Primary Stage	Advanced Stage
Activity in both blocks	O →0	R→0
Activity in the first block	P →10	T→1
Activity in the second block	Q →11	T→1

For (m×n) block, codewords E and F represent activity in one sub-block only. As location of activity in one sub-block is known from the previous stage it is immaterial to know whether the first or the second sub-block is active. It is the same case for codewords L

and M in (1×m) or (m×1) blocks. As the location of activity in one pixel of a (2×1) and (1×2) block in advanced stage is known, Code "T" represents non-activity while code "R" represents activity in the remaining pixel.

Figure 3.8(c) illustrates the proposed coding scheme.



(a) (b) (c)

Code-string:

B	(UB)	
W B W W	(UB) W	VC10
W W W B	W (UB)	B0
B B W W	(CB) W	F0
B B W W B B W W	W (UB)	E1
	W (CB)	J
	$A^{\bullet}$ $W^{\bullet}$	

Figure 3.8: Different coding schemes; (a) Quadtree coding, (b) Binary tree coding, (c) the proposed coding scheme.

Number of bits required in (a) Quadtree = 21 bits (W→0, B→1) (b) Binary Tree = 21 bits [UB(Unchanged Direction and Split) →10, CB(Changed Direction and Split) →11, W→0, $B^{\bullet}$→0, $A^{\bullet}$→11, $W^{\bullet}$→10] (c) Proposed Coding Scheme = 14 bits [V(Vertical) →1, C→11, B→10, F(Advanced Block) →1, E(Advanced Block) →1, J(All Black) →11].

Let us consider the example shown in **Figure 3.8(c).** The block to be coded is a (16×16) pixel area which has activity in four pixels at row 5 and columns 13-16. The proposed algorithm needs the initial cut direction to be transmitted, which is either horizontal (H), or vertical (V) and it can be coded by one bit. With initial cut direction codeword **V**, the block will be cut in the vertical direction. The block is initially at the Primary stage. With cut position at the middle of the block it is found that there is activity in the second sub-block. So pseudocode **C** is transmitted. Then continuation process will start and the sub-block with activity will be halved in the vertical direction as direction information does not change in the continuation process. It is found that the first sub-subblock has no activity so continuation mark **1** is transmitted. Then the remaining sub-subblock is examined and halved. It is found that now both sub-subblocks have activity. So a terminating mark **0** is transmitted. There will be no more cuts in that direction and the previous cut position is the final cut location at this stage. Then the sub-block with activity will be passed for further coding. As it results from the continuation process or code **1** was transmitted the cut direction will be changed in the next stage which is the horizontal direction. In the next stage **B0** is transmitted, as no continuation is possible. Due to non-continuation the next cut direction is not changed and the sub-block with activity is in the advanced stage because we know that there is activity in the second sub-subblock, which blocked the continuation process in the previous stage. So, codeword **F0** is transmitted. The remaining block with activity is still in the advanced stage as the cut direction is not changed and it resulted from one non-active sub-block in the previous stage. At this stage **E1** is transmitted. Here no terminating mark **0** is required because

after transmitting **1** the remaining sub-block is reduced to one pixel wide that definitely has activity. So the continuation process reaches its maximum level which is an inherent terminating mark. Then the cut direction will be changed. But the remaining block is now a (1×4) block and it has activity in all pixels. So codeword **J** is transmitted.

The decoder will perform the reverse process. At first it receives **V** and it becomes ready to cut in the vertical direction. Then it receives **C** that means there is activity in the second sub-block. Then it looks for **1** or **0**. So it continues division of the second sub-block until a terminating mark is found. The terminating mark will locate the cut position and all pixels of the first sub-block are made non-active (in general represented by white pixels). As there was continuation so cut direction is changed and the process continues until all codewords are decoded. This results in a lossless coding scheme when codes are transmitted and decoded until the final processing stage.

The number of processing steps required by the proposed technique in this example is five, whereas the number of steps required by the quadtree and the binary tree algorithm are five and seven respectively. It gives an idea of the relative processing time of these three hierarchical coding schemes.

3.9 Summary

This chapter proposes an effective object based motion estimation algorithm and a new coding technique to code the motion information. The important contribution of the

motion estimation technique is to develop a quantitative measure of the threshold to segment out objects with true motion vectors. This quantitative measure together with the *opening* operation shows an effective way of segmenting out objects undergoing a similar motion (**Figure 3.4**). On the other hand, a simple but an efficient hierarchical coding scheme is developed and proposed to code the motion information. The potentiality of the new coding scheme is shown in **Figure 3.8**.

CHAPTER 4

RESULTS WITH MOTION ESTIMATION

This chapter consists of experimental results of the proposed motion estimation and segmentation algorithm with some practical images. After a brief discussion of results a relative performance of the algorithm with traditional block matching technique will be presented for comparison.

Throughout the thesis, certain video sequences and certain frames are selected to illustrate the proposed algorithm. These video sequences are representatives of the practical world videos and are considered as the test sequences in the performance analysis of motion estimation techniques. The *mobile and calendar* sequence consists of translational and rotational movement of objects, the *table tennis* sequence consists of mainly translational motion. On the other hand, the *flower garden* sequence consists of translational motion, zoom and pan.

4.1 Motion Estimation and Segmentation

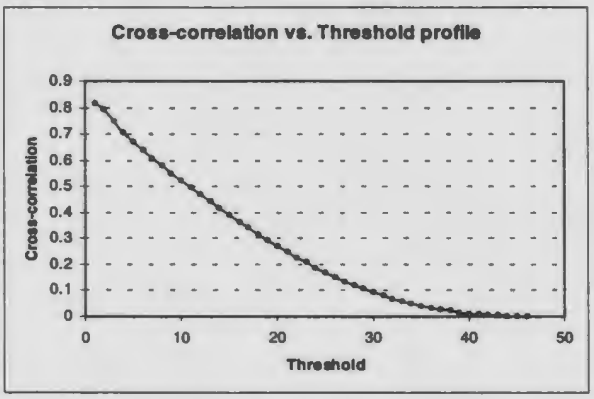
To demonstrate the performance of the proposed motion estimation and segmentation algorithm, let us consider Frame20 and Frame23 of the *mobile and calendar* video sequence. There are horizontal camera movement from the right to the left, train movement from the right to the left, a rolling ball in front of the train and a swinging toy. As the camera is moving from the right to the left, the background of the image moves

from the left to the right. At the first stage the region to be matched is the whole image and the motion vector (0, -2) corresponding to the camera motion is estimated. Here, the first component of the motion vector represents vertical motion and the second component corresponds to the horizontal motion. Negative values of the first component represent downward motion while positive values correspond to upward motion of objects. Negative values of the second component correspond to motion from the left to right while positive values represent motion from the right to the left. With the estimated motion vector the first frame is translated and the cross-correlation profile is obtained to calculate the appropriate threshold. **Figure 4.1** shows the threshold profile and the region to be matched at the second stage. At the second stage the motion vector (0, 1) is found which represents the moving train.

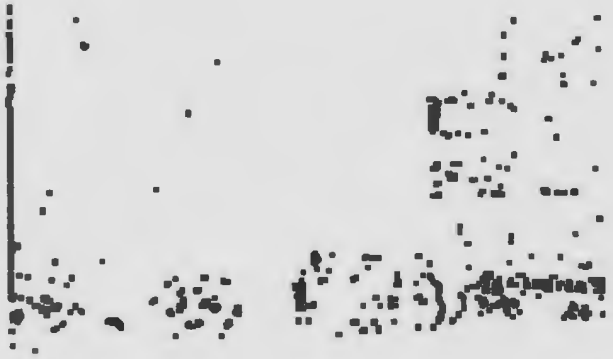
The process of motion vector calculation continues until the opened image does not consist of any pixel. With these two frames a total seven motion vectors are calculated. After the first two motion vector calculations the correlation between pixels in a region to be searched falls below 0.1. So a full search algorithm is applied to calculate successive motion vectors. Then both frames are segmented and calculated motion vectors are assigned to those segments. Motion vectors in the second frame are refined from the information of motion vectors in the first frame and two post processing techniques. **Figure 4.2(a) and (b)** show the segmentation of Frame20 and Frame23 respectively. Here the number of segments is very large compared to the total number of calculated vectors. Otherwise segments with two different motion vectors may merge together. This results in unreliable motion segmentation.

(a)

(b)



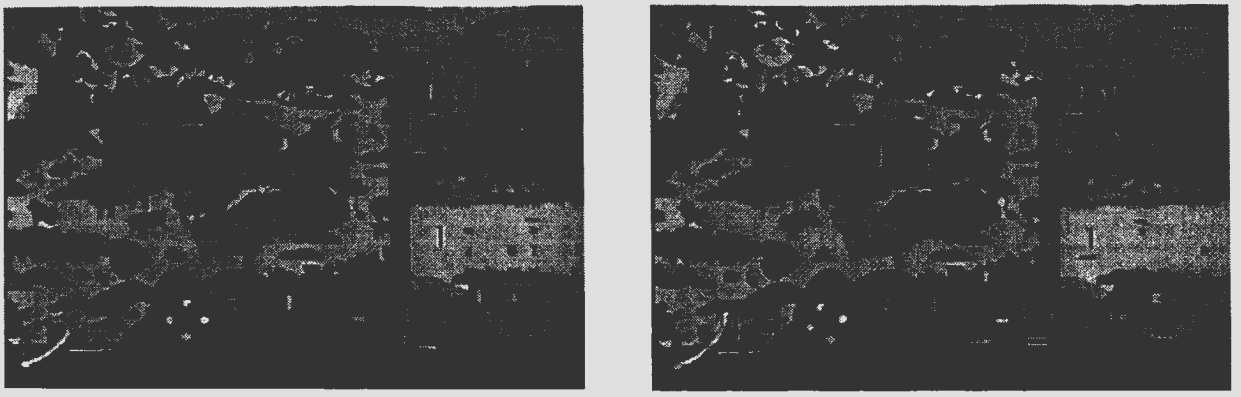
(c)



(d)

Figure 4.1: (a) Frame20 and (b) Frame23 of the *mobile and calendar* sequence, (c) Cross-correlation profile after the first stage of motion vector calculation [appropriate threshold = 18], (d) Opened image after thresholding which consists of the region for second stage motion vector calculation.

Movement of objects in the *mobile and calendar* sequence is very small. In order to test the algorithm with significant movement of objects, consecutive frames are used when testing the *table tennis* sequence, but frames separated by 3 are used when testing the *mobile and calendar* sequence.



(a) (b)

Figure 4.2: (a) Segmented image of Frame20 (652 segments) and (b) Segmented image of Frame23 (680 segments). Intensity threshold = 8 and minimum number of pixels in a segment = 10.

Figure 4.3(a)–(d) show motion vector boundaries and predicted pictures of the proposed algorithm and a standard block matching algorithm with (16×16) block size. It is observed that **Figure 4.3(b)** shows better object boundary extraction. There are some small differently moving objects in the calendar region. These are due to the temporary illumination changes in those regions. **Figure 4.3(c)** shows the blocky nature of motion estimation where the block boundaries do not match with object boundaries. **Figure 4.3(d) and (e)** show that the proposed technique predicts the image with higher PSNR (Peak Signal to Noise Ratio) than that of standard block matching technique. Here the PSNR is calculated for the recovered image compared to the original image and its unit is in dB (decibel). Moreover, from the visual point of view it can be observed that the shape of the ball is distorted with the block matching technique. Though the improvement of the proposed technique over the block matching technique in terms of PSNR is not

significant here, but the motion boundaries are significantly better with the proposed algorithm.

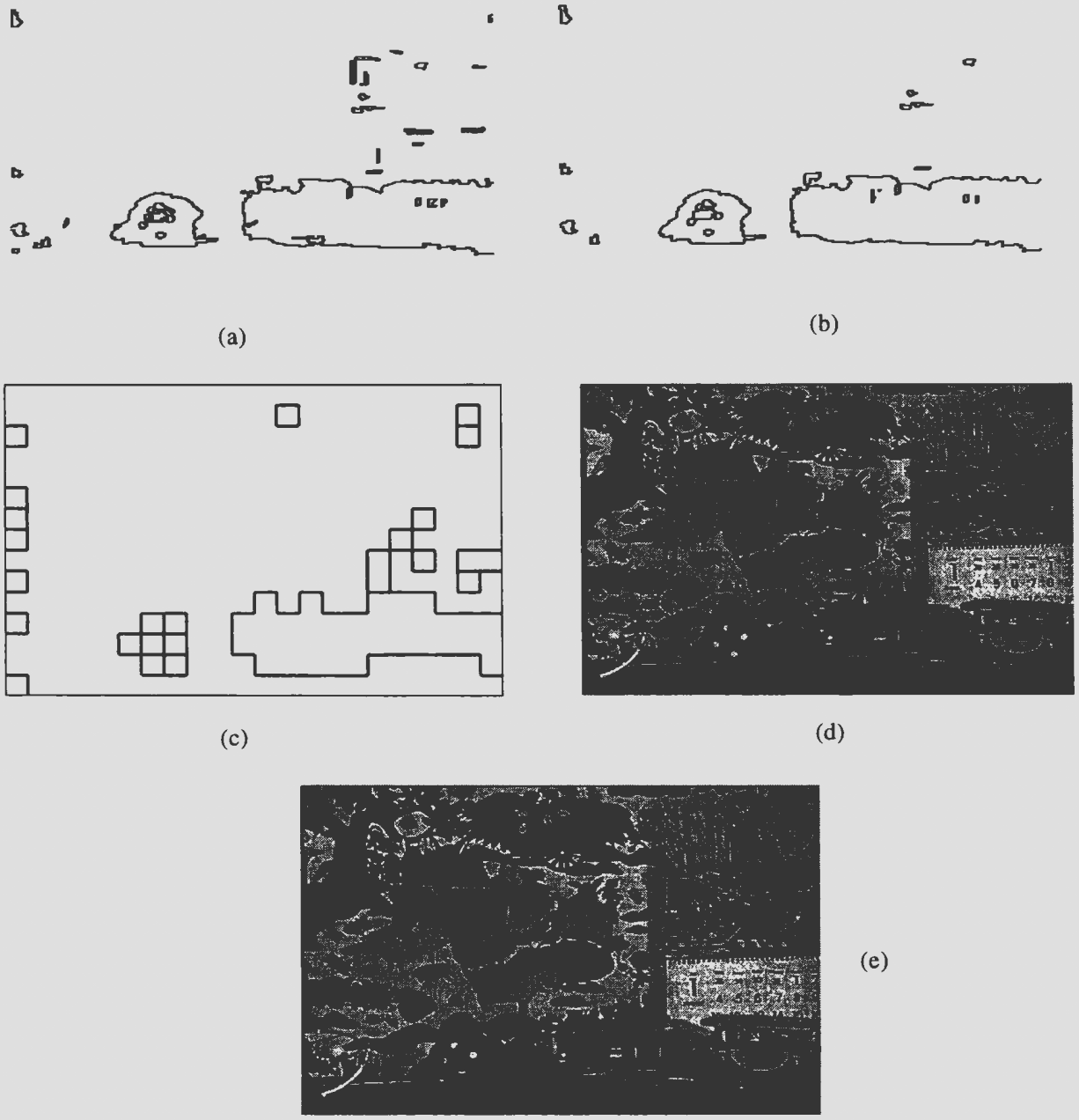
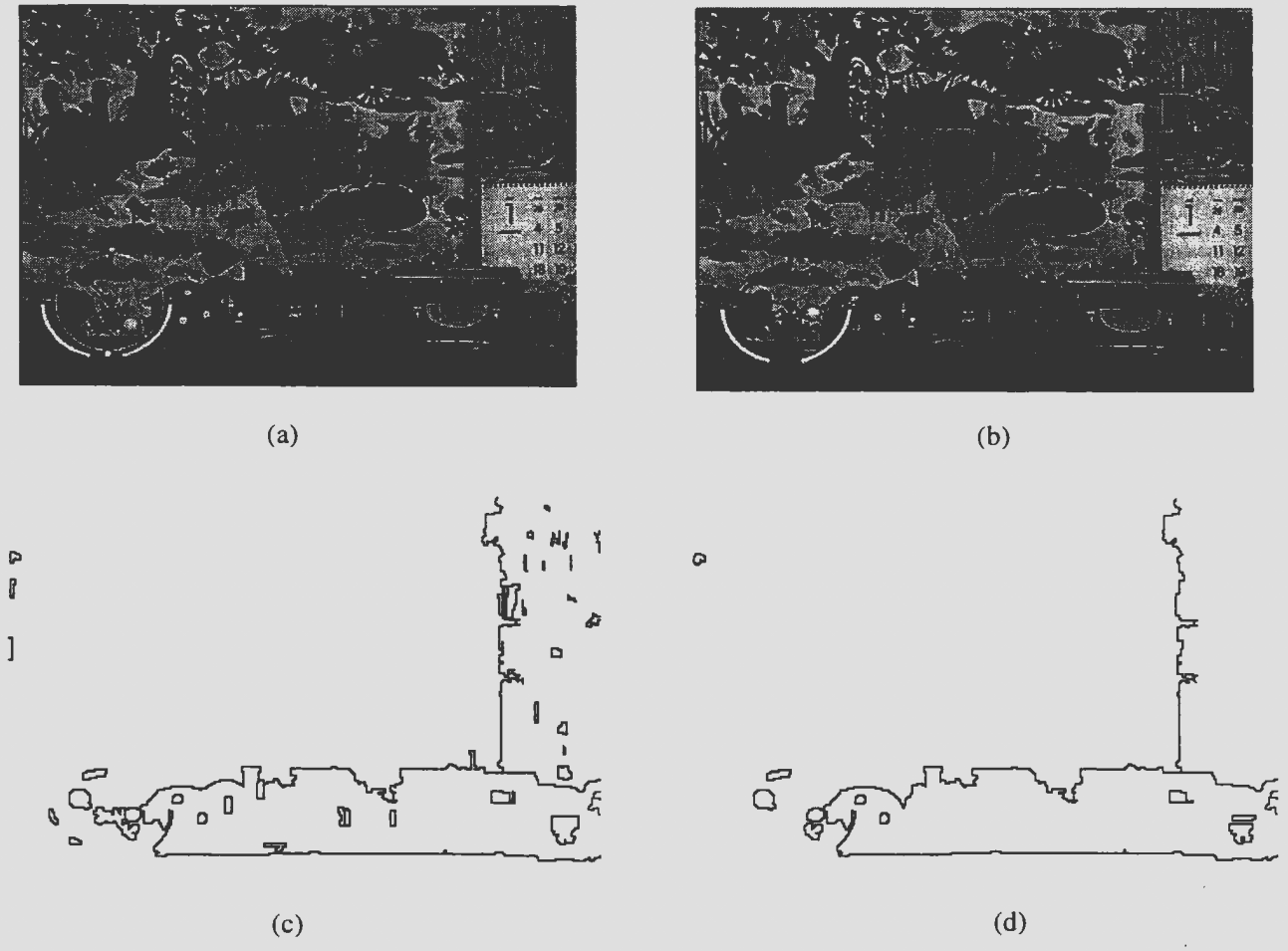


Figure 4.3: (a) Motion vector boundary before post processing, (b) Final motion vector boundary with the proposed technique, (c) Motion vector boundary with the block matching technique, (d) Predicted image with the proposed technique [PSNR=21.61], and (e) Predicted image with the block matching technique [PSNR=21.18].

Figure 4.4 and 4.5 show the application of the proposed motion estimation algorithm to segment motion information from Frame93 and Frame96 of the *mobile and calendar* sequence. Here the movement of objects is the same as that of the previous example except there is considerable vertical movement of the calendar.



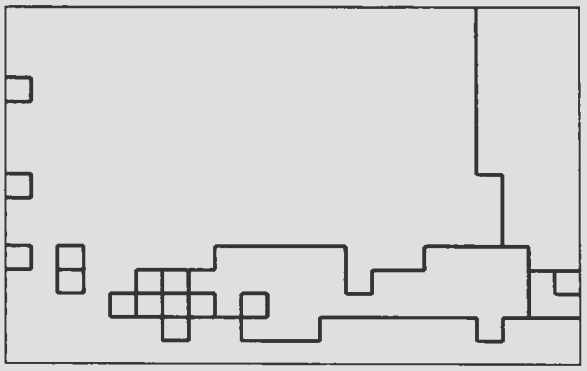
(a) (b)

(c) (d)

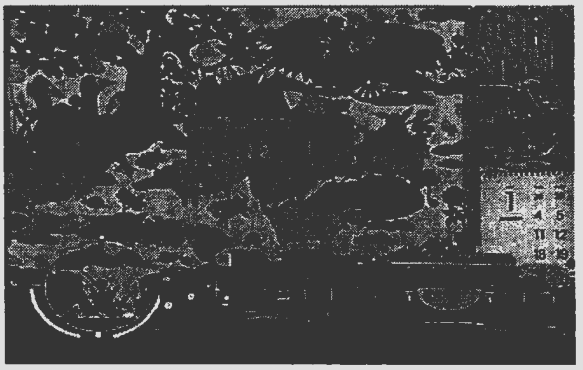
Figure 4.4: (a) Frame93 and (b) Frame96 of the *mobile and calendar* sequence, (c) Motion vector boundary before refinement, (d) Final motion vector boundary with the proposed technique.

Figure 4.5(a) shows that the block-matching algorithm finds the best possible motion vector for the corresponding block. Near the edge of the calendar, block boundaries

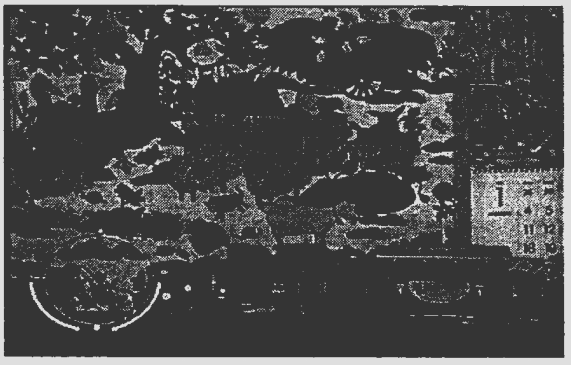
match with object boundaries. So, the calendar that is moving vertically is segmented out in a better way in block matching technique. But boundaries for the ball, train and the toy were not extracted well with this technique. On the other hand, **Figure 4.4(d)** shows better boundary extraction of objects moving with similar motion vectors. **Figure 4.5(c)** suggests that the proposed technique can predict the image with higher PSNR than that of standard block matching technique.



(a)



(b)



(c)

Figure 4.5: (a) Motion vector boundary with the block matching technique, (b) Predicted image with the block matching technique [PSNR=21.44], (c) Predicted image with the proposed technique [PSNR=22.47].

Figure 4.6 shows the performance of the proposed algorithm on the *table tennis* sequence. Here the ball and the hand with the bat are moving vertically. The block matching technique finds the local minimum for a block. This makes a lot of local motion vectors in the hand region. Moreover the ball gets four wrong motion vectors due to the effect of pixels from the background. The temporary illumination change on the table gets some incorrect motion vectors in the block matching technique as the block size is not enough to take care of this noisy effect. But the region based motion estimation technique can ignore this temporary change as it tries to find the global minimum of the whole object.

In general, Block matching takes the advantage of finding local minimum matching error. The sum of all local matching errors within an object is less than the matching error for the object with the global motion vector. This contributes to the quality of the predicted image using the block matching technique. But the proposed technique outperformed this technique due to better motion boundary extraction. The gain obtained from the corrected motion vector in the boundary region with the proposed technique compensated the gain obtained by the block matching technique with local matching error.

Figure 4.7 shows the performance comparison of a standard block matching technique (Block size 16×16) and the proposed technique for eight experiments on the mobile and calendar sequence. This comparison shows that the proposed technique outperforms the block matching technique.

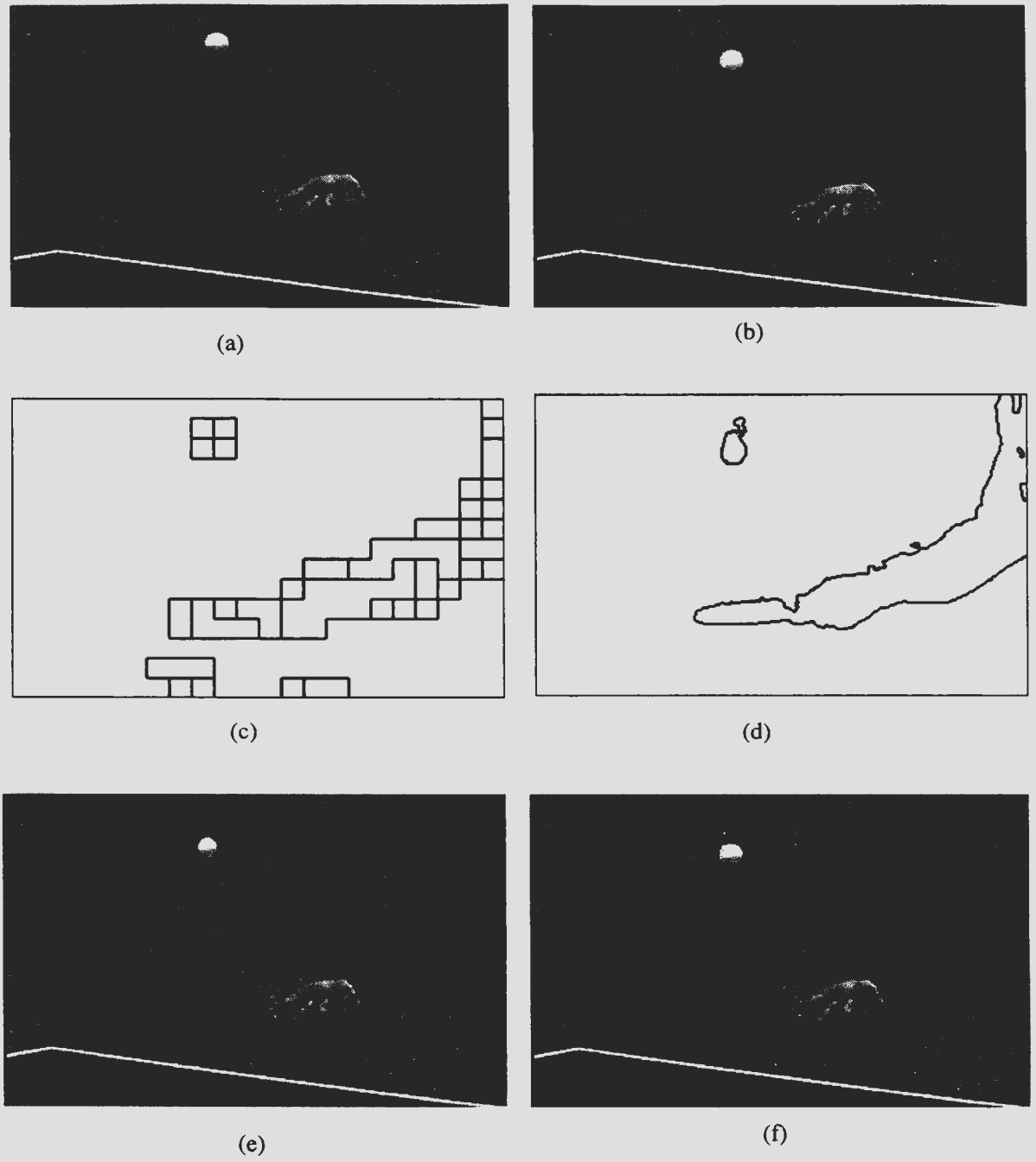


Figure 4.6: (a) Frame4 and (b) Frame7 of the *table tennis* sequence, (c) Motion vector boundary with the block matching technique, (d) Motion vector boundary with the proposed technique, (e) Predicted image with the block matching technique [PSNR=28.97], (f) Predicted image with the proposed technique [PSNR=29.53].

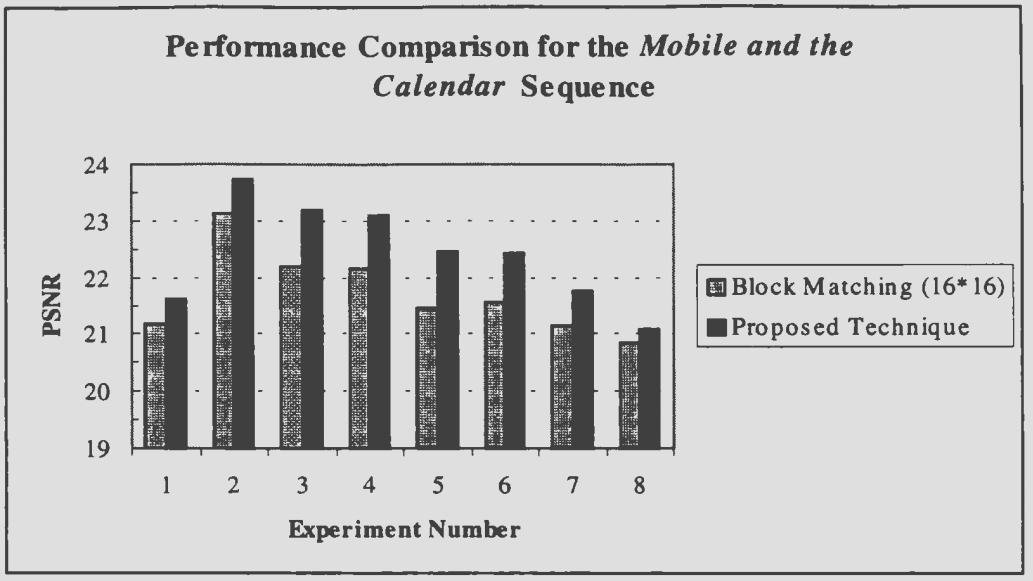


Figure 4.7: Performance comparison of the block matching technique and the proposed technique for the *mobile and calendar* sequence.

Table 4.1 shows the frames used in the experiment and the PSNR of the predicted image compared to the original image.

Table 4.1: Performance comparison for the *mobile and calendar* sequence.

Experiment No.	First Frame	Second Frame	PSNR (Block Match)	PSNR (Proposed)
1	Frame20	Frame23	21.18	21.61
2	Frame90	Frame93	23.14	23.73
3	Frame91	Frame94	22.19	23.20
4	Frame92	Frame95	22.14	23.11
5	Frame93	Frame96	21.44	22.47
6	Frame94	Frame97	21.54	22.42
7	Frame95	Frame98	21.14	21.75
8	Frame96	Frame99	20.83	21.08

Figure 4.8 and **Table 4.2** shows the performance comparison of a standard block matching technique (Block size 16×16) and the proposed technique for nine experiments on the *table tennis* sequence. These experiments show more PSNR improvement than those of the *mobile and the calendar* sequence do. It is because in most of these experiments there are nearly translating motions of the ball and the hand except some small rotational movement of the bat.

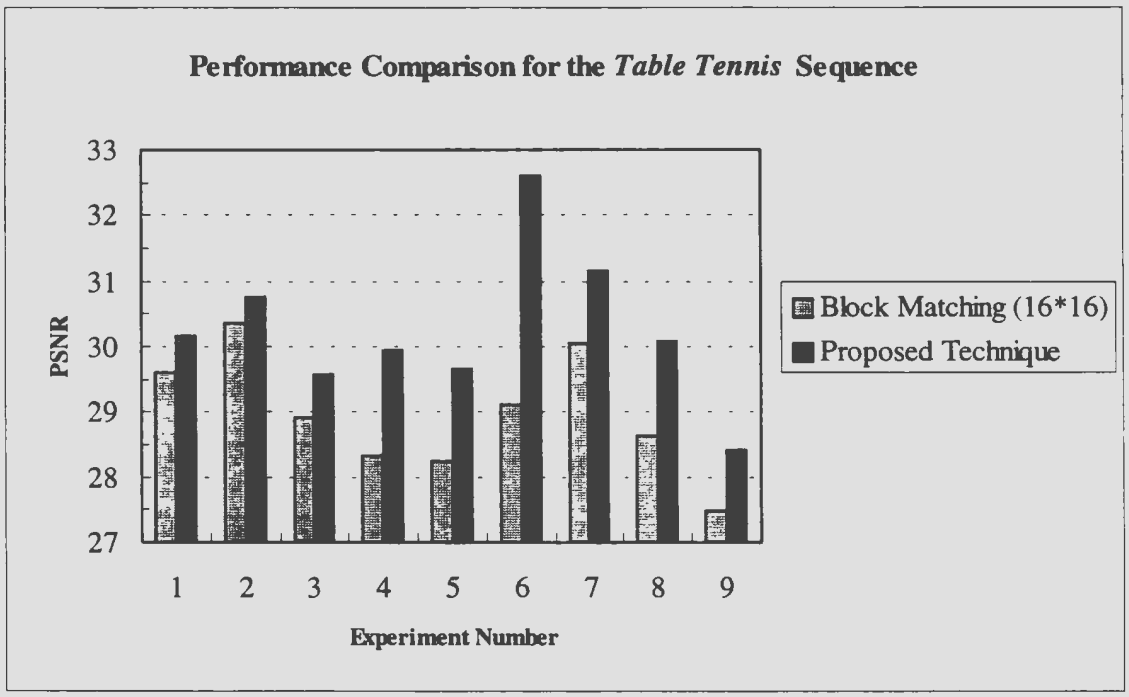


Figure 4.8: Performance comparison of the block matching technique and the proposed technique for the *table tennis* sequence.

Table 4.2 shows frames of the *table tennis* sequence used in the experiment and the PSNR of the predicted image compared to the original image.

Table 4.2: Performance comparison of the block matching technique and the proposed technique for the *table tennis* sequence.

Experiment No.	First Frame	Second Frame	PSNR (Block Match)	PSNR (Proposed)
1	Frame0	Frame1	29.6161	30.1705
2	Frame1	Frame2	30.3682	30.7565
3	Frame2	Frame3	28.9291	29.5884
4	Frame3	Frame4	28.3307	29.9463
5	Frame4	Frame5	28.2576	29.6724
6	Frame5	Frame6	29.1148	32.607
7	Frame6	Frame7	30.0704	31.1499
8	Frame7	Frame8	28.6361	30.0796
9	Frame8	Frame9	27.4897	28.4091

If the motion vector for the object is wrongly estimated or the boundary region near the moving object is not obtained due the translational motion vector then the advantage of the proposed technique is undermined. In that case the block matching technique will outperform the proposed technique.

Figure 4.9 shows the comparison of both techniques with two frames of the flower garden sequence. Here the second frame is a zoomed version of the first frame. There is *pan* as well. The quality of the predicted image using the proposed technique is worse than that of the block matching technique.

With the zooming effect a physical object is considered to have a large number of small moving regions. So the whole image is considered to a have large number of moving objects. More precisely each pixel in the object will have a different motion vector. This prevents the accurate motion estimation of the whole object. In addition, the layer

information of differently moving objects is also wrongly obtained. In the case of motion estimation using two frames shown in **Figure 4.9(a)** and **(b)** the motion vector corresponding to a portion of the tree is estimated before the background motion vector. This is because the background is separated into a number of separately moving objects due to the zooming effect whereas the tree close to the camera has less zooming effect. Moreover the proposed technique is based on translated motion vectors. Due to the wrongly estimated motion vector and object size variation with camera motion perpendicular to the image plane in addition to the horizontal movement, it is not possible to obtain a translated copy of the object in the reference frame. This degrades the quality of the predicted image. **Figure 4.9(d) and (f)** illustrates the poor quality of the proposed technique with camera movement perpendicular to the image plane. A possible solution to this problem is discussed in **Chapter 6**.

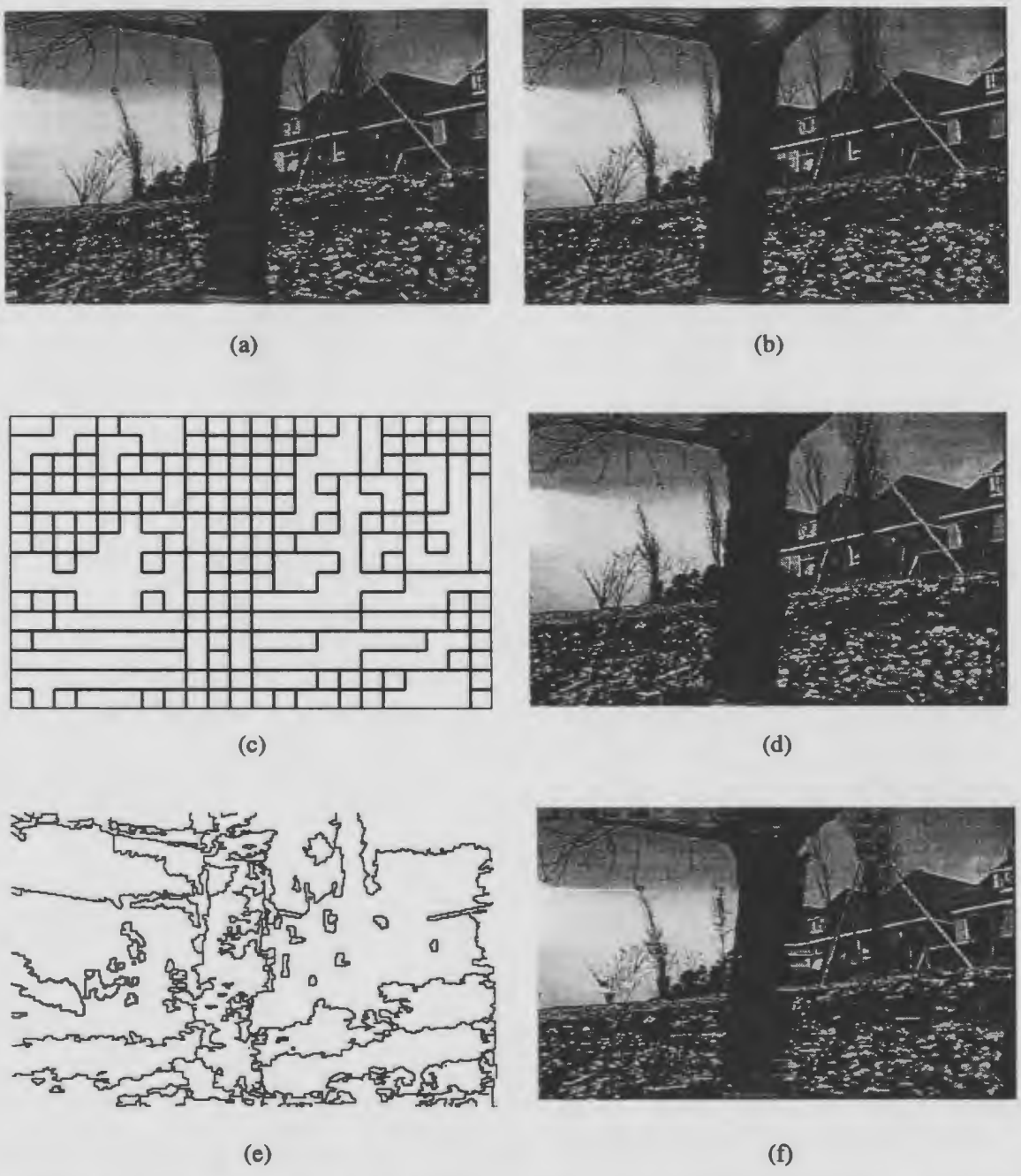


Figure 4.9: (a) Frame10 and (b) Frame13 of the *flower garden* sequence, (c) Motion vector boundary and (d) Predicted image with the block matching technique [PSNR=18.98], (e) Motion vector boundary and (f) Predicted image with the proposed technique [PSNR=16.98].

4.2 Summary

Results in this chapter show that the proposed motion estimation algorithm provides better quality recovered image with better motion boundary extraction in case of the *mobile and calendar* and the *table tennis* video sequence as compared to the conventional block matching technique. The proposed technique performs better on the *table tennis* sequence (**Figure 4.8**) than the *mobile and calendar* sequence (**Figure 4.7**) as the former sequence has mostly translational motion whereas the second sequence consists of translational and rotational movement of objects. The proposed technique did not perform well with the *flower garden* sequence (**Figure 4.9**). It is because the proposed algorithm performs best with video sequences having translational movement of objects whereas the *flower garden* sequence consists of zoom and pan.

One important aspect of the proposed motion estimation technique is that motion vectors of objects are estimated in sequence of their contribution to the total motion of objects in the sequence. The motion vector of the object, which contributes mostly to the total motion, is estimated at first and other motion vectors are estimated according to the decreasing contribution of corresponding objects.

CHAPTER 5

RESULTS WITH CODING SCHEMES

This chapter contains detailed results from testing the various options available under the new boundary coding scheme and its application to the motion information transmission.

5.1 Video Coding for Motion Compensation

The coded (compressed) information consists of two main parts. Firstly, the header information, which includes the total number of motion vectors, their values and the number of segmented regions with different motion vectors. Secondly, the information describing the structure of differently moving segmented regions.

5.1.1 Header Information

The header information is arranged to optimize the number of transmitted bits. Two strategies are followed to arrange the header information depending on the total number of motion vectors and total number of segmented regions with different motion vectors. In the first strategy, bits representing the total number of segmented regions are transmitted then the measured motion vectors *(i, j)* of sequentially obtained segmented regions are transmitted. In the second strategy, bits representing the total number of vectors are transmitted. Then the measured motion vectors *(i, j)* are transmitted followed by the motion index transmission of sequentially obtained segmented regions. The first

component of the motion vector represents the vertical motion whereas the second component represents the horizontal motion.

Let us consider that p bits are transmitted to represent a maximum number of 2^p motion vectors or segmented regions. a bits are required to represent each component of the motion vector and there are v total vectors and r total regions. **Figure 5.1** shows the difference between these two header arrangement strategies. Either of these two strategies is used and one bit is transmitted initially to indicate the selection of the technique.

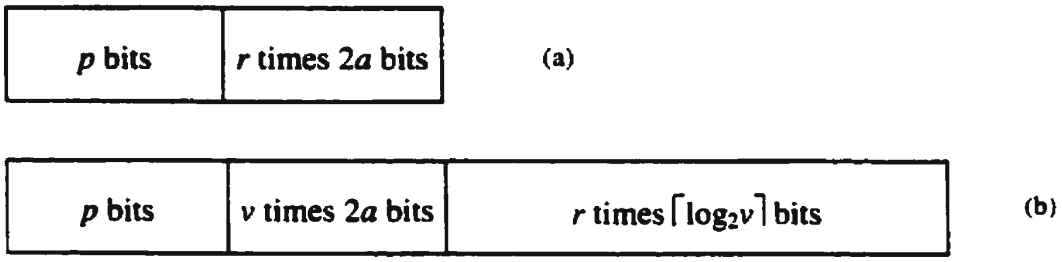


Figure 5.1: (a) First strategy and (b) Second strategy of header arrangement.

5.1.2 Structure Information

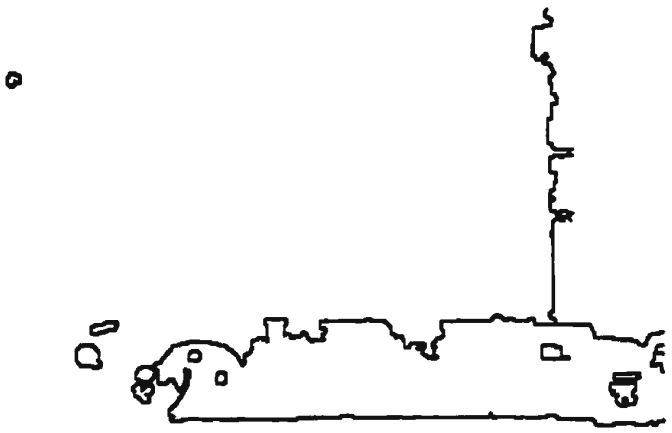
The structure information takes a large number of bits compared to the header information. Moreover, there is a probability of saving a huge number of bits with utilization of an efficient structural information coding algorithm. So, we focus on finding an effective technique to encode the structural information. In search of the algorithm, we have developed a modified binary tree, which minimizes the number of blocks with activity. Its performance is compared against quadtree and binary tree techniques.

Quadtree Technique

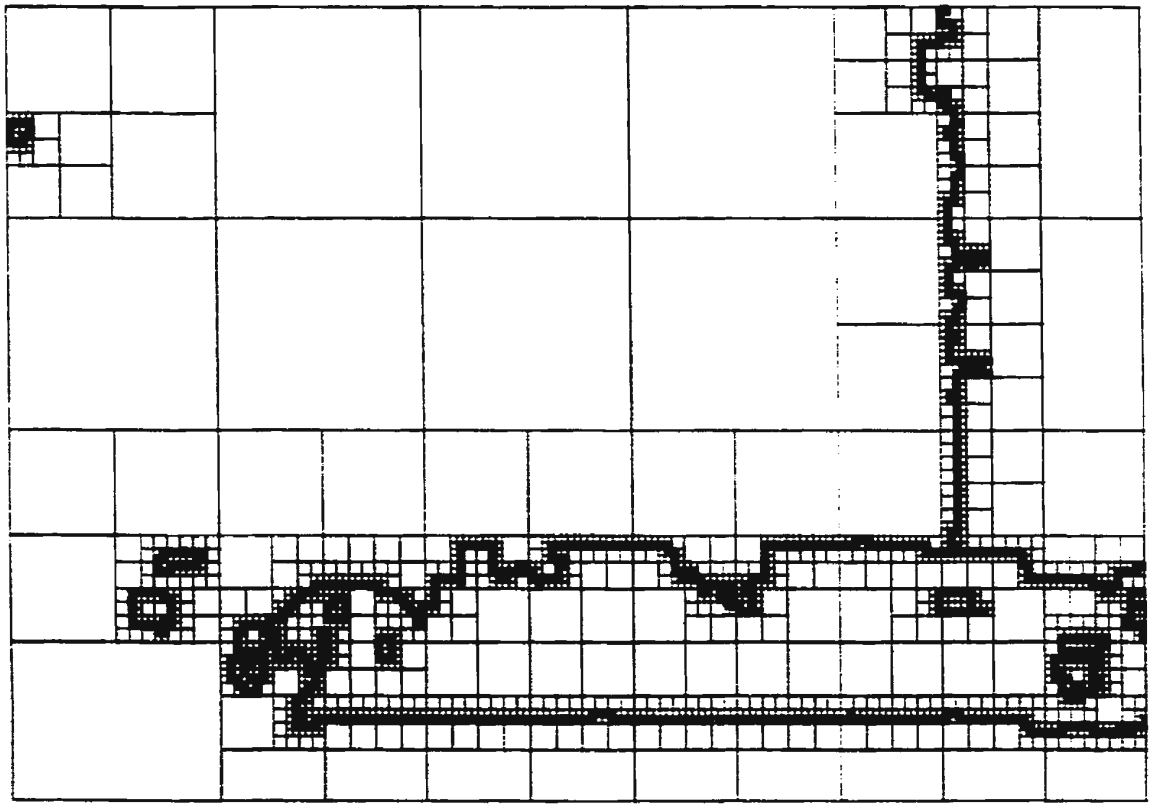
Let us consider the motion boundary of Frame93 and Frame96 of the *mobile and calendar* sequence. **Figure 5.2** shows the application of a quadtree algorithm. As the quadtree algorithm quarters each block with activity, the rectangular contour image is partitioned into a number of (64×64) blocks to obtain an unbiased quadtree technique. In order to obtain equal sized blocks the contour image is padded with pixels having no activity. **Figure 5.2(b)** shows a magnified contour image superimposed with grids where each block requires any of the two possible codewords, **W** indicating a complete white block and **B** otherwise.

Here 5576 bits are required to transmit the exact motion boundary information of the tested contour image of size 240 rows by 352 columns. If the processing steps stop before the maximum possible stage then the motion information can be transmitted with decreased resolution. In that case, a block having activity in at least one pixel will be considered to have activity in the whole block. This may reduce the quality of the predicted image.

Figure 5.3 shows the cumulative number of transmitted bits after each processing stage. The figure illustrates the high number of bits required to be transmitted at the finest level. This is because a block with activity at a specific stage results in four blocks at the next higher level. So a large number of bits can be saved if the information resolution and predicted image quality are sacrificed.



(a)



(b)

Figure 5.2: (a) Motion boundary of Frame93 and Frame96 of the *mobile and calendar* sequence, (b) Test image superimposed with grids to illustrate quadtree coding.

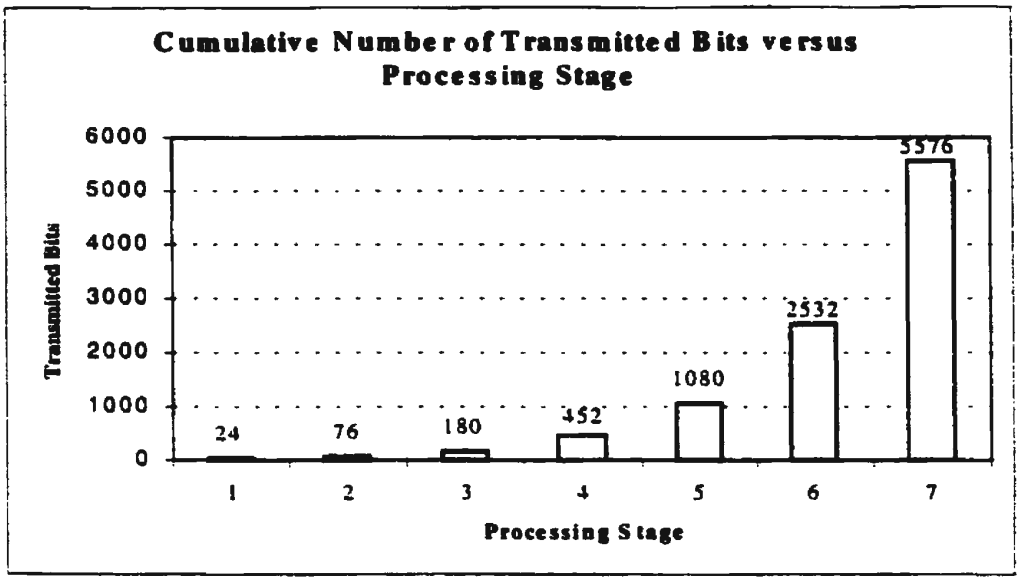


Figure 5.3: Variation of transmitted bits with processing stage using quadtree coding scheme.

Binary Tree Coding

Figure 5.4 illustrates the result using the binary tree algorithm suggested by **Cohen et al.** [34] with the same contour image. Here the total number of transmitted bits is 5515 bits, which shows slightly better performance than the quadtree coding. The advantage comes from the minimization of blocks with activity using effective cut direction. But there is no substantial amount of savings in terms of bits with this complex algorithm relative to the simple quadtree coding.

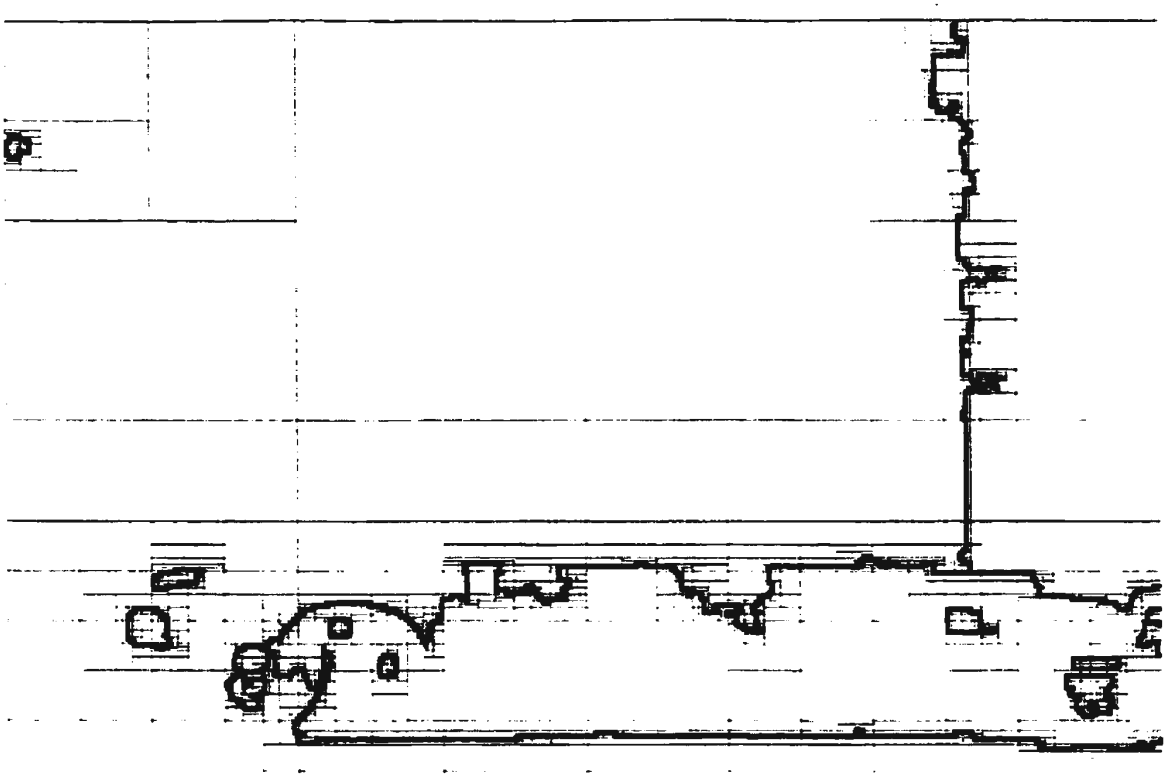


Figure 5.4: Application of the binary tree coding on a motion boundary image.

Proposed Coding Scheme

Figure 5.5 shows the application of the proposed coding algorithm on the contour image which is actually the motion boundary between frame93 and frame96 of the *mobile and calendar* sequence. This proposed scheme requires 4351 bits, which is considerably less than that required by the quadtree and conventional binary tree coding schemes. There is about 22% improvement over traditional quadtree and binary algorithm in terms of number of bits.

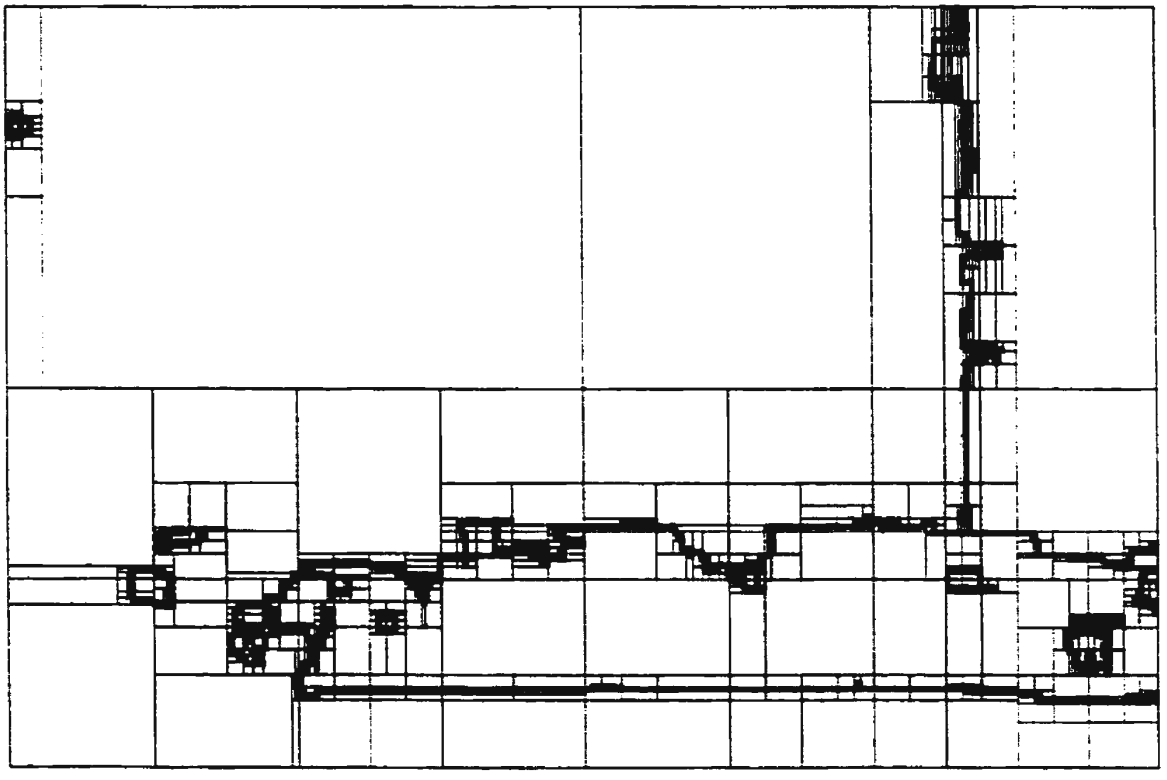


Figure 5.5: Application of the proposed coding scheme on the motion boundary image using Frame93 and Frame96 of *the mobile and calendar* sequence.

Figure 5.6 shows the application of the proposed coding algorithm on the motion boundary between frame5 and frame6 of the *table tennis* sequence. The proposed algorithm required 3393 bits whereas quadtree and conventional binary tree scheme required 4178 and 4566 bits respectively.

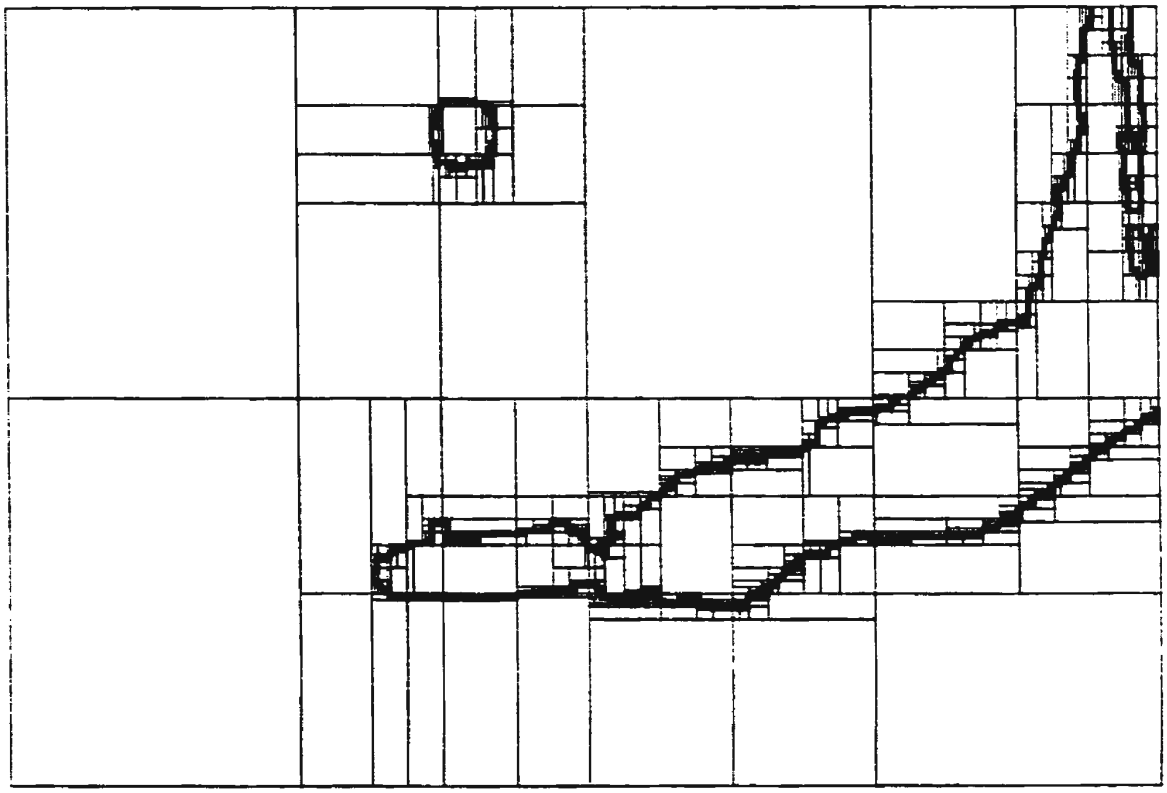


Figure 5.6: Application of the proposed coding scheme on the motion boundary image using Frame5 and Frame6 of the *table tennis* sequence.

Figure 5.7 shows the performance comparison of the quadtree, conventional binary tree, the proposed coding scheme and the READ coding algorithm [30] used in the Tagged Image File Format (TIFF) as well as in G4 fax. The comparison is performed based on

the application of some practical motion boundary images (**Table 5.1**). The graph shows better performance of the proposed algorithm than quadtree, traditional binary tree coding and G4 fax coding for all boundary images.

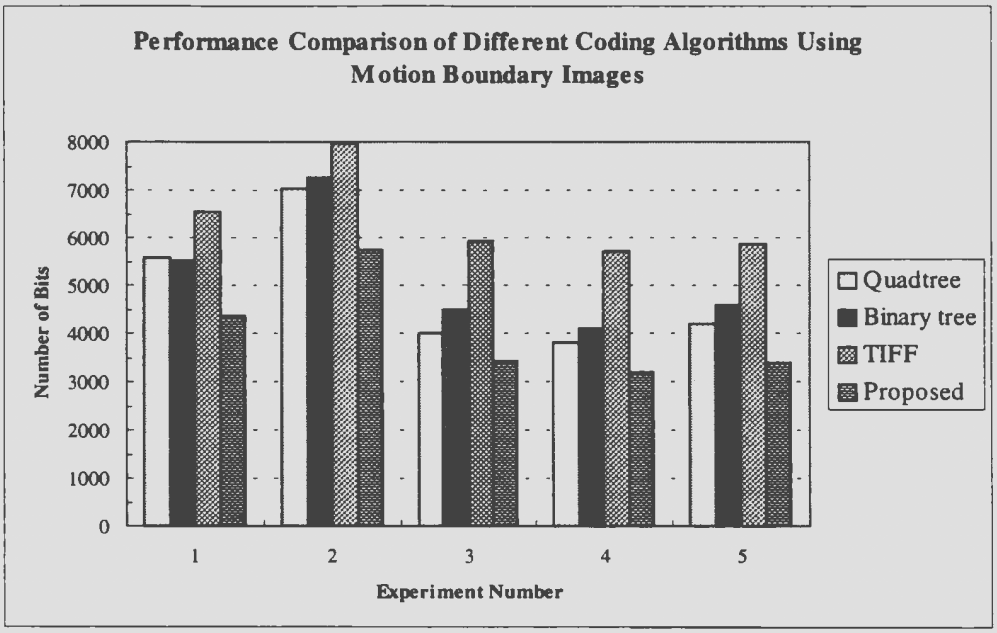


Figure 5.7: Performance comparison of different coding algorithms using motion boundary images.

Table 5.1: Performance comparison of the Block matching Technique and the proposed technique for the *mobile and calendar* sequence.

1st Frame	2nd Frame	Transmitted Bits			
		Quadtree	Binary tree	TIFF	Proposed
Mobile93	Mobile96	5576	5515	6560	4351
Mobile96	Mobile99	7042	7249	7952	5737
Tennis2	Tennis3	3996	4486	5920	3420
Tennis3	Tennis4	3798	4110	5712	3198
Tennis5	Tennis6	4178	4566	5856	3393

The number of bits per active pixel for experiment 1-5 is 2.79, 2.83, 3.13, 3.14, and 3.11 respectively. Chain coding scheme requires 3 bits plus bits that are required to transmit the branch information per active pixel.

Performance of different coding schemes is compared for several thicker textual images. **Figure 5.8** shows that the performance of the proposed technique is better than that with the quadtree and the conventional binary tree but worse than the READ coding scheme in case of textual images. Images used in this experiment are shown in **Appendix A**. This is because the proposed technique is developed to encode one-pixel thick binary images. This scheme does not have any codeword for all black pixels in a block of size larger than (2×2). So it continues to split a large block into smaller blocks until a codeword is found for the block. This requires a large number of bits to represent a block, which has more than one-pixel thickness activity.

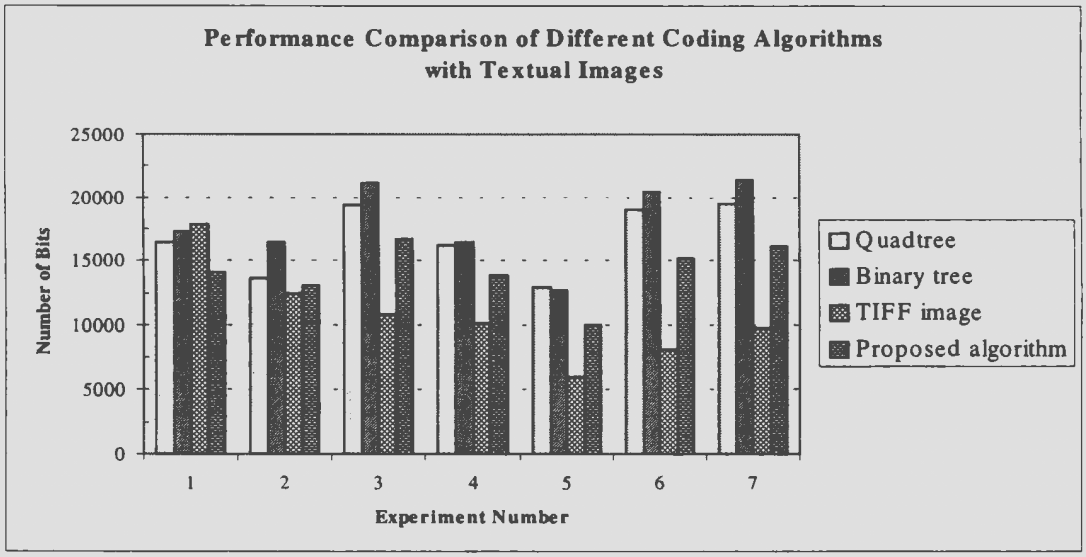


Figure 5.8: Performance comparison of different coding algorithms using textual images.

Figure 5.9 shows the application of the proposed algorithm on the textual image used in the fifth experiment (**Figure A.1(5)**). This shows the inefficiency of the proposed algorithm in case of thicker textual images. Such blocks do not occur in boundary images, but this is a limitation of the method when applied to more general 'two-level' images such as documents.

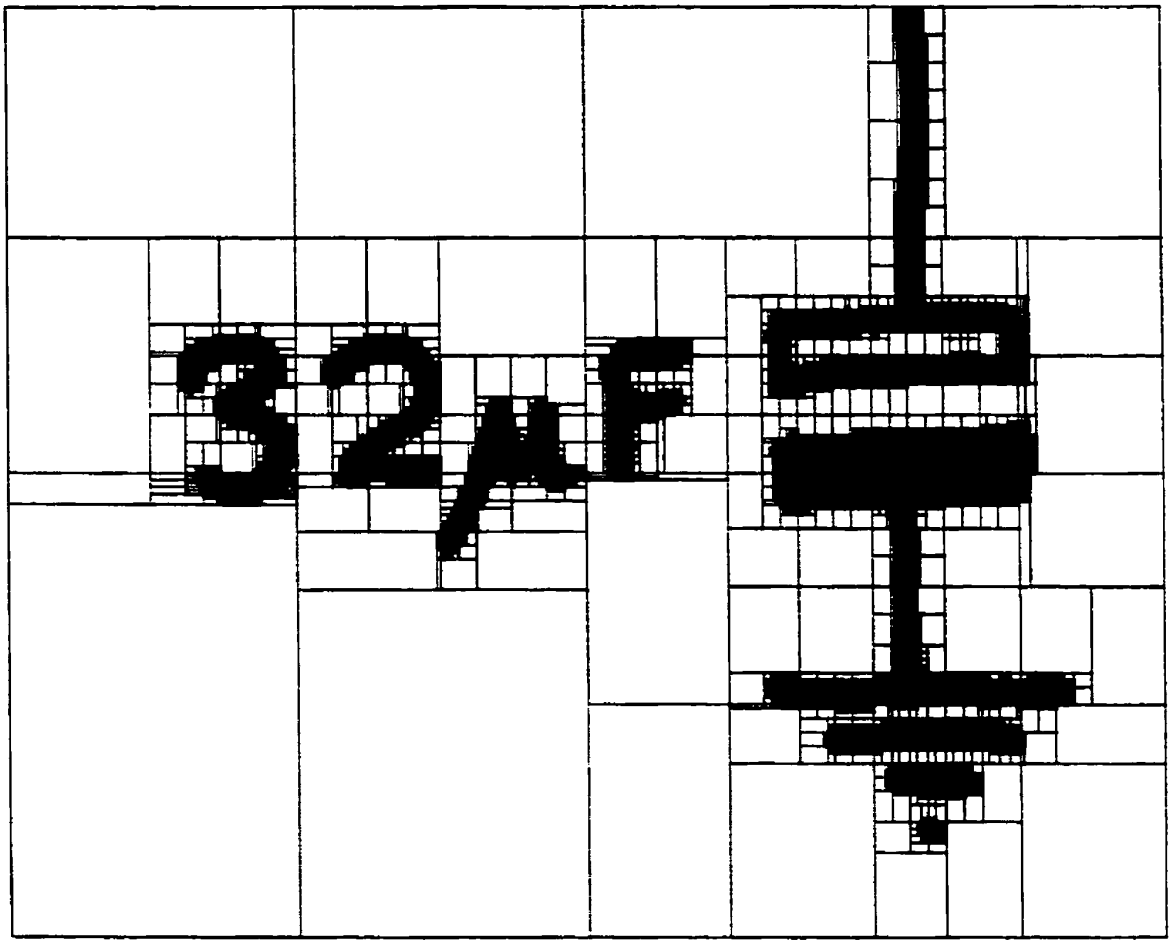


Figure 5.9: Application of the proposed coding scheme on the image used in the fifth experiment with textual images.

5.2 Variation of Predicted Image Quality with Variable Bit Rate

This section will illustrate the variation of the predicted image quality with the resolution of the transmitted motion boundary information. The variation of the resolution corresponds to the number of transmitted bits. If the motion boundary image is coded to its finest level then it will require more bits than the number of bits if coding is stopped before reaching its maximum level.

5.2.1 Estimation of the Non-Coded Regions

If the transmitter stops coding before reaching the finest level of blocks to be coded then there will be some blocks in uncoded condition. These blocks need to be estimated from the coded blocks. Here, the problem is dealt with a simple merging process. Let us consider that white regions in the decoded image correspond to coded regions and black regions correspond to uncoded regions. At first we segment out black and white regions. There will be two situations with the black segments. First: A black segment may have only one adjacent white segment, which is defined as an isolated black segment. Second: A black segment may have more than one adjacent white region. In the first case, we measure the total number of pixels in the black region. Then the white region adjacent to this black region continues merging adjacent black pixels to its region until half of the total black pixels are merged. This will result in a half of the number of pixels to be merged to the adjacent white region and the remaining half will result in a new white segment. In the second case, the merging process will be rotated among candidate white regions. To deal with the situation white regions are sequentially arranged based on the

order of finding the region while scanning the image. With a number of candidate white regions around a black segment, the first white segment will agglomerate its adjacent black pixels and if it leaves some black pixels unmerged then the merging process will be handed over to the next white region. The process continues until all black pixels are merged. This will follow a cyclic handover process. It is logical to merge according to this technique as the transmitter follows the same procedure while generating the boundary image. With many different motions around a pixel, the pixel is considered as a boundary point if the pixel corresponds to a motion region that comes before other candidate regions while scanning the image.

The transmitter needs to transmit a number of motion vectors equal to the sum of the number of white regions and the number of isolated black regions. During transmission the encoder will transmit one motion vector for each coded region according to the scanning process. Then a motion vector for each isolated black region is transmitted, which is the most probable motion vector within the possible newly created region. All these motion vectors are used to form the header information. In this case, the first strategy of arranging header messages is used, **Section 5.1.1**. Bits representing the total number of regions are transmitted at first. Then the motion vectors of corresponding regions are transmitted. After this the information of the structure of the motion boundary is transmitted.

Figure 5.10 shows the decoded region and estimation of the uncoded region from the decoded region of the motion boundary image between Frame93 and Frame96 of the

mobile and calendar sequence. Results are obtained when the transmitter will stop transmitting after a certain number of processing stages.

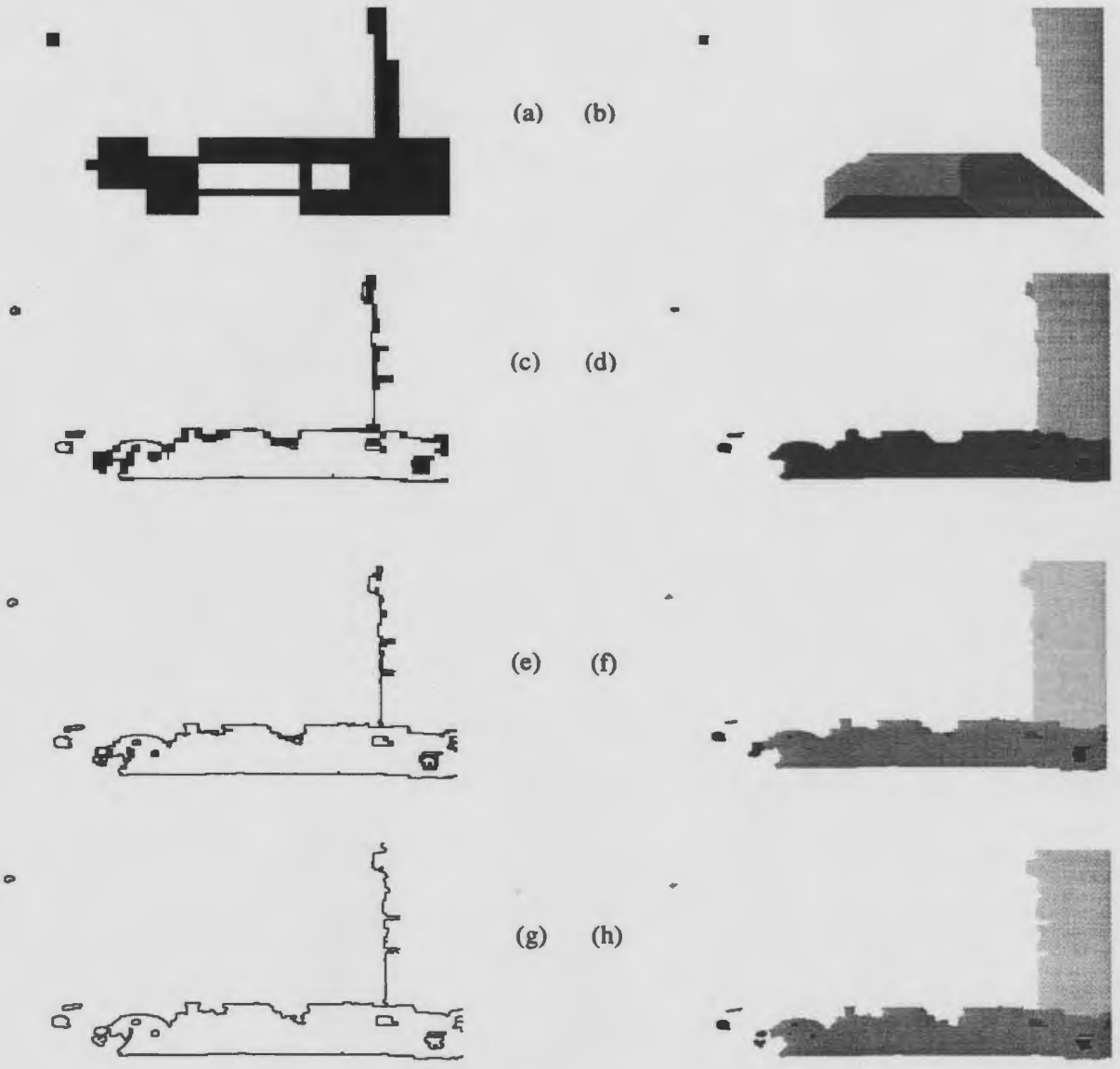


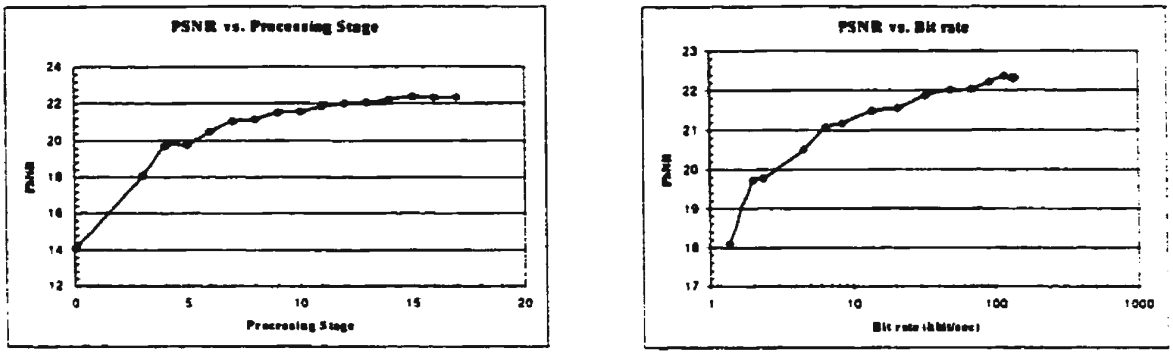
Figure 5.10: (a, c, e, g) Transmitted and decoded motion boundary image and (b, d, f, h) Recovered image after merging of black regions after processing stage 7, 12, 14, 17.

Table 5.2 shows the number of bits required to code the motion boundary information, total number of motion vectors to be transmitted, PSNR of the predicted image, and bit rate assuming 30 frames per second with respect to the processing stage used by the transmitter. Here Frame93 and Frame96 of the *mobile and calendar* sequence are considered. Let us assume that 6 bits (maximum 64 regions) are required to transmit the information about the total number of motion vectors to be transmitted and 6 bits (±32 pixel movement) are required to transmit each component of the motion vector. The PSNR between Frame93 and Frame96 is 14.0245, which is the PSNR of the recovered image without any transmission of bits.

Table 5.2: Variation of quality of the predicted image with bit rate using two frames of the *mobile and the calendar* sequence.

Stage No.	Transmitted bits for the boundary (A)	Total regions (B)	PSNR	Bit Rate (6+12×B+A) ×30
3	15	2	18.0919	1350
4	24	3	19.7208	1980
5	35	3	19.7739	2310
6	85	5	20.5075	4530
7	137	6	21.0653	6450
8	228	4	21.164	8460
9	366	7	21.4854	13680
10	598	7	21.5713	20640
11	961	9	21.8884	32250
12	1478	10	22.0345	48180
13	2100	14	22.0491	68220
14	2809	20	22.2082	91650
15	3612	23	22.366	116820
16	4204	19	22.3252	133140
17	4351	17	22.3158	136830

Figure 5.11 shows the variation of PSNR with respect to the processing stage and bit rate.



(a) (b)

Figure 5.11: (a) PSNR vs. Processing Stage and (b) PSNR vs. Bit Rate of the predicted image using Frame93 and Frame96 of the *mobile and calendar* sequence.

Some interesting analysis can be obtained from **Figure 5.10**, **Figure 5.11** and **Table 5.2**, which are discussed below:

- The total number processing stage obtained is 17. This is the maximum value of the number of the processing stage with the proposed technique using any frame of size (240×352), the size of Frame93 and Frame96 of the *mobile and calendar* sequence. The reason behind this is that the proposed technique uses a binary tree technique. So in the worst case there may be a block at the final stage to be coded that is obtained by splitting equally the corresponding previous stage blocks until the final stage sub-block is reduced to a single active pixel. So the total number of stages required to code this sub block is $\lceil \log_2 240 \rceil + \lceil \log_2 352 \rceil = 8 + 9 = 17$. As for comparison, the

maximum value of the number of processing stages in a quadtree is 7 (with the algorithm used in this report), because quadtree divides a block both horizontally and vertically at the same processing stage. So the maximum value of the number of the processing stage is $\lceil \log_2 64 \rceil + 1$(initially the whole image is divided into square blocks of size 64×64) = 6+1=7. This observation can be verified in **Figure 5.3**.

- With conventional block matching technique (16×16 block size), the number of transmitted bits is equal to 12 (6 bits for each component of motion vector) times the total number of blocks plus the number of bits required to transmit the block size (6 bits). For the *mobile and calendar* sequence, where each frame is of size (240×352), the number of blocks is equal to (8×9)=72. So the total number of transmitted bits is equal to 870 (bit rate = 26.1 kbit/sec). The PSNR for the predicted image with respect to the original image using motion information between Frame93 and Frame96 with block matching technique is 21.44. This PSNR can be achieved at the processing stage 9 and bit rate 13.68 kbit/sec with the proposed algorithm. This bit rate is lower than that of the block matching technique. The processing stage in the proposed scheme required to get the predicted image of the same quality as that of block matching technique deserves some attention. For a ($m_1 \times m_2$) block size, if there is a break in motion within a block then the whole block of size ($m_1 \times m_2$) will get a wrong motion vector. On the other hand, if the transmitter of the proposed technique stops transmission after a certain processing stage, $p=p_r+p_c$ then the worst possible uncoded block in the decoder will be of size $\left(2^{(mr-pr)} \times 2^{(mc-pc)}\right)$, where p_r is the number of

horizontal processing, p_c is the number of vertical processing, mr is equal to $\lceil \log_2(number\ of\ rows\ of\ the\ image) \rceil$ and mc is equal to $\lceil \log_2(number\ of\ columns\ of\ the\ image) \rceil$. This block is equivalent to getting a wrong motion vector in the conventional block matching technique. The number of processing stages required in the proposed technique to get the predicted image of same quality using block matching technique with block size ($m_1 \times m_2$) can be theoretically calculated, which is approximately equal to the sum of $(mr - \log_2 m_1)$ and $(mc - \log_2 m_2)$. When m_1=16 and m_2=16, then m_r=8, m_c=9, p_r=4, p_c=5 and p=9.

- **Figure 5.11(a)** shows the convex-shaped PSNR versus Processing Stage plot. This suggests that the proposed technique codes the largest possible block of uniform motion at a certain processing stage. So the rate of improvement in PSNR decreases as the processing stage increases.

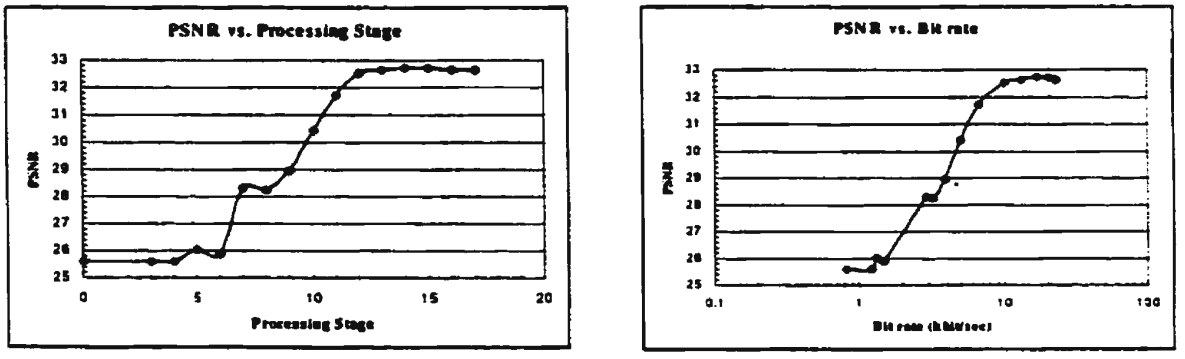
- **Figure 5.11** show that the final stage PSNR is slightly lower than that of one or two stages before the final processing stage. The reason may be that the final stage motion boundary is based on the spatial segmentation of the reference frame, which generated slight irregularity in moving object boundaries. Using the simple technique described in **Section 5.2.1** smoothes these irregularities and this is verified in **Figure 5.10**. Moreover, those steps with higher PSNR required more motion vectors to be transmitted which compensated the temporary illumination change in a certain region.

- **Table 5.2** shows that the final stage PSNR is slightly lower than that obtained with the estimated motion vector for individual pixels in the frame (**Figure 4.5**). This is because the toy in Frame93 of the *mobile and calendar* sequence is moving with the background motion vector and a thin portion of this toy is overlapping the moving ball. This results in a thick boundary plot in that region (**Figure 5.10(g)**), which is merged to the region corresponding the motion vector of the ball after using the simple merging algorithm of **Section 5.2.1** (**Figure 5.10(h)**).

Similar analysis is shown in **Table 5.3** and **Figure 5.12** for Frame5 and Frame6 of the *table tennis* sequence.

Table 5.3: Variation of quality of the predicted image with bit rate using two frames of the *table tennis* sequence.

Stage No.	Transmitted bits for the boundary (A)	Total regions (B)	PSNR	Bit Rate (6+12×B+A) ×30
3	14	2	25.5889	804
4	21	3	25.5889	1206
5	37	3	26.0362	1302
6	65	3	25.8765	1470
7	127	6	28.309	2922
8	228	5	28.2462	3168
9	343	5	28.9683	3858
10	531	5	30.4297	4986
11	819	5	31.7274	6714
12	1245	7	32.5341	9990
13	1770	7	32.6357	13140
14	2351	8	32.7374	16986
15	2952	8	32.7106	20592
16	3312	7	32.6394	22392
17	3393	7	32.6263	22878



(a) (b)

Figure 5.12: (a) PSNR vs. Processing Stage and (b) PSNR vs. Bit Rate of the predicted image using Frame5 and Frame6 of the *table tennis* sequence.

Some important observations with **Table 5.3** and **Figure 5.12** are given below:

- With conventional block matching technique (16×16 block size), the total number of transmitted bits is equal to 870 (bit rate = 26.1 kbit/sec). The PSNR for the predicted image with respect to the original image using motion information between Frame5and Frame6 with block matching technique is 29.1148. This PSNR can be achieved at the processing stage 10, which is very close to the theoretically calculated value 9 and bit rate 4.986 kbit/sec with the proposed algorithm. This bit rate is lower than that of the block matching technique.

- The PSNR improvement at the initial stage is negligible. It is because the coded uniform motion region corresponds to the static background, which merely contributes to the improvement of PSNR.

- **Figure 5.12** shows a better rate of PSNR improvement with the increment of processing step at the middle stage than that of **Figure 5.11**. This is because frames of the table tennis sequence used in the analysis have dominant translational moving objects where the proposed algorithm fits better than for the used frames of the *mobile and the calendar* sequence.

5.3 Efficiency Analysis of the Proposed Coding Technique in Terms of Entropy

This section analyzes the efficiency of the proposed coding scheme. The analysis is performed on two boundary images and two thicker textual images. **Table 5.4** shows a sample analysis of the proposed coding scheme for the motion boundary image using Frame5 and Frame6 of the *table tennis* sequence. The efficiency is compared against the entropy of a symbol, which is calculated by $H = -p \log_2 p$, where p is the probability of occurrence of the symbol. The entropy of a symbol represents its information content. It also provides the information on the minimum number of required bits to code a symbol. If the entropy is less than the number of bits required transmitting a symbol, the symbol is considered to be inefficiently coded. For efficient coding the entropy should be close to the average number of transmitted bits for a symbol.

Table 5.4: Sample calculation of efficiency analysis of the proposed coding technique in terms of entropy.

Case 1: For (m×n) block at Primary Stage					
Pseudo Symbol (Code)	**A (0)**	**B (10)**	**C (11)**		**Total**
Length of Code (L)	1	2	2		
No. of Occurrence (N)	369	207	218		794
Probability (P)	0.4648	0.26071	0.2746		1.0
Entropy (H)	0.5138	0.5056	0.5120		1.5314
Number of bits with the proposed tech. (M=L×N)	369	414	436		**1219**
Average No. of bits (Total M/Total N)					1.5353
Number of bits with Entropy (Total H×Total N)					**1216**
Case 2: For (m×n) block at Advanced Stage					
Pseudo Symbol (Code)	**D (0)**	**E and F (1)**			**Total**
Length of Code (L)	1	1			
No. of Occurrence (N)	130	19			149
Probability (P)	0.8725	0.1275			1.0
Entropy (H)	0.1717	0.3789			0.5506
Number of bits with the proposed tech. (M=L×N)	130	19			**149**
Average No. of bits (Total M/Total N)					1.0
Number of bits with Entropy (Total H×Total N)					**83**
Case3: For (m×1) or (1×m) block at Primary Stage					
Pseudo Symbol (Code)	**G (00)**	**H (01)**	**I (10)**	**J(11)**	**Total**
Length of Code (L)	2	2	2	2	
No. of Occurrence (N)	94	78	120	78	370
Probability (P)	0.2541	0.2108	0.3243	0.2108	1.0
Entropy (H)	0.5022	0.4735	0.5269	0.4735	1.9760
Number of bits with the proposed tech. (M=L×N)	188	156	240	156	**740**
Average No. of bits (Total M/Total N)					2
Number of bits with Entropy (Total H×Total N)					**732**

Case4: For (m×1) or (1×m) block at Advanced Stage					
Pseudo Symbol (Code)	**K (0)**	**L and M (10)**	**N (11)**		**Total**
Length of Code (L)	1	2	2		
No. of Occurrence (N)	13	1	9		23
Probability (P)	0.5652	0.0435	0.3913		1.0
Entropy (H)	0.4652	0.1967	0.5297		1.1916
Number of bits with the proposed tech. (M=L×N)	13	2	18		**33**
Average No. of bits (Total M/Total N)					1.4348
Number of bits with Entropy (Total H×Total N)					**28**
Case 5: For (2×1) or (1×2) block at Primary Stage					
Pseudo Symbol (Code)	**O (0)**	**P (10)**	**Q (11)**		**Total**
Length of Code (L)	1	2	2		
No. of Occurrence (N)	125	90	59		274
Probability (P)	0.4562	0.3285	0.2153		1.0
Entropy (H)	0.5165	0.5276	0.4771		1.5211
Number of bits with the proposed tech. (M=L×N)	125	180	118		**423**
Average No. of bits (Total M/Total N)					1.5438
Number of bits with Entropy (Total H×Total N)					**417**

Case 6: For (2×1) or (1×2) block at Advanced Stage					
Pseudo Symbol (Code)	**R (0)**	**T (1)**			**Total**
Length of Code (L)	1	1			
No. of Occurrence (N)	60	10			70
Probability (P)	0.8571	0.1429			
Entropy (H)	0.1906	0.4011			0.5917
Number of bits with the proposed tech. (M=L×N)	60	10			**70**
Average No. of bits (Total M/Total N)					1
Number of bits with Entropy (Total H×Total N)					**42**

Total number of '1', continuation marks: **431**

Total number of '0', terminating marks: **327**

Total bits with the proposed technique: 431+327+1219+149+740+33+423+70 = **3392**

Total entropy coded bits: 431+327+1216+83+732+28+417+42 = **3276**

Similar analysis is performed for the boundary images used in experiment 2, 3 (**Figure 5.7**) and experiment 1 (**Figure 5.9**). A summary of this analysis is shown in **Table 5.5.**

Table 5.5: A summary of efficiency analysis of the proposed coding technique in terms of entropy.

Experiment 1 (Figure 5.9)						
	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case5**	**Case6**
Average no. of bits with the proposed technique	1.4836	1	2	1.6482	1.5443	1
Total Entropy	1.4776	0.937	1.987	1.58	1.5131	0.9299
Experiment 2 (Figure 5.7)						
	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case5**	**Case6**
Average no. of bits with the proposed technique	1.4804	1	2	1.7667	1.5115	1
Total Entropy	1.4789	0.9951	1.8679	1.4984	1.4952	0.9627

Experiment 3 (Figure 5.7)						
	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case5**	**Case6**
Average no. of bits with the proposed technique	1.4123	1	2	1.5	1.5271	1
Total Entropy	1.3897	0.6415	1.9711	1.4549	1.4317	0.5505

From **Table 5.4** and **5.5**, it can be observed that the codewords selected in the *primary stages* of the proposed technique are very close to the entropy for all four experiments. But there is a scope to improve the codeword selection in the *advanced stages*. But this will not affect the total number of transmitted bits too much as the number of blocks in *advanced stages* is small compared to that of the number of blocks in the *primary stages*.

Table 5.6 shows the statistics of the continuation portion in the proposed coding scheme for the motion boundary image obtained from Frame5 and Frame6 of the *table tennis* sequence.

Table 5.6: Sample calculation of efficiency analysis of the run length coding part of the proposed coding technique in terms of entropy.

Case 1: For (m×n) block at Primary Stage (Pseudocode: **B**)						
Run length	**0**	**1**	**2**	**3**	**4**	**5**
Number of occurrence (without terminating '0')	67	19	14	6		1
Number of occurrence (with terminating '0')	53	38	4	2	3	
Case 2: For (m×n) block at Primary Stage (Pseudocode: **C**)						
Run length	**0**	**1**	**2**	**3**	**4**	**5**
Number of occurrence (without terminating '0')	42	25	18	6	2	
Number of occurrence (with terminating '0')	87	26	8	3	1	
Case 3: For (m×n) block at Advanced Stage (Pseudocode: **E** and **F**)						
Run length	**0**	**1**	**2**	**3**	**4**	**5**
Number of occurrence (without terminating '0')	8	2				
Number of occurrence (with terminating '0')	4					

Case 4: For (m×1) and (1×m) block at Primary Stage (Pseudocode: **H**)						
Run length	**0**	**1**	**2**	**3**	**4**	**5**
Number of occurrence (without terminating '0')	18	15	11	1		
Number of occurrence (with terminating '0')	22	10	1			
Case 5: For (m×1) and (1×m) block at Primary Stage (Pseudocode: **I**)						
Run length	**0**	**1**	**2**	**3**	**4**	**5**
Number of occurrence (without terminating '0')		41	13	5	1	
Number of occurrence (with terminating '0')	47	12	1			
Case 6: For (m×1) and (1×m) block at Advanced Stage (Pseudocode: **L** and **M**)						
Run length	**0**	**1**	**2**	**3**	**4**	**5**
Number of occurrence (without terminating '0')		1				
Number of occurrence (with terminating '0')						

The summary of the entropy analysis for the same four experiments used in the previous entropy analysis is shown in **Table 5.7.**

Table 5.7: A summary of efficiency analysis of the run length coding part of the proposed coding technique with two-level images.

Experiment 5 (Figure 5.9)						
	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case5**	**Case6**
Average no. of bits with the proposed technique	1.13043	1.22477	0.57895	1.08974	1.33333	1
Total Entropy	2.5065	2.50835	1.37796	2.40024	2.04472	0

Experiment 1 (Figure 5.9)						
	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case5**	**Case6**
Average no. of bits with the proposed technique	0.96818	1.121	0.84615	0.92727	1.23129	0.89474
Total Entropy	2.26072	2.4219	2.02233	2.14159	1.76489	1.08699

Experiment 2 (Figure 5.7)						
	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case5**	**Case6**
Average no. of bits with the proposed technique	1.02711	1.18362	1.00943	0.85	1.15663	0.625
Total Entropy	2.35119	2.4783	2.20256	2.19538	1.62406	1.40564

Experiment 3 (Figure 5.7)						
	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case5**	**Case6**
Average no. of bits with the proposed technique	0.92401	1.2932	0.69231	0.52507	1.11623	0.77778
Total Entropy	2.11504	2.33056	1.74957	1.61733	1.53071	1.54071

Table 5.7 shows that the continuation-coding portion of the proposed technique requires fewer bits on average than the entropy of runlength coding for all four experiments and in all cases. So, the entropy coding will not be efficient on this portion of the coding technique. For experiment 5 (case 6), the entropy is **0** which is an exceptional case. This means that any block with codeword **L** or **M** must have a run length **1**. The extra bit for run length does not carry any extra information.

5.4 Summary

This chapter shows the effectiveness of the new coding scheme in terms of transmitted bits compared to the quadtree, binary tree and READ coding techniques (**Figure 5.7** and **5.8**). As the coding scheme is optimized for thin boundary images, the scheme works better on the contour images than the thicker textual images.

The important aspect of the proposed coding technique is that it codes the largest possible block of uniform region at a certain processing stage. So the rate of improvement of

PSNR of the recovered image compared to the original image is high at the initial stages and decreases as the processing stage increases (**Figure 5.10, 5.11,** and **5.12**).

Table 5.4, 5.5, 5.6, and **5.7** show the effectiveness of the codeword selected with the proposed coding scheme. In most of the cases, the entropy is very close the average number of transmitted bits.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

6.1.1 Motion Estimation and Segmentation

The proposed motion estimation and segmentation algorithm shows better quality of the predicted image and motion boundary extraction than the block matching technique in case of frames with translational moving objects. This algorithm considers region based matching instead of block based matching, where the ill-defined pixels at one processing stage constitute the region to be matched in the next stage. This solves the problem of block size selection as in the case of traditional block matching technique. In addition, the proposed techniques needs to transmit fewer motion vectors. This makes the transmission of motion information efficient.

6.1.2 Motion Boundary Coding

The proposed coding algorithm is found to be very efficient in transmitting motion boundaries, in terms of the number of transmitted bits, as compared to the conventional quadtree, binary tree and READ coding scheme. The advantage of this scheme is that the position of the cut in an uncoded block is put at a location nearest to the boundary. So the algorithm tries to code the maximum possible uniform regions at its initial stages, which means larger regions of uniform motion are predicted at the earlier stages. This will help in the rough construction of objects with very low bit rate.

6.2 Recommendations for Further Studies

There are several avenues of the work that are recommended for further studies.

6.2.1 Motion Estimation and Segmentation

- The proposed motion estimation algorithm estimates only translational motion with sufficient degree of accuracy. So object warping, scale and rotation can be incorporated with the scheme to make a more robust system. The possible solution to this problem will be to consider eight-parameter global motion estimation at the initial stage.

- As the number of pixels to represent an object decreases with the increment of the processing steps, the reliability of the gradient descent search algorithm decreases. This requires a full search algorithm, which needs a search range to be defined and makes the motion vector calculation more time consuming than the gradient descent algorithm. In order to avoid the use of the full search algorithm, thresholding of the difference image after motion vector compensation can be made ineffective to some isolated regions where the number of pixels affected is very small. This may avoid progressive thinning of unwanted objects as the motion estimation proceeds.

- In the proposed algorithm several post-processing strategies are used to avoid temporal illumination changes, which causes the formation of small regions with

incorrect motion vector. A suitable criterion can be suggested to deal with the problem of temporal illumination changes in an efficient way.

- The motion vector is calculated to the full pixel accuracy. Sub-pixel accuracy can be studied. This may increase the accuracy of motion vector calculation and the quality of the predicted image.

- Progressive motion estimation between frames in a long video sequence can be studied to deal with uncovered background and to refine motion boundaries as the sequence proceeds. This may also help in the application of thresholding to the difference image on a per-segment basis.

6.2.2 Motion Boundary Coding

- The proposed coding algorithm is designed for on pixel thick contour images. The scheme can be studied to make it applicable to more general two level images, such as thicker textual images where the algorithm performs worse than the READ coding scheme. The possible solution may be to merge some special blocks at each processing stage to form a more advanced stage block, which has no possibility of having a white sub-block and should be prevented from subdividing. Then these more advanced blocks should be investigated to choose one codeword out of two, whether all pixels in the block have activity or not. This may reduce the number of costly bits required to transmit the information at the finest level.

REFERENCES

[1] **Sikora, T.**, "The MPEG-4 Video Standard Verification Model", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, No. 1, pp 19-31, February 1997.

[2] **Koenen, R.**, "MPEG-4 Overview - (V.14 - Geneva Version)", *ISO/IEC JTC1/SC29/WG11 N3444*, Geneva, May/June 2000.

[3] **Ngo, C.W.**, **Pong, T. C.**, and **Chin, R. T.**, "A Survey of Video Parsing and Image Indexing Techniques in Compressed Domain", *Symposium on Image, Speech, Signal Processing, and Robotics*, Vol. 1, pp. 231-236, Hong Kong, 1998.

[4] **Vass, J.**, "MPEG-7 Multimedia Content Description Interface", *Technical Report, Multimedia Communications and Visualization Laboratory*, University of Missouri-Columbia, February 1998.

[5] **Murray, D. W.**, and **Buxton, B. F.**, "Experiments in the Machine Interpretation of Visual Motion", *The MIT Press*, 1990.

[6] **Limb, J. O.**, and **Murphy, J. A.**, "Estimating the Velocity of Moving Images in Television Signals", *Computer Graphics and Image Processing*, Vol. 4, pp 311-327, 1975.

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

[8] **Marr, D.**, and **Ullman, S.**, "Directional Selectivity and Its Use in Early Visual Processing", *Proceedings of the Royal Society of London, Section B*, pp. 269-294, 1981.

[9] **Marr, D.**, and **Hildreth, E. C.**, "Theory of Edge Detection", *Proceedings of the Royal Society of London, Section B*, pp. 187-217, 1980.

[10] **Buxton, B. F.**, and **Buxton, H.**, "Computation of Optic Flow from the Motion of Edge Features in the Image", *Image and Vision Computing*, Vol. 2, pp. 59-75, 1984.

[11] **Duncan, J. H.**, and **Chou, T. C.**, "Temporal Edges the Detection of Motion and the Computation of Optical Flow", *Proceedings of the 2nd International Conference on Computer Vision*, Tampa, FL, pp. 374-382, 1988.

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

[13] **Koga, J., Iiunuma, K., Hirani, A., Iijima, Y.**, and **Ishiguro, T.**, "Motion Compensated Interframe Coding for Video Conferencing", *Proceedings of the National Telecommunications Conference*, pp. G5.3.1-5.3.5, 1981.

[14] **Liu, L. K.**, and **Feig, E.**, "A Block-Based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 4, pp. 419-422, August 1996.

[15] **Bierling, M.**, "Displacement estimation by hierarchical block matching", *Proceedings of 3rd SPIE Symposium on Visual Communications and Image Processing*, Cambridge, USA, pp. 942-951, November 1988.

[16] **Ghanbari, M.**, "The Cross-Search Algorithm for Motion Estimation", *IEEE Transactions on Communications*, Vol. 38, No. 7, pp. 950-953, July 1990.

[17] **Chan, M., Yu, Y.**, and **Constantinides, A.**, "Variable size block matching motion compensation with applications to Video Coding", *Proceedings of Institutes for Electrical Engineers*, Vol. 137, No. 4, pp. 205-212, August 1990.

[18] **Rhee, I., Martin, G. R., Muthukrishnan, S.**, and **Packwood, R. A.**, "Quadtree-Structured Variable-Size Block-Matching Motion Estimation with Minimal Error", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 1, February 2000.

[19] **Haskell, B. G.**, "Frame-to-frame Coding of Television Pictures using Two-dimensional Fourier Transforms", *IEEE Transactions on Information Theory*, Vol. 20, pp. 119-120, 1974.

[20] **Magarey, J. F. A.**, and **Kingsbury, N. G.**, "Motion Estimation Using Complex Wavelets", *Technical Report, Department of Engineering, University of Cambridge*, U. K., 1995.

[21] **Cicconi, P.**, and **Nicolas, H.**, "Efficient Region-based Motion Estimation and Symmetry Oriented Segmentation for Image Sequence Coding", *IEEE Transactions on Circuits Systems for Video Technology*, Vol. 4, No. 3, pp. 357-364, 1994.

[22] **Wang, D., Labit, C.**, and **Ronsin, J.**, "Region-based Motion Compensated Video Coding using Morphological Filters", *IEEE Transactions on Circuits Systems for Video Technology*, Vol. 7, No. 3, pp. 549-555, 1997.

[23] **Salembier, P., Torres, L., Meyer, F.,** and **Gu, C.,** "Region-based Video Coding using Mathematical Morphology", *Proceedings of IEEE*, Vol. 83, No. 6, pp. 843-857, 1995.

[24] **Yokoyama, Y., Miyamoto, Y.,** and **Ohta, M.,** "Very Low Bit Rate Video Coding using Arbitrary Shaped Region-based Motion Compensation", *IEEE Transactions on Circuits Systems for Video Technology*, Vol. 5, No. 6, pp. 500-507, 1995.

[25] **Kunt, M., Benard, M.,** and **Leonardi, R.,** "Recent Results in High-Compression Image Coding", *IEEE Transactions on Circuits and Systems*, Vol. CAS-34, No. 11, November 1987.

[26] **Musmann, H. G., Hötter, M.,** and **Ostermann, J.,** "Object-Oriented Analysis-Synthesis Coding of Moving Images", *Signal Processing: Image Communications*, Vol. 1, No. 2, pp.117-138, October 1989.

[27] **Shi, J.,** and **Malik, J.,** "Motion Segmentation and Tracking Using Normalized Cuts", *International Conference on Computer Vision*, January 1998.

[28] **Nguyen, H. T., Worring, M.,** and **Dev, A.,** "Robust Motion-based Segmentation in Video Sequences", *Technical Report*, Department of Computer Science, University of Amsterdam, The Netherlands, 1999.

[29] **Wang, D.,** "Improvement of Region-based Motion Estimation by Considering Uncovered Regions", *Signal Processing: Image Communications*, Vol. 14, No. 10, pp. 841-849, 1999.

[30] **Robinson, J. A.**, "Low-data-rate Visual Communication using Cartoons: a Comparison of Data Compression Techniques", *IEE Proceedings*, Vol. 133, No. 3, pp. 236-256, June 1986.

[31] **Freeman, H.**, "On the Encoding of Arbitrary Geometric Configurations", *IRE Transactions*, EC-10, pp. 260-268, 1961.

[32] **Koplowitz, J.**, and **Deleone, J.**, "Hierarchical Representation of Chain-Encoded Binary Image Contours", *Computer Vision and Image Understanding*, Vol. 63, No. 2, pp. 344-352, March 1996.

[33] **Knowlton, K.**, "Progressive Transmission of Grey Scale and B/W Pictures by Simple Efficient and Lossless Encoding Schemes", *Proceedings of IEEE*, Vol. 68, pp. 885-896, 1980.

[34] **Cohen, Y.**, **Landy, M. S.**, and **Pavel, M.**, "Hierarchical Coding of Binary Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 3, pp 284-298, May 1985.

[35] **Gonzalez, R. C.**, and **Woods, R. E.**, "Digital Image Processing", *Addison-Wesley Publishing Company*, 1992.

APPENDIX A

TEST TEXTUAL IMAGES

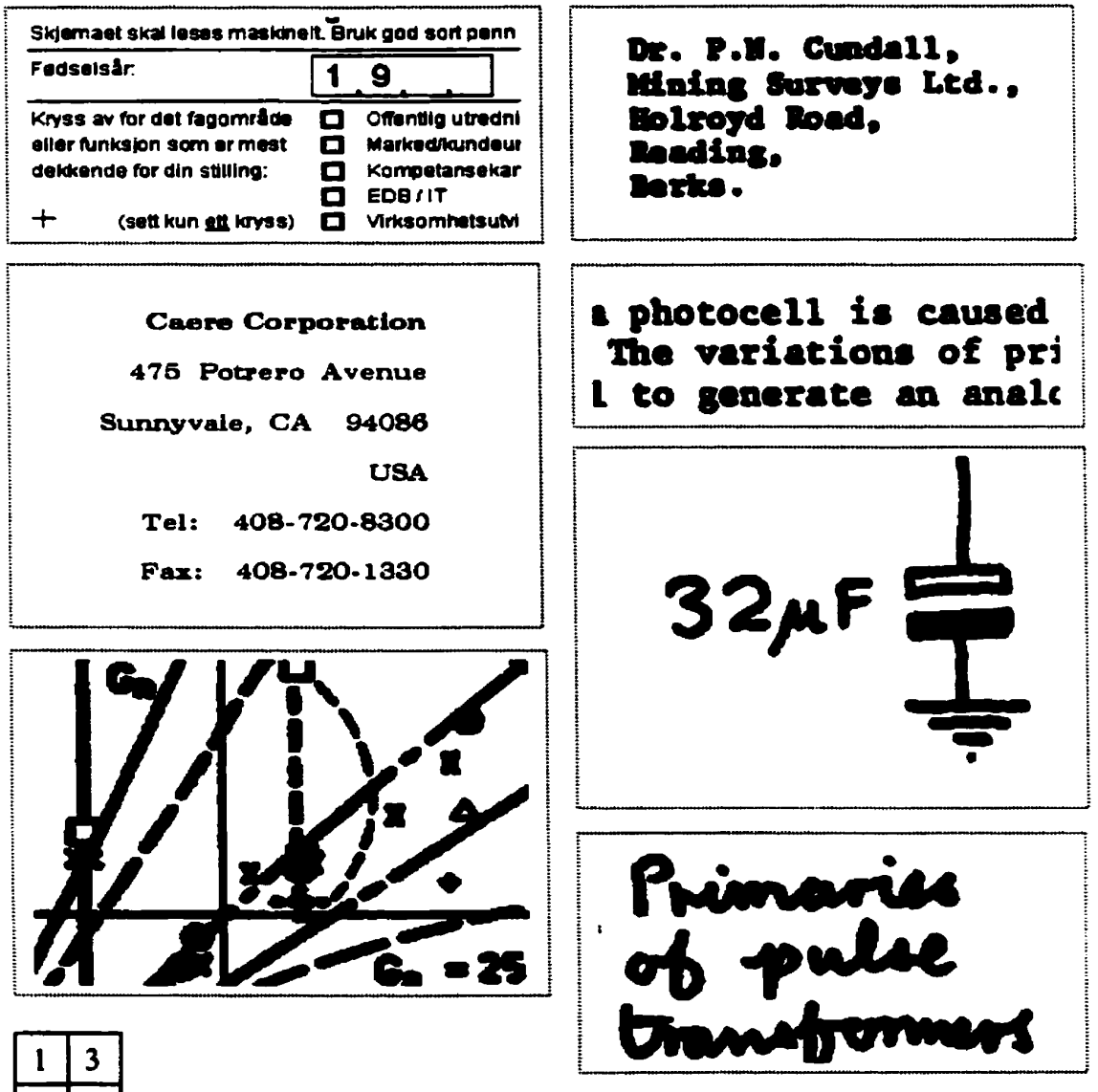


Figure A.1: Textual images used in the performance comparison of different coding schemes.