UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

CONCEPTION DES MAILLES BASÉE SUR UNE SEGMENTATION D'IMAGE POUR L'ESTIMATION DES MOUVEMENTS

SEGMENTATION-BASED MESH DESIGN FOR MOTION ESTIMATION

Mémoire de maîtrise es sciences appliquées
Spécialité : Génie électrique et génie informatique

Lijun CHEN

Sherbrooke (Québec), Canada

Décembre 1998

0-612-67699-4

Résumé

Dans la plupart des codec vidéo standard, l'estimation des mouvements entre deux images se fait généralement par l'algorithme de concordance des blocs ou encore BMA pour « Block Matching Algorithm ». BMA permet de représenter l'évolution du contenu des images en décomposant normalement une image par blocs 2D en mouvement translationnel. Cette technique de prédiction conduit habituellement à de sévères distorsions de l'artefact de bloc lorsque le mouvement est important. De plus, la décomposition systématique en blocs réguliers ne tient pas compte nullement du contenu de l'image. Certains paramètres associés aux blocs, mais inutiles, doivent être transmis; ce qui résulte d'une augmentation de débit de transmission.

Pour paillier à ces défauts de BMA, on considère les deux objectifs importants dans le codage vidéo, qui sont de recevoir une bonne qualité d'une part et de réduire la transmission à très bas débit d'autre part. Dans le but de combiner les deux exigences quasi contradictoires, il est nécessaire d'utiliser une technique de compensation de mouvement qui donne, comme transformation, de bonnes caractéristiques subjectives et requiert uniquement, pour la transmission, l'information de mouvement. Ce mémoire propose une technique de compensation de mouvement en concevant des mailles 2D triangulaires à partir d'une segmentation de l'image. La décomposition des mailles est construite à partir des nœuds répartis irrégulièrement le long des contours dans l'image. La décomposition résultant est ainsi basée sur le contenu de l'image. De plus, étant donné la même méthode de sélection des nœuds appliquée à l'encodage et au décodage, la seule information requise est leurs vecteurs de mouvement et un très bas débit de transmission peut ainsi être réalisé. Notre approche, comparée avec BMA, améliore à la fois la qualité subjective et objective avec beaucoup moins d'informations de mouvement.

Dans la premier chapitre, une introduction au projet sera présentée. Dans le deuxième chapitre, on analysera quelques techniques de compression dans les codec standard et, surtout, la populaire BMA et ses défauts. Dans le troisième chapitre, notre algorithme proposé et appelé la conception active des mailles à base de segmentation, sera discuté en détail. Ensuite, les estimation et compensation de mouvement seront décrites dans le chapitre 4. Finalement, au chapitre 5, les résultats de simulation et la conclusion seront présentés.

Abstract

In most video compression standards today, the generally accepted method for temporal prediction is motion compensation using block matching algorithm (BMA). BMA represents the scene content evolution with 2-D rigid translational moving blocks. This kind of predictive scheme usually leads to distortions such as block artefacts especially when the motion is important. In addition, because the block decomposition is not content adapted, some useless block parameters may be transmitted and consequently result in increasing bit-rate cost.

The two most important aims in video coding are to receive a good quality on one hand and a low bit-rate on the other. To combine these almost contradictory requirements, it is necessary to use a motion compensation technique which has good subjective transformation characteristics and does not need the transmission of much more than motion information. This thesis proposes a motion compensation scheme using segmentation-based 2-D triangular mesh design method. The mesh is constructed by irregularly spread nodal points selected along image contour. Based on this, the generated mesh is, to a great extent, image content based. Moreover, the nodes are selected with the same method on the encoder and decoder sides, so that the only information that has to be transmitted are their motion vectors, and thus very low bit-rate can be achieved. Compared with BMA, our approach could improve subjective and objective quality with much less motion information.

In the first chapter, an introduction of the project is given. In the second chapter, we analyse the techniques of some existing video coding standards and the widely used motion estimation method --- BMA. Afterwards, our approach to overcome drawbacks of BMA for video coding is proposed. In chapter 3, our algorithm, called segmentation-based active mesh design, is discussed in detail. Then, motion estimation and compensation are described in chapter 4. Finally, in chapter 5, the simulation results and conclusion are presented.

Acknowledgements

I am indebted to Professor Chon Tam Ledinh, my research supervisor at the Université de Sherbrooke, for his constant advice and encouragement. His generous help always made me confident in overcoming the difficulties in both research and personal life.

I wish to express my gratitude to Dr. Demin Wang at CRC (Communications Research Centre) for his help and valuable suggestions.

I am grateful to Professor Sarto Morrissette in the Department of Electrical and Computer Engineering, for his kind reviewing text and many helpful discussions.

I am thankful to all the faculty members in the Department of Electrical and Computer Engineering, Université de Sherbrooke, for their kindness and helpful encouragement.

Finally, many thanks to my parents and sisters for their support and patience.

Contents

Résumé I

Abstract II

Acknowledgements III

Contents IV

List of Figures and Tables VI

Chapter 1. Introduction 1

Chapter 2. Analysis and Proposal of Video Compression Techniques 4

2.1 Video Compression Basics 4

2.1.1 Image Sequence Model 5

2.1.2 Spatial Redundancy Reduction Scheme 5

2.1.3 Spatial -Temporal Redundancy Reduction Scheme 7

2.2 Video Compression Standards 9

2.2.1 JPEG Image Compression and Motion JPEG 9

2.2.2 The P $\times$ 64 Compression Standard 11

2.2.3 MPEG Video Compression Standard 13

2.3 Block Matching Algorithm for Motion Estimation 15

2.3.1 Principles 15

2.3.2 The Matching Criterion 17

2.3.3 Motion Vector Search Algorithm 18

2.4 Segmentation-based Mesh Design Scheme for Video Compression 19

Chapter 3. Active Mesh Model and Generation 24

3.1. Existing Transformation Models 24

3.2. Affine Transformation Mesh Model 27

3.3 Content-based Mesh Generation 29

3.3.1 Image Segmentation 29

Basic Morphological operations 30

The Multiscale Gradient Algorithm 31

Elimination of Small Local Minima........33
3.3.2 Image Simplification........34
Principle........34
3.3.3. Selection of Control nodes........37
3.3.4. Triangular Mesh Generation with Triangle Reduction........44
3.4. Spatial-Temporal Gradients Method of Mesh Design........50
Chapter 4. Motion Estimation........52
4.1. Initialization by Block Matching........53
4.2. Avoiding Triangle Inconsistency........54
4.3. Refinement by Deformation of the Triangulation........56
4.3.1. Principle........56
4.3.2. Support Region and Search Region for a Node........58
4.3.3. Refining........60
4.3.4. Application........62
4.5. Image Reconstruction........67
Chapter 5. Simulation Results and Conclusion........68
5.1. Simulation Results........68
5.2. Conclusion........72
References........74

List of Figures and Tables

Figures

Figure 2.1 Temporal correlation in an image sequence. 5
Figure 2.2 Generic DCT-based coding scheme. 5
Figure 2.3 Two-stage video coding process. 7
Figure 2.4 Block diagram of JPEG encoder/decoder 10
Figure 2.5 Diagram of the *P*×64 encoder/decoder. 12
Figure 2.6 Types of frames in the MPEG standard. 14
Figure 2.7 Motion estimation process. 15
Figure 2.8 Motion block artefacts. 19
Figure 2.9 Problems of motion compensation based on region. 20
Figure 2.10 Segmentation-based triangular mesh design scheme. 21
Figure 2.11 Compensation of motion and deformation based on contour control nodes. 22
Figure 3.1 Output of conventional gradient operators. 31
Figure 3.2 Result of the multiscale gradient algorithm for ramp edges 33
Figure 3.3 Elimination of small local minima. 34
Figure 3.4 Image connectivity, (a) 4-connected (b) 8-connected 36
Figure 3.5 Eight neighbour points of a given point p. 36
Figure 3.6 The first class of Control nodes. 38
Figure 3.7 Flow chat of segments tracing and eliminating. 40
Figure 3.8 Insert high curvature nodes. 41
Figure 3.9 Flow chat of inserting high curvature control points 42
Figure 3.10 Insert nodes to long segments 43
Figure 3.11 Triangulation constraint: no contour cross. (a) triangles across over a contour; (b) not across over the contour. 45
Figure 3.12 Triangle reduction. (a) before fusion (b) after fusion. 46

Figure 3.13 Flow chat of triangle fusion and node reduction. 47
Figure 3.14 Results of triangulation for sequence Miss America and Foreman..................49
Figure 4.1 Image Enlargement...53
Figure 4.2 Illustration of inconsistent motion vectors. 54
Figure 4.3 Refinement process: N_0 is the considered node, N_0' is its new location. 57
Figure 4.4 Support region and search space, (a) inner node; (b) corner and boundary nodes. 58
Figure 4.5 Search range of a considered node. 59
Figure 4.6 (a) Problem of ROI, (b) Problem of discovered region......................................61
Figure 4.7 Flow chat of motion vector refinement. 63
Figure 4.8 Backward Affine transformation...64
Figure 4.9 Detection of treated triangle. 65
Figure 4.10 Flow chat of image reconstruction. 67
Figure 5.1 PSNR curves obtained by proposed method and BMA, (a) Miss America; (b) Forman. 69
Figure 5.2 Refinement algorithm convergence..70
Figure 5.3 Original and reconstructed images, left: Miss America; right: Foreman. 72

Tables

Table 3.1 Node number of Miss America and Foreman...48
Table 5.1 PSNR of BMA and proposed method...70

Chapter 1. Introduction

With the increasing importance of digital television, teleconferencing, and multimedia applications, video coding has become a highly active research area. In most video coding standards, motion compensated prediction and transform coding are used to reduce temporal and spatial redundancy in the video signal. The success of the temporal prediction method is essential for the overall coding result. The generally accepted algorithm of motion compensation for temporal prediction is the block matching algorithm (BMA). It divides a given frame into small non-overlapping rectangular blocks and assumes that all the pixels in a block undergo the same translation. The translation or displacement vector for each block is conventionally determined by an exhaustive search. The strength of the BMA is that the underlying motion model is very simple and can be efficiently determined and specified. When the block size is sufficiently small, this model is surprisingly powerful and works well for most blocks.

However, if a block contains more than one object moving in different directions, one motion vector will result in inaccurate motion estimation of moving objects, and the prediction error will be large in that block. Consequently, more bits are needed to code the prediction error, and block artefacts will become visible if the algorithm is designed for a low-bit-rate application.

To avoid the block effect, some new segmentation-based coding techniques [1-5] have been proposed. In these methods either the still image or the difference of two successive frames in a video sequence is first segmented into a number of regions. The region texture and boundary information is then coded for transmission. The segmentation-based techniques are considered very promising. Compared with block-based motion compensation, segmentation-based motion compensation has two advantages: the first is that it produces fewer motion parameters and the second is that no block artefacts are created in reconstructed images. However, the performance of this approach is limited by two facts. Firstly, contours of segmentation maps typically require more bits to be encoded. Secondly, commonly used motion models cannot accurately describe complex motion in a region of irregular shape.

One way to avoid contour coding is to segment the reconstructed frame (image) at time *t*-1, instead of the original frame at time *t* in both encoder and decoder. There is no need to transmit the region shape information. To overcome the second drawback of segmentation-based method, we need to find a more appropriate region shape for the modelling of the motion field. One of the promising approaches is compensation based on control nodes interpolation [5-7, 11-13]. Such nodal approaches are used for global deformation estimation as well as for piecewise bilinear transformation estimation, where images are partitioned in a set of patches. Two types of meshes have been used for image partition, namely quadrangular meshes [5-7] and triangular meshes [11-13]. Recently, triangular mesh has been defined in a content-adaptive way in order to improve the motion estimation quality and decrease the motion representation cost.

In this thesis, a segmentation-based triangular mesh design for motion estimation scheme is proposed. The proposed method takes into account image contours. The image contours are obtained by using watershed-based [8] segmentation algorithm. It can guarantee that each element of the image mesh structure has an homogenous intensity distribution. To estimate motion vectors, our goal is to generate an active triangular mesh. So the control nodes are selected along with the image contour, and then triangulation is implemented based on these nodes. Because this segmentation-based mesh is content-adaptive, compared with blocks of BMA, much less control nodes need to be selected for motion estimation. To further reduce the bit rate, we propose a triangle reduction algorithm based on exploiting image statistics of each triangle. Thus, the low bit-rate objective can be achieved. Moreover, to improve the image quality of reconstructed frames, an iterative algorithm [11, 13] based on minimizing the prediction error is used to refine the motion vectors of control nodes. Such an approach could improve subjective and objective quality of reconstructed images.

The thesis is as follows. Chapter 2 gives an analysis of video compression standards and techniques, and proposes a segmentation-based mesh design scheme for motion estimation. In chapter 3, the triangular mesh model is described, and the mesh generation is performed with a triangle fusion algorithm. The motion estimation based on control nodes is given in chapter 4. Chapter 5 shows simulation results, and a conclusion is drawn.

Chapter 2. Analysis and Proposal of Video Compression Techniques

In this chapter, we present an overview of the most popular compression techniques and standards, as well as motion estimation techniques. Finally, a segmentation-based mesh design for motion estimation scheme is given.

2.1 Video Compression Basics

2.1.1 Image Sequence Model

In the still image case, we can observe that image data tend to have a high degree of spatial redundancy. Consider now the problem of capturing the movements of a 3-D object through time (Figure 2.1). In the first image, we capture a spatial projection of the object, say, in region A. Since this projection is comprised of pixels from the object, we expect correlation within the image. If the object is moving, it will yield a spatial projection in the next image as well, say in region X. Thus, we would expect a high degree of temporal redundancy between neighbour images as well; that is, there is strong correlation between pixels in region A in one image and in region X in the next image. The goal of video compression algorithm is to exploit both the spatial and the temporal redundancy within an image sequence for optimum compression.

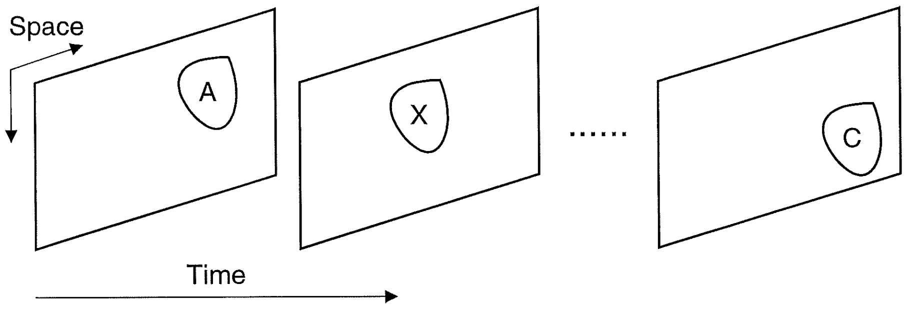


Figure 2.1 Temporal correlation in an image sequence.

2.1.2 Spatial Redundancy Reduction Scheme

The generic form of the spatial redundancy reduction scheme is shown in Figure 2.2. This diagram represents the core computation pipeline employed in all the lossy images and video compression standards discussed afterward.

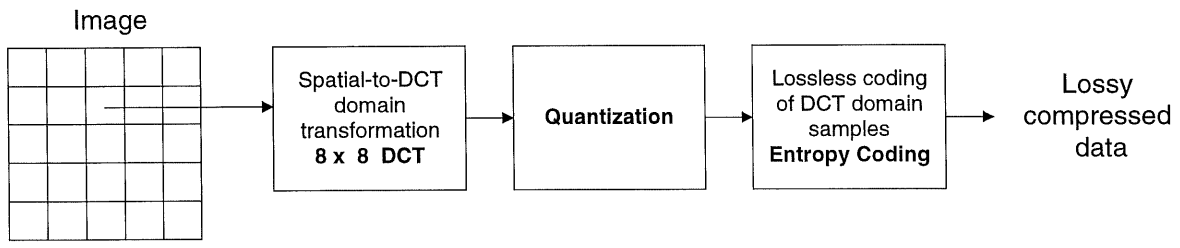


Image
Spatial-to-DCT domain transformation
8 x 8 DCT
Quantization
Lossless coding of DCT domain samples
Entropy Coding
Lossy compressed data

Figure 2.2 Generic DCT-based coding scheme.

Denote the spatial-domain samples in the 8 × 8 block shown in Figure 2.2 as (x_{ij}). The spatial-to-DCT (discrete cosine transformation) domain transformation expressed in Equation (2.1) yields an 8 × 8 block (y_{kl}) in the DCT domain.

$$y_{kl} = \frac{c(k)c(l)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7} x_{ij}\cos\left(\frac{(2i+1)k\pi}{16}\right)\cos\left(\frac{(2j+1)l\pi}{16}\right) \qquad (2.1)$$

where k, l = 0, 1, …, 7 and

$$c(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k = 0 \\ 1 & \text{otherwise} \end{cases} \qquad (2.2)$$

The choice of DCT is motivated by the many benefits it offers:

- For highly correlated image data, the compaction efficiency due to the use of a DCT is close to that obtained with the optimum transform, namely, the Karhunen-Loeve Transform (KLT).
- The DCT is a unitary orthogonal transform. Thus, if in matrix form the DCT output is $Y = T^t XT$, then the inverse transform is $X = T^t YT$.
- DCT gives real output.
- The DCT basis is image independent. This is an important issue in compression, since an image-dependent basis would imply that additional information need to be sent to the receiver for establishing the varying basis.

For DCT-based image coding system, the DCT process in the encoder transforms each 8 × 8 block X into a set of DCT coefficients Y. Each of these coefficients is the weight associated with the corresponding DCT basis waveform. In the lossy compression mode, weights will be quantized and transmitted to the receiver. The quantization process is a lossy and irreversible process. It is the main source of picture degradation in a DCT coding scheme.

After quantization, the quantized y_{kl} values are then compressed in a lossless manner using an entropy coder.

2.1.3 Spatial -Temporal Redundancy Reduction Scheme

In image sequences, there is significant spatial and temporal correlation. The DCT coding scheme of Figure 2.2 exploits the spatial redundancies within an image. In order to exploit both the spatial and temporal redundancy, one might suggest that using a 3-D DCT instead of 2-D DCT. This approach has been shown [10] to be quite effective from the compression viewpoint; however, the excessive complexity of a 3-D DCT renders this approach impractical. Instead, most video coders use a two-stage process to achieve good compression. The first stage uses a method that exploits the temporal redundancy between frames. The output of this stage is followed by a coding method (such as DCT) that exploits spatial redundancy within the frame. The basic two-stage process is illustrated in Figure 2.3.

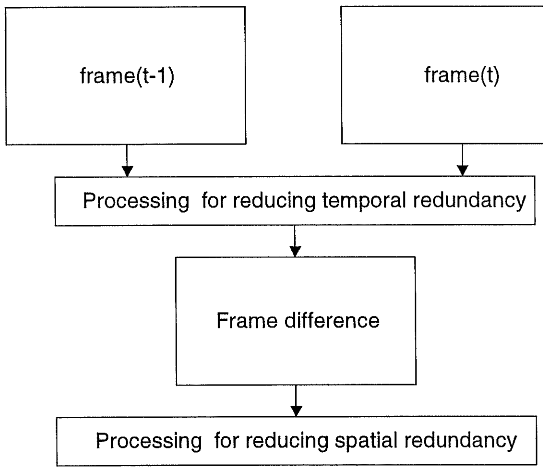

frame(t-1)
frame(t)
Processing for reducing temporal redundancy
Frame difference
Processing for reducing spatial redundancy

Figure 2.3 Two-stage video coding process.

In order to create the difference frame of Figure 2.3, the temporal redundancy processor may have to track every pixel from frame to frame. This is computationally intensive; hence, video compression standards only allow tracking information for 16×16 pixel regions, commonly referred to as macroblocks. The macroblock dimension of 16×16 is chosen because it provides a good compromise between providing efficient temporal redundancy reduction and requiring moderate computational requirement.

Let the two contiguous frames in Figure 2.3 be denoted as *frame*(t - *1*) and *frame*(t). In the first stage, we segment *frame*(t) into non-overlapping 16×16 blocks and determine a corresponding 16×16 pixel region in *frame*(t - *1*) (for the time being, we ignore how one can find such a region.)

Using the corresponding 16×16 pixel region from *frame*(t - *1*), the temporal redundancy reduction processor generates a representation for *frame*(t) that contains only the changes between the two frames. If the two frames have a high degree of temporal redundancy, then the difference frame would have a large number of pixels that have values near zero. Thus, the output image of stage 1 has lower energy than the original frame and is more amenable to compression. On the other hand, if *frame*(t) were completely different than *frame*(t - *1*), then the temporal redundancy reduction processor may fail to find corresponding regions between the two frames. In this case, one would not expect any benefit with respect to compression from using the process in stage 1.

In video compression terminology, a compression method that employs only temporal redundancy reduction is referred to as an *interframe* coder; and a compression method that employs only spatial redundancy reduction is referred to as an *intraframe* coder. The combination of interframe and intraframe coding, is referred to as a hybrid (intraframe/interframe) coding method.

The process of computing changes among frames by establishing correspondence between frames is referred to as temporal prediction with *motion compensation*. We define motion compensation as the process of compensating for the displacement of moving objects from one frame to another. In practice, motion compensation is preceded by *motion estimation*, the process of finding corresponding pixels among frames. If the temporal redundancy reduction processor employs motion compensation, then we can express its output as

$$e(x, y, t) = I(x, y, t) - I\,(x - u,\ \ y - v,\ \ t - 1) \qquad (2.3)$$

Where $I(x, y, t)$ are pixel values at spatial location (x, y) in *frame*(t) and $I(x - u, y - v, t - 1)$ are corresponding pixel values at spatial location $(x - u, y - v)$ in *frame*$(t - 1)$. The output of the motion estimator, coordinates (u, v), defines the relative motion of a block from one frame to another and is referred to as the motion vector for the block at (x, y). $I(x - u, y - v, t - 1)$ is referred to as the motion-compensated prediction of $I(x, y, t)$, and $e(x, y, t)$ is the prediction residual for $I(x, y, t)$.

Note that the notion of temporal prediction and the formation of the difference signal $e(x, y, t)$ is very similar to the differential coding scheme DPCM (differential pulse code modulation). So we call the process of Figure 2.3 as temporal DPCM scheme.

In the next section, it should be useful to consider some practical standard video codecs, namely, JPEG, P x 64, and MPEG.

2.2 Video Compression Standards

2.2.1 JPEG Image Compression and Motion JPEG

The JPEG standard specification defines a family of encoding/decoding algorithms for continuous-tone still images and a data-stream architecture for encapsulating and describing the compressed data. JPEG stands for *Joint Photographic Experts Group*, the name of the original ISO working group interested in continuous-tone image compression. JPEG was the first international compression standard for still images including both gray scale and colour images. The JPEG still picture compression standard has been extremely successful, having been implemented on virtually all platforms. This standard is fairly simple to implement, is not computationally complex, and gets 10:1 to 15:1 compression ratios without significant visual artifacts. This standard is based upon entropy encoding of quantized coefficients of the DCT of 8×8 blocks of pixel data, as discussed in section 2.1.

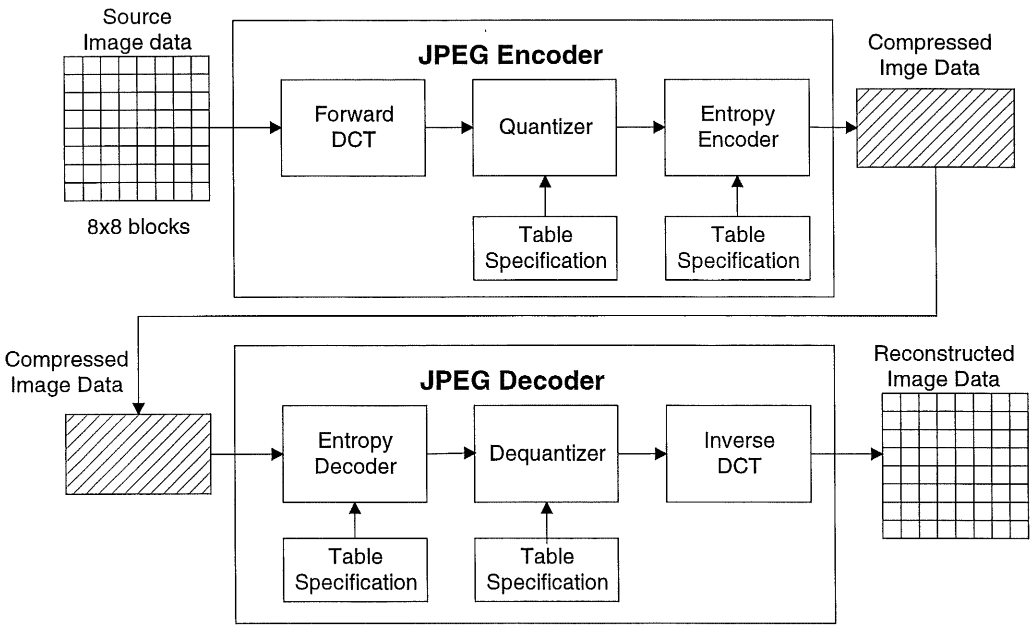


Figure 2.4 Block diagram of JPEG encoder/decoder.

Figure 2.4 shows the block diagram of both the JPEG compression and decompression algorithms. A single frame is subdivided into 8×8 blocks, each of which is independently processed. Each block is transformed into DCT space, resulting in an 8×8 block of DCT coefficients. Computation of the DCT of each block is followed by quantization of the DCT coefficients. In the quantization step each of the 64 DCT coefficients is divided by a user-selectable quantizer step-size constant and rounded to the nearest integer. The quantizing constant for each DCT coefficient is chosen to produce minimal visual arifacts, while maximally reducing the representational entropy of the coefficients. The quantized coefficients are then entropy coded into a compressed data stream. The reduced entropy of the quantized coefficients is reflected in the higher compression ratio of the data.

The motion JPEG (M-JPEG) uses the JPEG compression for each frame. It provides random access to individual frames, however the compression ratios are too low (same as in JPEG) because the technique does not take the advantage of the similarities between adjacent frames.

2.2.2 The P $\times$ 64 Compression Standard

In the late 1980s, collaboration among telecommunication operators and manufacturers of videoconferencing equipment led to the development of the H.320 video conferencing standard by the ITU (International Telecommunication Union). The video component of this standard is known as H.261, which is also known as the $P\times64$ standard because it describes video coding and decoding methods at the rates $P\times64$ kbits/s, where P is an integer from 1 to 30. For $P = 1$ to 2, due to limited available bandwidth, only a low quality video signal for picture phone and desktop face-to-face visual communications can be implemented using this compression algorithm. For $P \geq 6$, more complex pictures for video conferencing can be transmitted.

The $P\times64$ algorithm operates with two picture formats adopted by the CCITT, Common Intermediate Format (CIF), and Quarter-CIF (QCIF), as illustrated in Table 2.1.

Table 2.1 Parameters of CIF and QCIF video formats.

	CIF		QCIF	
	Lines/frame	Pixel/line	Lines/frame	Pixel/line
Luminance (Y)	288	352	144	176
Chrominance (Cb)	144	176	72	88
Chrominance (Cr)	144	176	72	88

The $P\times64$ video compression algorithm combines intraframe and interframe coding to provide fast processing for on-the-fly video. This is kind of a temporal DPCM coding scheme as described in the last section. The algorithm creates two types of frames:

(1) DCT-based intraframes, compressed using DCT, quantization, and entropy coding (similar to JPEG), and
(2) Predictive interframes, compressed using DPCM and motion estimation.

The block diagram of the video encoder is shown in Figure 2.5. The *P*×64 coding algorithm begins by coding an intraframe block and then sends it to the video multiplex coder. The same frame is then decompressed using the inverse quantizer and inverse DCT, and then stored in the frame memory for interframe coding.

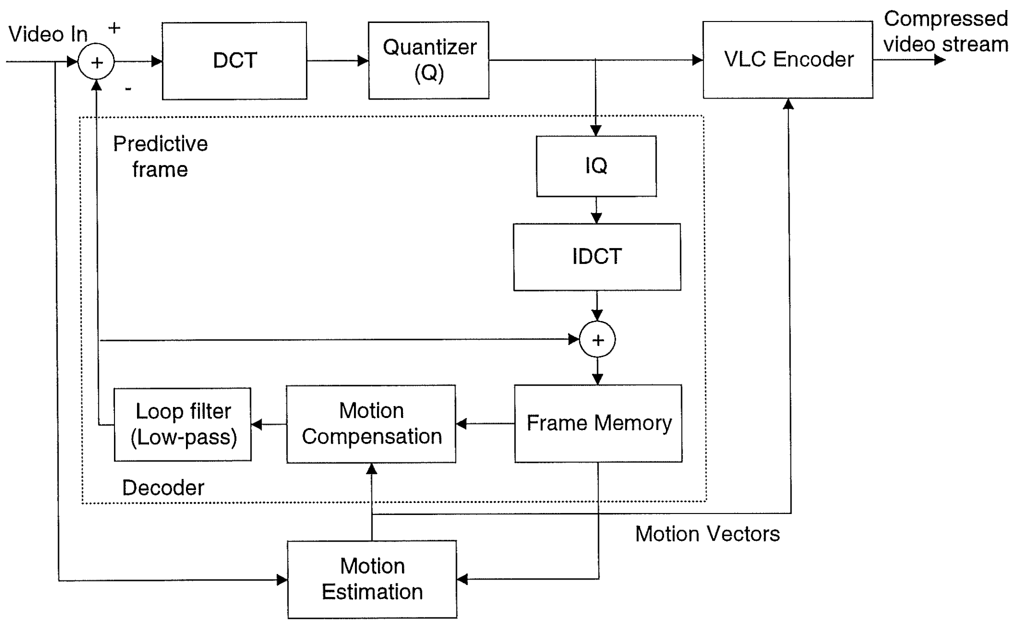


Figure 2.5 Diagram of the *P*×64 encoder/decoder.

During the interframe coding, the prediction based on the temporal DPCM algorithm is used to compare every macro block of the actual frame with the available macro blocks of the previous frame (we will describe this in details in the next section). To reduce the encoding delay, only the closest previous frame and present frame are used for prediction.

Then, the difference, denoted as error terms, is DCT-coded and quantized, and sent to the video multiplex coder together with the motion vectors. At the final step, variable-length coding (VLC), such as an Huffman encoder, is used to produce a more compact code. An optional loop (low-pass) filter can be used to minimize the prediction error by smoothing the pixels in the previous frame.

The main part of $P\times64$ video decoder is shown inside the dotted line region in Figure 2.5.

Note that the standard H.263, a more advanced version than the H.261, exists for low bit-rate video compression.

2.2.3 MPEG Video Compression Standard

The Motion Picture Experts Group, from which the MPEG standard derives its name, has defined video compression techniques that make use of the temporal redundancy inherent in sequence of full-motion video. The compression method uses interframe compression and can achieve compression ratios of 200:1 through storing only the differences between successive frames.

The MPEG first-phase standard (MPEG-1) is targeted for compression of full motion video at rates of 1 to 1.5 Mbps in applications, such as interactive multimedia video. MPEG-2 features higher data rates, with a similar architecture. It specifies compressed bit streams for high-quality digital video at the rate of 2-80 Mbps. The MPEG-2 supports interlaced video formats and a number of features for HDTV. The MPEG-4 standard is intended for compression of full-motion video consisting of small frames and requiring slow updating. The data rate required is 9-40Kbps, and the target applications include tele-surveillance, interactive multimedia and video telephony.

In the MPEG standard, the video compression scheme is also the temporal DPCM scheme, which is similar to $P\times64$. For the purpose of full motion compression, frames in a sequence are coded using three different algorithms, as illustrated in Figure 2.6.

I frames (intra frames) are self-contained and coded using a DCT-based technique similar to JPEG. I frames are used as random access points in MPEG streams, and they give the lowest compression ratios within MPEG.

P frames (predicted frames) are coded using forward predictive coding, where the actual frame is coded with reference to a previous frame (I or P). This process is similar to *P*×64 coding. Figure 2.7 illustrates the MPEG compression algorithm for predictive frames. The compression ratio of P frames is significantly higher than that of I frames.

B frames (bidirectional or interpolated frames) are coded using two reference frames, a past and a future frame (which can be I or P frames). Bidirectional, or interpolated coding provides the highest amount of compression.

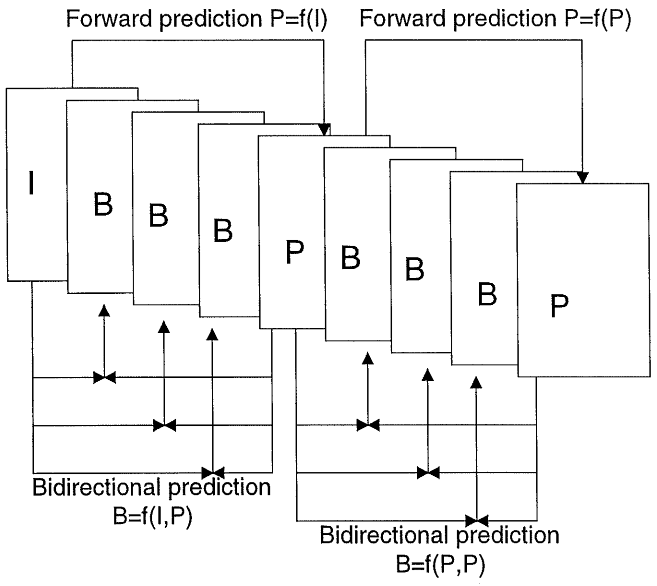

Forward prediction P=f(I)
Forward prediction P=f(P)
I
B
B
B
P
B
B
B
P
Bidirectional prediction B=f(I,P)
Bidirectional prediction B=f(P,P)

Figure 2.6 Types of frames in the MPEG standard.

The MPEG application determines a sequence of **I**, **P**, and **B** frames. If there is a need for fast random access, the best resolution would be achieved by coding the whole sequence as **I** frames (MPEG becomes identical to Motion JPEG). However, the highest compression ratio can be achieved by incorporating a large number of **B** frames. **I** frames are created similarly to JPEG encoded pictures, **P** frames are encoded based on forward prediction technique, and finally **B** frames are encoded in terms of previous and future frames. The motion vector is estimated for the **P** and **B** frames, and the difference between the predicted and actual blocks (error terms) are calculated. The error terms are then DCT encoded and the entropy encoder is used to produce the compact code.

2.3 Block Matching Algorithm for Motion Estimation

The interframe DPCM coding method is quite effective for video compression. Most video standards are based on this method. One of the most computation-intensive operations in inter-frame coding is the motion estimation process. In this section, we will describe in more details the motion estimation process and the widely used Block Matching Algorithm (BMA).

2.3.1 Principles

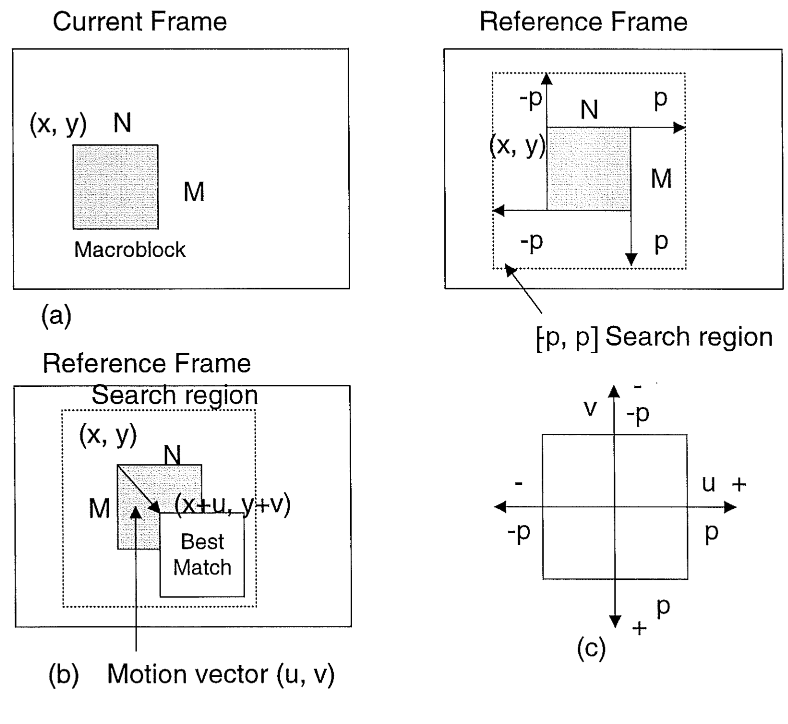


Figure 2.7 Motion estimation process.

Figure 2.7 illustrates the motion estimation problem as it is posed in the video coding standards. Given a reference picture and an N $\times$ M macroblock in a current picture, the objective of motion estimation is to determine the N × M block in the reference picture that better matches (according to a given criterion) the characteristics of the macroblock in the current picture. As current image, we define an image or frame at time *t*. As reference picture, we define an image or frame either at past time $t - n$, for forward motion estimation, or at future time $t + k$, for backward motion estimation. In the more general case of motion estimation, the geometry of the matching block at reference picture need not be the same as the geometry of the block in the current picture, since objects in the real world undergo scale changes as well as rotation and warping (as the proposed new approach in the next section). However, in the video standards, only the translational motion model is assumed for objects in the scene, and thus a rectangular geometry is adopted.

The location of the macroblock regions is given usually by the (x, y) coordinates of their left-top corner. Ideally, we would like to search the whole reference picture for the best match; however, this is impractical. Instead, we restrict the search to a [–p, p] search region around the original location of our macroblock in the current picture. (Many implementations restrict the search range to [– p, p – 1]. Both definitions are equally common.) Let (x + u, y + v) be the location of the best matching block in the reference picture (Figure 2.8 b). In motion compensation terminology, the vector from (x, y) to (x + u, y + v) is referred to as the associated motion vector in relative coordinates; that is, we assume that (x, y) is at location (0, 0), and thus the motion vector is simply expressed as (u, v).

Note that, our assumption for a common displacement (u, v) for all pixels in the macroblock implies that we are essentially imposing a local smoothness constraint on the motion vector field. The local smoothness constraint is only satisfied for small macroblock sizes. The choice of the dimensions of the macroblock is the result of tradeoffs among three conflicting requirements. Specifically,

1. Small values for N and M (from four to eight) are preferable, since the smoothness constraint would be easily met at this resolution.

2. Small values for N and M reduce the reliability of the motion vector (u, v), since few pixels participate in the matching process; and
3. Fast algorithms for finding motion vectors are more efficient for large values of N and M.

In the video coding standards, N= M = 16.

The coordinate system associated with the motion vector is shown in Figure 2.7 (b). For the search region shown in Figure 2.7 (a), $-p \le u \le p$ and $-p \le v \le p$. For broadcast TV, good performance is obtained at p = 15 for head-and-shoulder type video scenes, and at p = 63 for sporting events (high motion).

2.3.2 The Matching Criterion

The block matching motion estimation algorithms allows us to obtain the motion vector by minimizing a cost function. Various cost functions have been proposed and analyzed in the literature. They differ in their complexity and their efficiency to find the global minimum. Let the pixels of the macroblock in the current frame be denoted as $C(x + k, y + l)$ and the pixels in the reference picture be denoted as $R(x + i + k, y + j + l)$. The following cost functions have been proposed in the literature:

(a) The Mean Absolute Difference (MAD), or Mean Absolute Error (MAE) cost function is defined as:

$$MAD(i, j) = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} | C(x+k, y+l) - R(x+i+k, y+j+l) | \qquad (2.4)$$

where i and j are defined in $-p \le i \le p$ and $-p \le j \le p$.

We define as the best matching block, the block $R(x + i, y + j)$ for which MAD(i, j) is minimized. Thus, the coordinates (i, j) for which MAD is minimized define also the motion vector.

(b) The Mean Squared Difference (MSD), or Mean Squared Error (MSE) cost function is defined as:

$$MSD(i,j) = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} [C(x+k, y+l) - R(x+i+k, y+j+l]^2 \tag{2.5}$$

(c) The Cross-Correlation Function (CCF) is defined as:

$$CCF(i,j) = \frac{\sum_k \sum_l C(x+k, y+l) R(x+i+k, y+j+l)}{\left(\sum_k \sum_l C^2(x+k, y+l)\right)^{1/2} \left(\sum_k \sum_l R^2(x+i+k, y+j+l)\right)^{1/2}} \tag{2.6}$$

The mean absolute difference (MAD) cost function is considered as a good candidate for video applications due to its computational simplicity. The other two cost functions, MSD and CCF, can be more efficient, however they are too complex for computation and hardware implementations. Within a typical coding system, the MAD cost function performs just as well. So we choose MAD as the matching criterion.

In video coding terminology, since the match is being performed between rectangular regions, this is referred to as a block matching criterion, and search techniques to find the motion vector (u, v) that yields the smallest MAD are referred to as block matching algorithms (BMA). In the next section, we describe algorithms for block matching motion estimation.

2.3.3 Motion Vector Search Algorithm

Many block matching techniques for motion vector estimation have been developed and evaluated in the literatures. They are as follows:

- The full search algorithm;
- The three-step search algorithm;
- The two-dimensional search algorithm;
- The conjugate direction search algorithm;
- The parallel hierarchical one-dimensional search algorithm;
- The hierarchical motion estimation algorithm, etc.

In this thesis, we will focus only on the full search algorithm. It is the most obvious search technique for finding the best possible weight in the search area. The ISO MPEG specification refers to this implementation as the full search technique. The algorithm is actually quite simple. All possible displacements in the search range are evaluated using the block matching criteria, that is, using (2.1) to compute MAD(i, j) at each location in the search area. That means that no specialized algorithm is required, it is just a two-dimensional search. Within our work, we use a full search algorithm in the subsequent chapters.

2.4 Segmentation-based Mesh Design Scheme for Video Compression

In the last sections, we discussed video coding techniques and BMA for motion estimation in most video coding standards. As mentioned in Chapter 1, although BMA has a good overall performance, it usually leads to distortions such as motion block artefacts, especially when the motion is important, and the frame rate is low. For instance during forward motion compensation, the projection from frame *t*-1 to frame *t* may create some empty areas where no values are assigned, and some conflict areas where more than one value is assigned to one pixel, as shown in Figure 2.8.

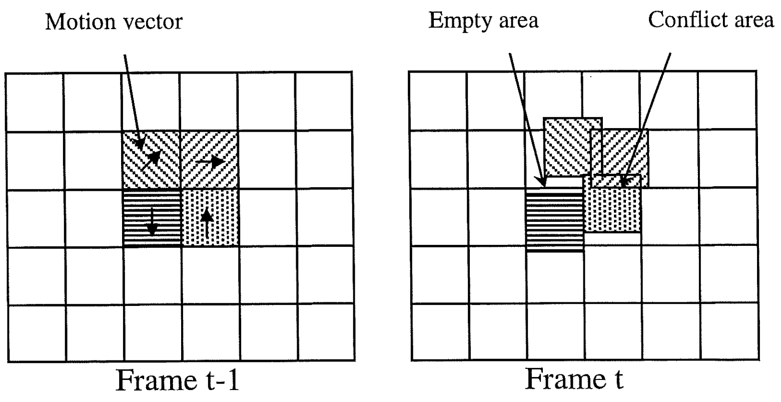


Figure 2.8 Motion block artefacts.

To overcome the drawback of block effect, some new segmentation-based coding techniques [1-5] incorporating image structure information have been proposed. The segmentation-based techniques are considered very promising. The techniques in this approach can be arranged in two classes: The first class starts from motion estimation by pixel recursive or block matching algorithm, then segments the resulting motion vector fields. The second class first segments an image into homogeneous regions, then motion estimation and compensation are performed on the basis of region. With this approach, contours in segmentation maps and motion parameters have to be transmitted to the receiver for reconstructing images. Compared with block-based motion compensation, segmentation-based motion compensation has two advantages: one is that it produces fewer motion parameters and the other is that no block artefacts are created in reconstructed images. However, the performance of this approach is limited by the following facts:

1) Contours of segmentation maps may require a significant number of bits to be transmitted.
2) A commonly used motion model cannot accurately describe complex motion in a region of irregular shape, especially if shape deformation takes place in the region.

One way to avoid contour coding is to segment the reconstructed frame (image) at time *t*-1, instead of the original frame at time *t* on both encoder and decoder, so that the only information that has to be transmitted are motion vectors. However, in the compensation procedure, the projection of frame *t*-1 into frame *t* may create some empty areas and conflict areas, similar to BMA as mentioned above (Figure 2.9). That is representative of the second drawback of the segmentation-based approach. The conflict and empty areas, existing even in BMA technique, require a large proportion of the total codec bit-rate.

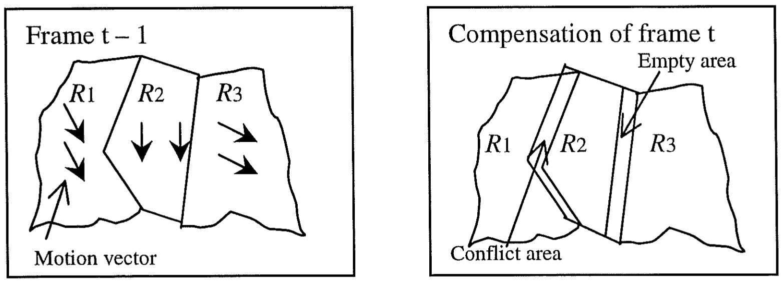


Figure 2.9 Problems of motion compensation based on region.

Therefore, in order to reach an acceptable coded image quality at very low bit-rate, more appropriate modelling of motion field is needed. One of the promising approaches is compensation based on control nodes interpolation [5-7, 11-13]. Such nodal approaches are used for global deformation estimation as well as for piecewise bilinear transformation estimation, where images are partitioned in a set of patches. Two types of mesh have been used for image partition, namely quadrangular mesh [5-7] and triangular mesh [11-13]. The main advantages of this approach reside in the following facts:

1) This approach is capable of modelling shape deformation as well as translational motion.
2) It preserves continuity and connectivity in compensated images, thus avoiding the problem of empty and conflict areas.

Compare with quadrangular mesh, triangular mesh is more flexible. Triangular mesh is usually built from a spatially uniform distribution of nodes. Recently, triangular mesh has been defined in a content-adaptive way in order to improve the motion estimation quality and decrease the motion representation cost.

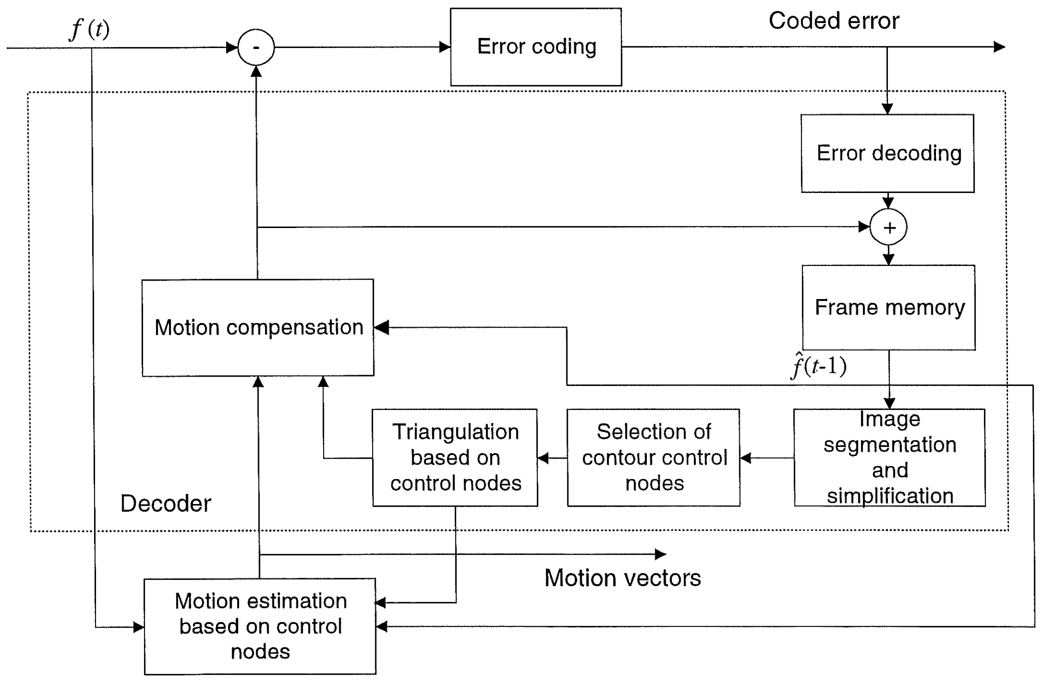

$f(t)$
-
Error coding
Coded error
Error decoding
+
Frame memory
$\hat{f}(t\text{-}1)$
Motion compensation
Triangulation based on control nodes
Selection of contour control nodes
Image segmentation and simplification
Decoder
Motion vectors
Motion estimation based on control nodes

Figure 2.10 Segmentation-based triangular mesh design scheme.

Based on the above analysis, in this thesis we introduce the principle of compensation based on active triangular mesh into segmentation-based scheme. The general structure of this scheme is described in Figure 2.10.

Let $f(t)$ and $\hat{f}(t)$ denote, respectively, the original frame and the reconstructed frame at time t. In the new scheme, reconstructed frame $\hat{f}(t - 1)$, is first segmented into homogeneous regions. Since $\hat{f}(t - 1)$ has been known in the receiver, the contours in the segmentation map do not need to be transmitted. The points where two or more contours intersect and where contours have great curvature are taken as *control nodes* of contours. An active triangular mesh can be obtained from these control nodes. Motion and deformation estimation is then performed from $\hat{f}(t - 1)$ toward $f(t)$ on the basis of control nodes. That is to say, each control node is associated with a motion vector. The motion and deformation in a triangle are described by an affine model based on those motion vectors of control nodes, which are vertices of triangles. Since the control nodes are selected on image contours, so that the active mesh structure is image content-adaptive, the segmentation map of reconstructed image at time t-1, denoted as $S(t - 1)$, can be approximated by triangles. Because the affine model can describe deformation between two triangles of current frame and reference frame, and also preserves image connectivity, so from $\hat{f}(t - 1)$ and the motion vector field we can reconstruct $f(t)$, as shown in Figure 2.11. Hence, the problem of empty and conflict area can be avoided.

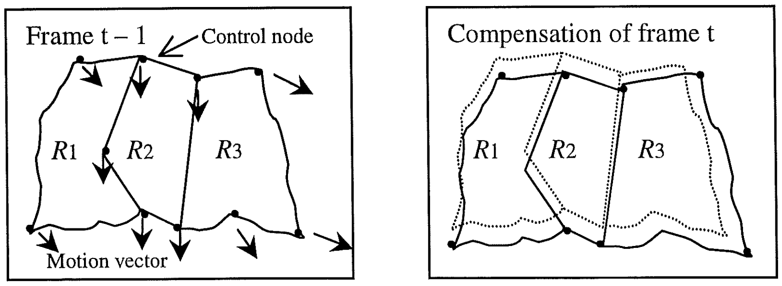


Figure 2.11 Compensation of motion and deformation based on contour control nodes.

Because the triangular mesh structure is content-adaptive, some flat area, especially the background regions, contain no control nodes, so compared with block numbers of BMA, the number of control nodes is much less. To further reduce the control nodes, we also propose an algorithm of triangle fusion. That is to say, within a frame, adjacent triangles that have similar statistical properties can be fused together. Thus, some nodes of these triangles can be deleted, which means that even less motion vectors are needed, or that the bit-rate is further reduced.

Moreover, to improve image quality of reconstructed frames, an iterative algorithm [11-13] based on minimising prediction error is used to refine the motion vectors of control nodes. Such approach could improve subjective and objective quality of reconstructed images.

Chapter 3. Active Mesh Model and Generation

3.1. Existing Transformation Models

Motion compensation methods can be defined as techniques that divide images into local regions (blocks or patches) and estimate for each region a set of motion parameters. The procedure that synthesizes the predicted image of the nth frame $I_n(x, y)$ from the decoded image of the previous frame $I_{n-1}(x', y')$ can be regarded as an image warping [22] process. This process can be written as

$$I_n(x, y) = I_{n-1}(x', y') \qquad (3.1)$$

where the geometric relationship between $I_n(x, y)$ and $I_{n-1}(x', y')$ is defined by the displacement functions $u(x, y) = x' - x$ and $v(x, y) = y' - y$. When (x', y') is not a sampling point of the image, the intensity value $I_{n-1}(x', y')$ is obtained by interpolation.

In BMA, where the displacement is supposed to be translational, the transformation functions for the pixels in the ith block of the image are

$$\begin{aligned} u_i(x, y) &= x' - x = u_i \\ v_i(x, y) &= y' - y = v_i \end{aligned} \qquad (3.2)$$

in which (u_i, v_i) is the estimated motion vector of the ith block.

Various motion compensation methods can be composed by adopting different transformation functions. It is obvious that the properties of the transformation functions are important factors that determine the performance of these motion compensation methods. What is required for the transformation function is not only to approximate the original motion vector field precisely with a smaller number of parameters, but also with a smaller amount of computation. The problem of adopting the proper transformation function is difficult since a complicated transformation function, which gives a very accurate approximation, is generally expensive in computation.

Image warping can be considered as a sum of the following three processes:

a) computation of the motion parameters (displacement functions),
b) computation of motion compensation, and
c) interpolation of intensity values.

Among these factors, b) and c) are important since they are performed for each pixel. The computation cost of b) depends on the existence of an effective scan line algorithm, which scans the image computing x' and y' for each pixel. We will discuss them in the next chapter.

In the following discussions, we assume that the patches are polygons. The motion parameters are estimated by first extracting the motion vectors of the vertices of a patch and then computing the parameters of the patch from these motion vectors. Considering the above-mentioned conditions, from among many of the transformation functions studied in the field of computer graphics [22], we examine the following three functions.

1) Affine transformation: the transformation functions for the pixels included in the ith patch of the image are

$$u(x, y) = a_1 + a_2x + a_3y$$
$$v(x, y) = a_4 + a_5x + a_6y \qquad (3.3)$$

where $a_1 \sim a_6$ are the six motion parameters of the patch. Since there are six degrees of freedom in this transformation, the parameters can be determined from motion vectors of three vertices. Therefore the patches of this transformation are triangles. One of the advantages of this method is that the cost in computing the motion parameters remains low regardless of the shape of the patch. An effective scan line algorithm exists for this transformation, which requires only two additions for each pixel.

2) Bilinear transformation: the transformation functions are

$$u(x, y) = a_1 + a_2\,x + a_3\,y + a_4xy$$
$$v(x, y) = a_5 + a_6\,x + a_7\,y + a_8xy \qquad (3.4)$$

Where $a_1 \sim a_8$ denote the eight parameters of this transformation. Since there are eight degrees of freedom, this transformation requires a quadrilateral patch. However, only rectangular patches are allowable since otherwise the computation of the motion parameters becomes complicated. An effective scan line algorithm with two additions per pixel exists for this transformation.

3) Perspective transformation: the transformation functions are

$$u(x, y) = (a_1 + a_2\,x + a_3y)/(\,1 + a_7\,x + a_8y)$$
$$v(x, y) = (a_4 + a_5\,x + a_6y)/(\,1 + a_7\,x + a_8y) \qquad (3.5)$$

where $a_1 \sim a_8$ denote the eight parameters of this transformation. As in bilinear transformation, the patches are quadrilaterals. The disadvantage of this transformation is the computational cost of the scan line algorithm: it requires two divisions for each pixel. Despite this disadvantage, that transformation is used in many systems because of the fact that it describes the motion of a rigid planar patch when perspective projection is assumed.

Some algorithms [22] that approximate the transformation function with efficient computation have been proposed.

3.2. Affine Transformation Mesh Model

The choice of a particular transformation model for motion estimation depends on the considered application. Within the context of coding, the two-dimensional motion model should be able to represent many kinds of motions with low representation and computation costs. The motion model we chose for this work is a piecewise affine transform. Such transformation is content adaptive, and can mimic higher-order transformation when many patches are placed in a region where deformation is complex. For each point *(x, y)* of a considered triangle e, the corresponding transformed position *(x', y')* is given by (3.3)

i.e.

$$d(x,y) = \begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_6 \end{bmatrix} \qquad (3.6)$$

where the a_i coefficients are the six motion parameters of the considered triangle e. As discussed in the last section, the motion parameters are determined from motion vectors of triangle vertices.

In order to express the displacement d(x, y) as a function of nodal vectors, let us consider that (3.6) is available in each vertex (x_1, y_1), (x_2, y_2), (x_3, y_3) of e:

$$\begin{bmatrix} u(x_1,y_1) \\ u(x_2,y_2) \\ u(x_3,y_3) \\ v(x_1,y_1) \\ v(x_2,y_2) \\ v(x_3,y_3) \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & 0 & 0 & 0 \\ 1 & x_2 & y_2 & 0 & 0 & 0 \\ 1 & x_3 & y_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_1 & y_1 \\ 0 & 0 & 0 & 1 & x_2 & y_2 \\ 0 & 0 & 0 & 1 & x_3 & y_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} \Phi & 0 \\ 0 & \Phi \end{bmatrix} [a]^e \quad (3.7)$$

By rewriting the vector $[a]^e$ as a function of $[\Phi]^{-1}$, Eq. (3.6) becomes

$$d(x,y) = \begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix} \begin{bmatrix} \Phi^{-1} & 0 \\ 0 & \Phi^{-1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ v_3 \end{bmatrix} \quad (3.8)$$

where

$$[\Phi]^{-1} = \frac{1}{\det(\Phi)} \begin{bmatrix} x_2y_3 - x_3y_2 & x_3y_1 - x_1y_3 & x_1y_2 - x_2y_1 \\ y_2 - y_3 & y_3 - y_1 & y_1 - y_2 \\ x_3 - x_2 & x_1 - x_3 & x_2 - x_1 \end{bmatrix} \equiv \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_2 & \beta_3 \\ \gamma_1 & \gamma_2 & \gamma_3 \end{bmatrix}$$

and $\det(\Phi) = x_2y_3 + x_3y_2 + x_1y_2 - x_2y_1 - x_3y_2 - x_1y_3$.

Finally, the displacement d(x, y) can be expressed as follows:

$$d(x,y) = \sum_{k=1}^{3} \varphi_k(x,y) d_k \quad (3.9)$$

where, $\varphi_k(x, y) = \alpha_k + \beta_k x + \gamma_k y$ (3.10)

and α_k, β_k and γ_k depend on vertices location. Then it has been shown that d(x, y) can be formulated as a function of nodal displacements and vertices location. In order to know the motion at each point, we have to estimate motion at nodal points. Thus, motion estimation is realised in two steps: first nodal points are detected, and then displacements at these points are estimated.

3.3 Content-based Mesh Generation

In the previous sections, we proposed a segmentation-based mesh design scheme, and we chose the affine mesh model for motion estimation. The following work in this section will be focused on segmentation based mesh design, which is one of the most important sections throughout our work.

This section will be divided into 4 parts as follows:

1. Image segmentation;
2. Image simplification;
3. Selection of control node;
4. Triangulation with triangle fusion.

3.3.1 Image Segmentation

Image segmentation is an essential step for many image analysis tasks, such as object recognition, computer vision and image compression as in our case. The goal of image segmentation is to partition an image into homogeneous regions and locate the contours of the regions as accurately as possible. Among a large number of techniques and algorithms for image segmentation, those based on watershed transformation can potentially provide accurate segmentation with very low computational cost. In this section, we will perform image segmentation by using a multiscale gradient algorithm based on watersheds [8].

Watershed transformation starts with the gradient of the image to be segmented. It views the gradient image as a three-dimensional surface (3-D) where gradient values act as surface heights. Intensity edges in the image to be segmented generally have high gradient values which appear as *watershed lines* (also known as mountain ridges) on the 3-D surface, while the interior of each region usually has a low gradient value which is considered as a *catchment basin* on the 3-D surface. The watershed lines partition the gradient image into different catchment basins which correspond to homogeneous regions of the image to be segmented.

Watershed transformation involves a search for watershed lines in the gradient image. Therefore, the performance of a watershed-based image segmentation method depends on the algorithm used to compute the gradient. Conventional gradient algorithms exhibit a serious weakness for watershed-based image segmentation. A conventional gradient operator, such as the first partial derivation of Gaussian filter [32] and morphological gradient operators [33], produce too many local minima because of noise quantization error within homogenous regions, which result in over-segmentation. And conventional gradient operators produce low gradient values at blurred edges, even though the intensity change between the two sides of an edge may be high. In [8], a multi-scale gradient algorithm based on morphological operators for watershed-based image segmentation was proposed. This algorithm efficiently enhances blurred edges while being very robust to multi-edge interactions. This enhancement increases the gradient value for blurred edges above those caused by noise and quantization error. So we can eliminate the local minima produced by noise and quantization error.

Basic Morphological operations

Before discussing multi-scale Gradient algorithm, we first introduce the basic morphological operations. Most morphological operations can be defined in terms of two basic operations, *erosion* and *dilation* [34]. Suppose the object **X** and the *structuring element* **B** are represented as sets in two-dimensional Euclidean space. Let B_x denote the translation of B so that its origin is located at x. Then the erosion of X by B is defined as the set of all points x such that B_x is included in X, that is,

$$\text{Erosion:} \qquad X \ominus B = \{ x: B_x \subset X\} \qquad (3.11)$$

Similarly, the dilation of X by B is defined as the set of all points x such that B_x hits X, that is, they have a nonempty intersection:

Dilation: $X \oplus B = \{ x: B_x \cap X \neq \phi \}$ (3.12)

From the definition, we know that erosion is a shrinking operation, whereas dilation is an expansion operation.

The Multiscale Gradient Algorithm

Many gradient operators and edge detection algorithms have been based on the step edge model. However, ideal step edges do not exist in natural images since every edge is blurred to some extent. A blurred edge can be modelled by a ramp and the intensity change between two sides of the edge is referred to as *edge height.* For a ramp edge, the output of a conventional gradient operator, such as Prewitt gradient [34], is the slope of the edge. Hence, the ramp edge cannot be separated from noise and quantization error by thresholding if the slope of the edge is small. Figure 3.1 shows a 1-D example where a ramp edge and step edge are digitized in order to illustrate the effect of quantization error. The ideal gradient operator for watershed transformation is the one whose output is equal to the input edge height, but not the edge slope.

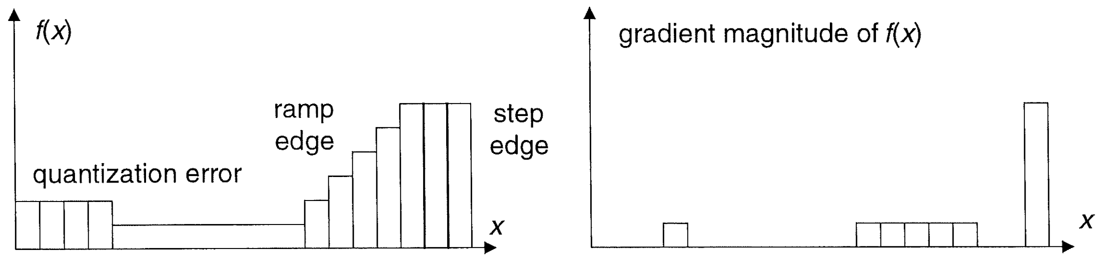


Figure 3.1 Output of conventional gradient operators.

The morphological gradient operators used in reference [32] can be described as

$G(f) = (f \oplus B) - (f \ominus B)$ (3.13)

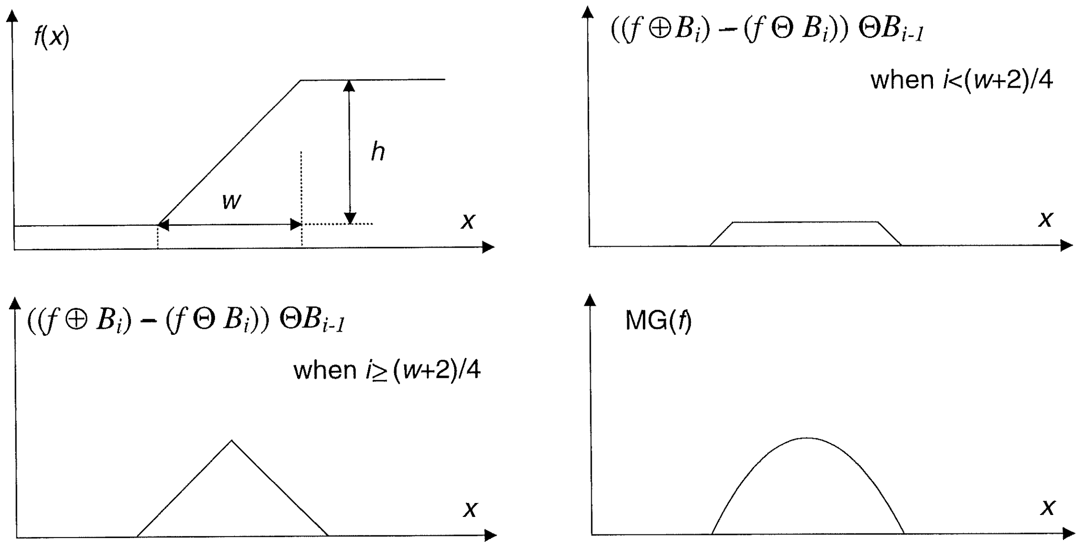
This gradient operator is referred to as mono-scale morphological gradient operator. Its performance depends on the size of structuring element B. If B is large, the output of this gradient operator for a ramp edge is equal to the edge height. Unfortunately, large structuring elements result in serious interaction among edges which may lead to gradient maxima not coinciding with edges. However, if the structuring element is very small, this gradient operator has a high spatial resolution, but produces a low output value for ramp edges.

In order to exploit the advantage of both small and large structuring elements, we propose a multiscale morphological gradient algorithm. Let B_i, for $0 \le i \le n$, denotes a group of square structuring elements. The size of B_i is $(2i + 1) \times (2i + 1)$ pixels, i.e. B_0 contains only one pixel and B_1 is a 3×3 square and so on. The multiscale gradient is defined by

$$MG(f) = \frac{1}{n}\sum_{i=1}^{n}[((f \oplus B_i) - (f \Theta B_i))\Theta B_{i-1}] \qquad (3.14)$$

For a step edge, the operation $((f \oplus B_i) - (f \Theta B_i))\ \Theta B_{i-1}$ produces a line of two pixels wide which coincides with the edge. The intensity (height) of the line is equal to the edge height. Hence, the multiscale gradient algorithm is equivalent to the mono-scale morphological gradient operator in this case. In practice, it is more robust to noise due to the averaging operation used in the algorithm.

For a ramp edge, we denote respectively the edge width and height by w and h, as shown in Figure 3.2. The operation $((f \oplus B_i) - (f \Theta B_i))\ \Theta B_{i-1}$ produces a line coinciding with the edge. The cross section of the line appears as a trapezoid if $i < (w + 2) / 4$ and as a triangle otherwise. The width of the bottom side of the trapezoids or triangles is always equal to $w + 2$ pixels. The height of the trapezoid is $2ih / w$ and that of the triangle is $h(w + 2) / (2w)$, which are greater than the edge slope h / w. The value of MG(f) approaches to $h(w + 2) / (2w)$ if n is large enough. Therefore, the multiscale algorithm responds effectively to ramp edges without enlarging edges.



f(x)
h
w
x
((f ⊕ B_i) − (f ⊖ B_i)) ⊖B_{i-1}
when i<(w+2)/4
x
((f ⊕ B_i) − (f ⊖ B_i)) ⊖B_{i-1}
when i≥ (w+2)/4
x
MG(f)
x

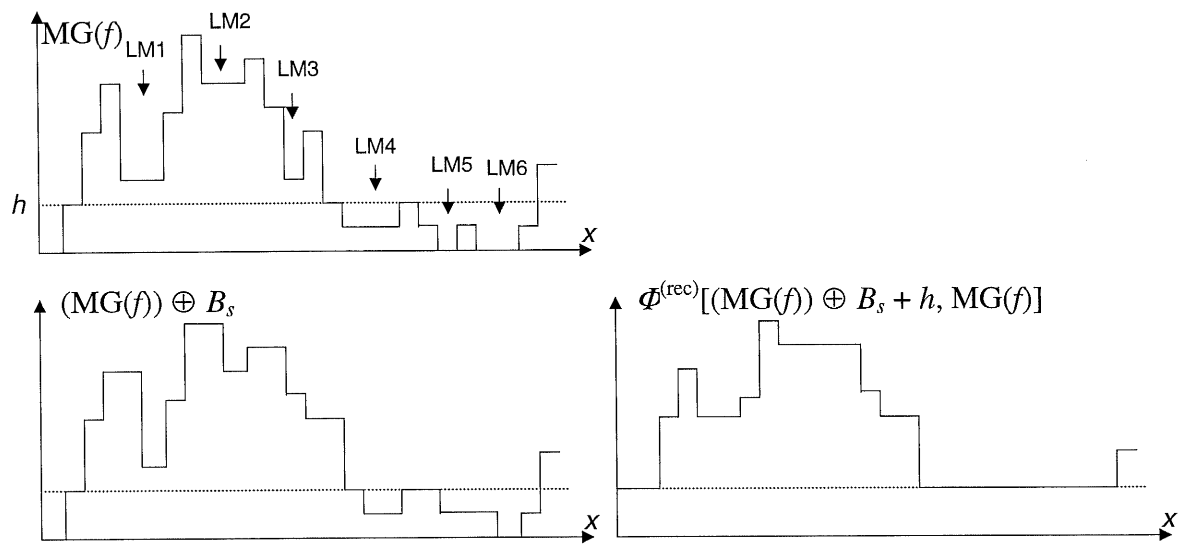
Figure 3.2 Result of the multiscale gradient algorithm for ramp edges.

Elimination of Small Local Minima

Small local minima are defined as local minima consisting of a small number of pixels or having a low contrast with their neighbors. This kind of local minima in gradient images is generally caused by noise or quantization error, and therefore should be eliminated.

Local minima consisting of a small number of pixels are eliminated by dilation with a square structuring element B_s of 2×2 pixels, denoted by $(\mathrm{MG}(f)) \oplus B_s$. To remove the local minima with a low contrast, a constant denoted by h is first added to the dilated gradient image. Then the local minima with a contrast lower than h can be filled using the reconstruction by erosion of MG(f) from $(\mathrm{MG}(f)) \oplus B_s + h$. Hence, the final gradient image can be expressed as

$$\Phi^{(\mathrm{rec})}[(\mathrm{MG}(f)) \oplus B_s + h, \mathrm{MG}(f)].$$



MG(f)
LM1
LM2
LM3
LM4
LM5
LM6
h
x
$(\mathrm{MG}(f)) \oplus B_s$
x
$\Phi^{(\mathrm{rec})}[(\mathrm{MG}(f)) \oplus B_s + h, \mathrm{MG}(f)]$
x

Figure 3.3 Elimination of small local minima.

This algorithm is illustrated in Figure 3.3, where MG(f) has six local minima. Local minima (LM) 3 and 5 consisting of one pixel are removed by dilation, while local minima 2, 4 and 6 having a contrast lower than h are eliminated using reconstruction by erosion. Local minimum 1 is not removed because it is both wide and deep. The constant h is used to control the number of segmentation regions. As h increases, the number of regions produced decreases. The reconstruction by erosion fills all of the local minima where the contrast is lower than h, irrespective of their absolute values. However, thresholding removes only the minima with low absolute value.

The above-mentioned algorithm followed by watershed transformation can produce meaningful image segmentations. The experimental results for image sequences Miss America and Forman are shown in section 3.3.4.

3.3.2 Image Simplification

Principle

We have obtained image contours after image segmentation as described in the last section. Our goal is analyzing the contours and extracting the control nodes, which we need for motion estimation. However, the control node extraction of image contour is based on the single-pixel-wide contour segment. Thus, image simplification is necessary before the extraction of control nodes. In this section, we will briefly describe the process of image simplification, or thinning of image.

In general, a thinning operator extracts a network of thin curve from a bilevel image. The thin curve should be placed "about in the middle" of the objects and should describe the "skeleton" of the objects. This skeleton characterizes the shape of the object regions. General strong criteria about the quality of the resultant images do not exist because of the broad variety of object shapes and specific aims of image processing. However the following requirements specify some properties which are widely accepted to be typical for skeletons:

(1) The skeleton should consist of curves with a width of single image pixel (curve thickness property);
(2) The topological connected relations of the skeleton should be identical to those of the original image, i.e. the number of 8-components of object segments should remain the same (topology preserving property).
(3) The skeleton curves should lie in the middle of the objects (medial axis property).
(4) In the case of thick or rugged objects there should not be too many irrelevant "skeleton branches" (noise robustness property);
(5) The skeletonization process should converge to a stable skeleton after a certain number of iterations (convergence property).

A skeleton procedure deletes step-by-step all "superfluous" object points in an image for achieving these aims. The medial axis property (3) leads to the conclusion that only border points of objects should be deleted during the repeated iterations of a skeletonization procedure.

Criterion and Method

We shall introduce two kinds of connectivity before defining the simplification criterion. Conceptually, image boundaries can be found by tracing the connected edges. On a rectangular grid a pixel is said to be *four-* or *eight-connected* when it has the same properties as one of its nearest four or eight neighbours, respectively (Figure 3.4).

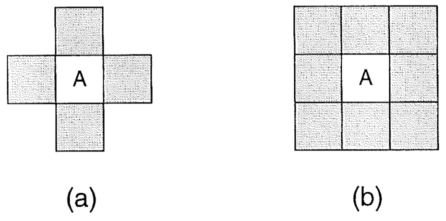


Figure 3.4 Image connectivity, (a) 4-connected (b) 8-connected .

The *connectivity number* of four- or eight-connected segments inside a placed 3×3 window is a value of a window function which can be used as deletion criterion. If this value is equal to 1, then the current point p can be deleted, provided that p is no curve end point.

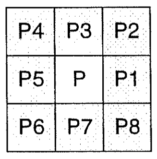


Figure 3.5 Eight neighbour points of a given point p.

The function of *connectivity number* $F(p)$ can be defined as follows. Given a point p, we define its 8 neighbour pixels as $p_1, p_2, \ldots, p_8$, as denoted in Figure 3.5, where p_1 is the right neighbour, and p_2 to p_8 are obtained by counter-clockwise as shown. And then, $F(p)$ for 8-connected and 4-connected segments are:

$$F(p)_4 = \sum_{k=1,3,5,7} (p_k - p_k p_{k+1} p_{k+2})$$
$$F(p)_8 = \sum_{k=1,3,5,7} [(1-p_k) - (1-p_k)(1-p_{k+1})(1-p_{k+2})] \quad (3.15)$$

Note that $p_1 \sim p_8$ are all binary numbers, the value is 1 if pixel intensity value is greater than 0, and the value is 0 if pixel intensity value is equal to 0. So before starting with the thinning, the program converts the image intensity to 0, 1 values. And in our case, we choose the 8-connected method.

Because the image contours obtained from the last section are almost single pixel segments, the simplification procedure is very easy and fast to implement.

3.3.3. Selection of Control nodes

This section is concerned with the choice of a representative set of points whose motion will be estimated. These nodal points play the same role as control grid in [5]. In this thesis, we choose the control nodes by means of the following method.

Given a segmentation map, control nodes are extracted for motion prediction in video compression. In order to get an accurate prediction, control nodes should be situated on the contours of the segmentation map. In this algorithm, the border of the image is also considered as contour and every region is therefore surrounded by one or more closed contours. A closed contour may not intersect with other contours.

The control nodes in this algorithm consist of two classes, which will be described in details in the following sections.

First Class of Control Nodes

The first class of control nodes is extracted from contour intersections because each contour intersection is associated with more than two regions. This class of control nodes can be easily determined from the segmentation map. However, it should be noticed that:

(i) Four corner points will also be defined as first class of key nodes;

(ii) A closed contour may not intersect with any other contours.

Therefore, the algorithm to determine this class of control nodes is performed in the following steps:

(1) Set four corner points as control nodes;

(2) Search for all contour intersections in the segmentation map;

(3) If a closed contour does not intersect with any other contours (isolated contour), the first contour point according to the scanning order is taken as a control node.

These three kinds of nodes are illustrated in Figure 3.6

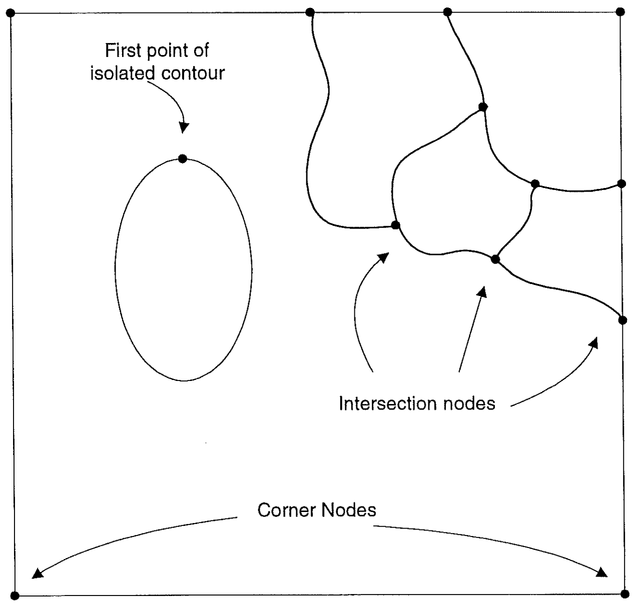


Figure 3.6 The first class of Control nodes.

The first step is so easy that we just set the four corner points as control nodes. For the second step, a function of *Cross Number* will be used to detect the intersections of image contours. Cross number, is the value of a window function *CRN*(p). Given a point p, as illustrated in Figure 3.5, it has eight neighbours, from p_1 to p_8. *CRN*(p) is the summary of transition number from $p_1 \sim p_8$, which can be expressed in the following equation.

$$CRN(p) = \frac{1}{2}\sum_{k=1}^{8} | p_k - p_{k+1} | \qquad (3.16)$$

where $p_9 = p_1$.

After obtaining the cross number of non-zero pixels, we can easily determine if a pixel is an intersection or not by using the following criteria.

CRN(p) = 0: p is an isolate point;
CRN(p) = 1: p is an start point;
CRN(p) $\geq$ 2: p is an intersection point.

For the third step of determining the first class of control nodes, the major point is how to detect the isolated contours. The method we used is to delete all the contours that intersect with other contours, so only isolated contours are left. To achieve this goal, we shall delete all intersection points and their neighbours within a 3×3 window at first. So all intersected contours are disjointed into non-closed segments. After that, we eliminate these segments pixel by pixel using contour tracking method.

Because of the simplification of image contour, all the segments are single-pixel-wide. So the non-intersection points of the contour image have only one neighbour inside a 3×3 window. It is easy to trace and delete the open segments beginning with a start point. The flow chat of tracing and eliminating is illustrated as in Figure 3.7.

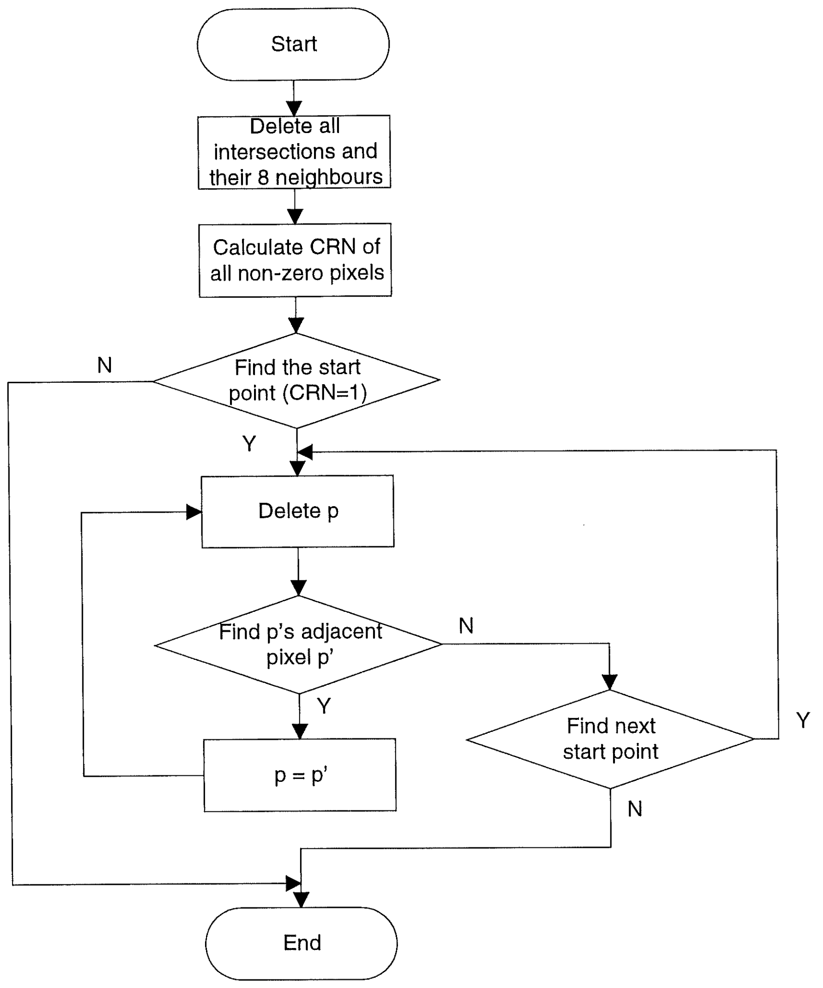

Start
Delete all intersections and their 8 neighbours
Calculate CRN of all non-zero pixels
Find the start point (CRN=1)
N
Y
Delete p
Find p's adjacent pixel p'
N
Y
p = p'
Find next start point
Y
N
End

Figure 3.7 Flow chat of segments tracing and eliminating.

Finally, we have only all the isolated closed contours, their first points will be detected using a scan line method, and be set as first kind of control nodes.

Second Class of Control Nodes

The first class of control nodes may not be dense enough for accurate motion prediction. Since intersection nodes alone cannot accurately represent image contour, so the high curvature points should be detected and set as control nodes to approximate contour segments. High curvature points are the inflection points on the image contour. On the other hand, the distance between two adjacent control nodes of the first class may be too large. Hence, additional control nodes should be inserted between them.

The principle of detecting and inserting high curvature nodes is the following:

(1) Approximate the contour segments as straight lines. That is, straight lines are drawn between every two consecutive control nodes.

(2) For each contour point between two adjacent control nodes, the separation, or distance from the point to the approximated straight line is calculated. We call this distance the approximated error *E*. If the maximum value of E exceeds a given threshold T_e, a new control node is inserted at the point where the maximum approximation error takes place.

(3) If a contour segment is connected to only one control node, namely single-node-connected contour, then the distance from each point of this contour to the single control node is calculated. The point that has maximum distance is considered as another control node (see Figure 3.8).

(4) Repeat recursively step (1) and (2) until the maximum approximation error for every two adjacent control nodes is smaller than T_e.

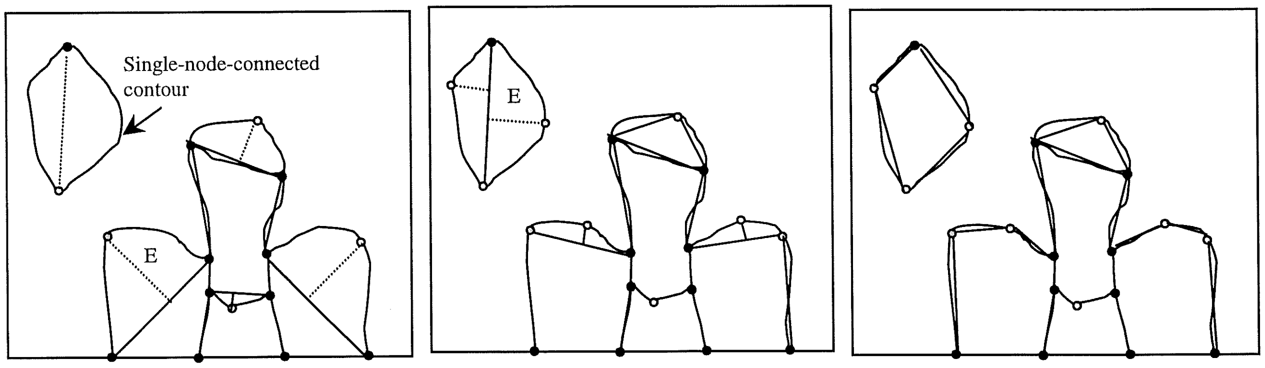
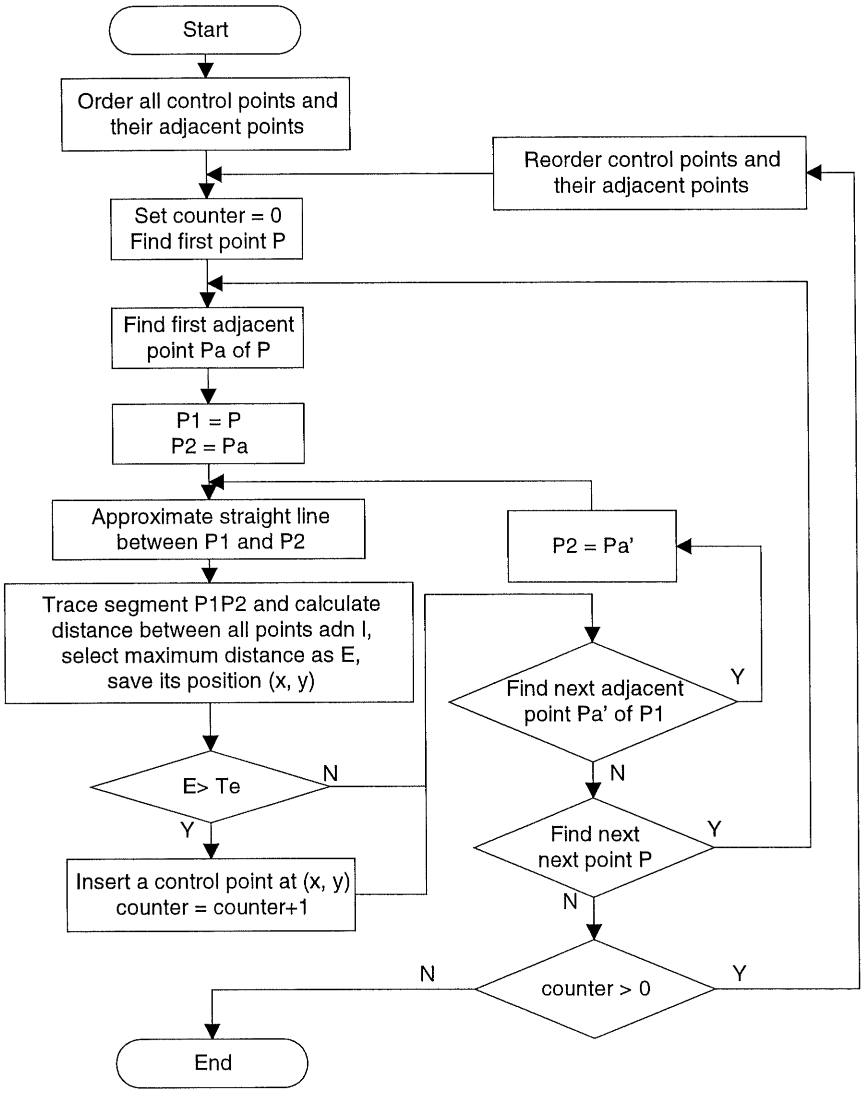


Figure 3.8 Insert high curvature nodes.

This process is illustrated in Figure 3.8. For the implementation, an iterative method is applied. The flag *counter* is used to indicate the number of inserted new nodes during one iteration loop. The process will terminate when the counter is equal to zero. The detailed procedure is shown by the flow chat in Figure 3.9.



Start
Order all control points and their adjacent points
Reorder control points and their adjacent points
Set counter = 0
Find first point P
Find first adjacent point Pa of P
P1 = P
P2 = Pa
Approximate straight line between P1 and P2
P2 = Pa'
Trace segment P1P2 and calculate distance between all points adn l, select maximum distance as E, save its position (x, y)
Find next adjacent point Pa' of P1
Y
N
E> Te
N
Y
Find next next point P
Y
Insert a control point at (x, y) counter = counter+1
N
N
counter > 0
Y
End

Figure 3.9 Flow chat of inserting high curvature control points.

Intersections and high curvature points may be sufficient to represent image contour, however, if the length of an image segment is very large, the area between two distant control nodes is generally difficult to predict with these two control nodes. Thus it is necessary to insert some nodes between these two distant nodes. The principle is as follows: (Figure 3.10)

(1) Calculate the contour length L between every two adjacent control nodes.

(2) If L is equal to or larger than a predetermined threshold $2*T_l$, insert N new control nodes (with equal distance) between the adjacent control nodes so that $L/N > T_l$ and $L/(N+1) \leq T_l$.

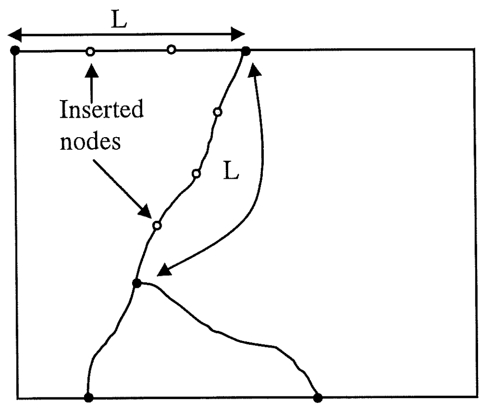


Figure 3.10 Insert nodes to long segments.

Since $L/N > T_l$ and $L/(N+1) \leq T_l$, so $(L/T_l - 1) \leq N < L/T_l$. Because N should be equal to or larger 1, so we take the threshold strategy as : $L \geq 2*Tl$. And to determine the value of N, we use the following criterion:

$$N = \lfloor L/Tl + 0.5 \rfloor - 1, \text{ where } \lfloor x \rfloor \text{ denotes the largest integer smaller or equal than } x.$$

Since L/Tl may not be an integer, and its residual may be ≥ 0.5, we add a constant 0.5 to L/Tl in order to round it toward its nearest integer. The operation – 1 at the end of the equation is due to the interval number between two distant nodes is N + 1.

Until now two kinds of control nodes have been selected. However, two control nodes, especially the contour intersections, may be close to each other. The motion described by these intersections may be highly coherent and it is not efficient to take all of them as control nodes. Hence, a post-processing should be taken to eliminate these unnecessary nodes.

(1) Determine the distance between every two adjacent nodes (which are connected by a contour segment);

(2) If the distance between two adjacent nodes is too small (less than a threshold T_d), one of the nodes is not considered as a control node. All other nodes are taken as control nodes.

We have all the control nodes so far, so the next step is triangular mesh generation based on these control nodes, which will be shown in the following section.

3.3.4. Triangular Mesh Generation with Triangle Reduction

After a successful selection of control nodes, the mesh generation is performed using the method of Delaunay triangulation. Delaunay triangulation is a well-known triangulation method in the field of computational geometry and provides meshes with several nice properties, e.g., it maximizes the minimal angle between triangle edges [23].

Although Delaunay triangulation has good performance, it does not take the image data into account. Ordinary Delaunay triangulation algorithm inputs the selected nodes, and outputs the Delaunay triangulation without considering image contours, so the triangle edges may cross over image contours, as shown in Figure 3.11 (a). In this case, one triangle consists of image pixels of two or more regions, but different regions may undergo different motions, so the motion information of this triangle can not be accurately estimated. To solve this problem, a constrained Delaunay triangulation should be employed to construct a content-based triangular mesh. The solution is to link every two adjacent control nodes by straight lines. These straight lines can form polygons. So the image contours are approximated by the produced polygons. Then the edges of the polygon are used as constrains in the triangulation to make sure that polygon edges become the edges of generated triangles, as shown in Figure 3.11 (b). This way, no triangle will cross over a contour, each triangle only represents one or part of one object.

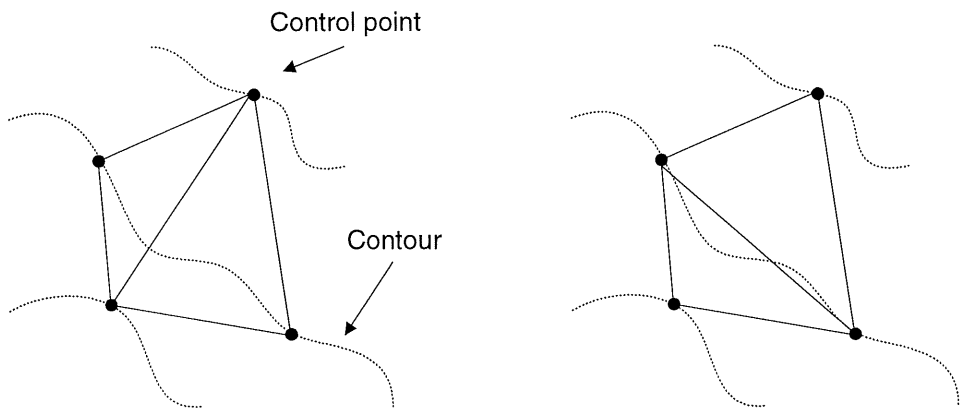


Figure 3.11 Triangulation constraint: no contour cross. (a) triangles across over a contour; (b) not across over the contour.

Triangle Fusion

After applying the constrained Delaunay triangulation method, we can obtain a content-based triangular mesh structure. Each triangle contains uniformly distributed image pixels. However, if we continue to study the properties of the triangles, we can find that some adjacent triangles are very similar, especially in the regions where we insert the second class of control nodes on the long contour segments. Because we insert the nodes at equal distance, the triangles produced by these nodes cannot accurately reveal image content, some adjacent triangles may share the same or similar properties. Thus, these similar triangles can be fused together to form a new polygon, which is image intensity homogeneous. In fact, to form this polygon, some nodes are unnecessary, since some consecutive edges may have the same or very close slopes, so these edges can be approximated by one straight line, and the nodes between the vertices of this line can be deleted. After this approximation, only vertex nodes of this polygon are kept. Based on these retained nodes and the polygon map, we reapply the constrained Delaunay triangulation method and the final triangular mesh structure is achieved.

In order to estimate the similarity between two adjacent triangles, we shall exploit the statistical properties of triangular image patches. It is well known that an image can be considered as a set of random signals. Mean and variance are two important factors to estimate a signal's statistical property. So, let x denote the image pixel series of a triangle, and μ and σ^2 denote the mean and variance of this triangle, we will then have:

$$\mu = E(x) = \frac{1}{N}\sum_{k=1}^{N} I_k \tag{3.17}$$

$$\sigma^2 = E\left[(x - E(x))^2\right] = E(x^2) - E^2(x) = \frac{1}{N}\sum_{k=1}^{N} I_k^2 - \mu^2 \tag{3.18}$$

where E and I denote the mathematics expectancy and image pixel intensity.

Then, let i and j be the number of adjacent triangles. We shall define the following criterion to determine the triangle fusion:

if $\Delta\mu = |\mu_i - \mu_j| < T_\mu$ and $\Delta\sigma^2 = |\sigma^2_i - \sigma^2_j| < T_\sigma$ then, triangle i and j can be merged.

Where T_μ and T_σ are two thresholds for mean and variance values.

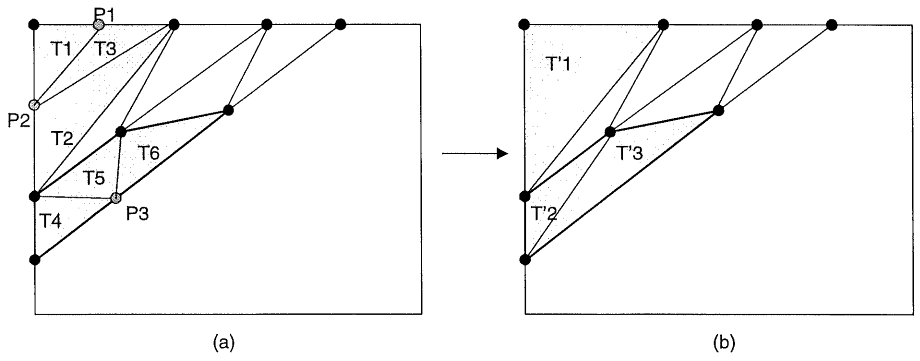
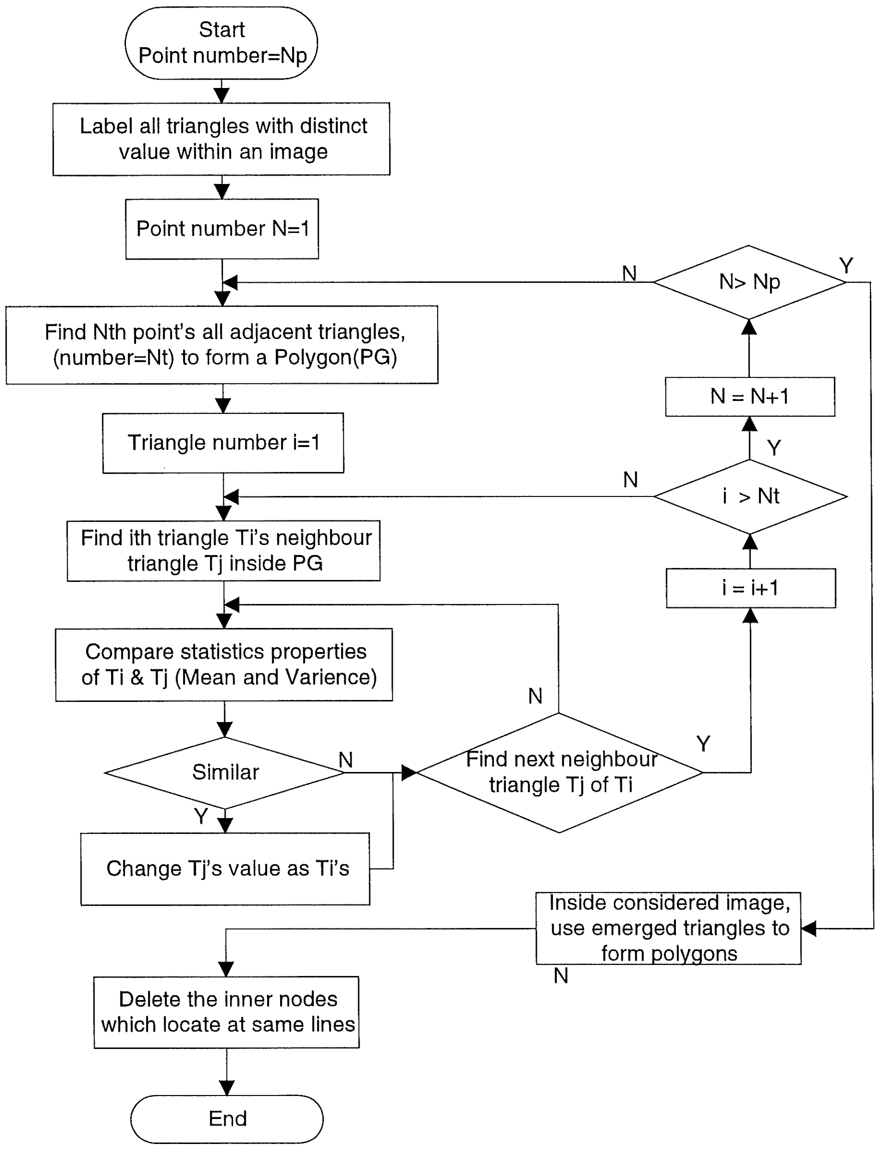


Figure 3.12 Triangle reduction. (a) before fusion (b) after fusion.

The example is illustrated in Figure 3.12, let T and P denote triangle and control node respectively. At first, we calculate the mean and variance of all triangles within an image frame. Then, we compare these two statistics for any two adjacent triangles following the above criterion. In Figure 3.12, triangle sets $\{T_1, T_2, T_3\}$ and $\{T_4, T_5, T_6\}$ satisfy the above criterion, so they are fused together and become a new polygon. After triangle emerging, some control nodes located on the same straight segments can be deleted, such as points P_1, P_2, and P_3 in Figure 3.12 (a). Then we reapply Delaunay triangulation and obtain a new mesh structure like Figure 3.12 (b). The new structure represents the same image intensity distribution, but need less control points, which leads to the low bit-rate.

The simplified flow chat of triangle fusion and node reduction is shown in Figure 3.13.



Start
Point number=Np
Label all triangles with distinct value within an image
Point number N=1
Find Nth point's all adjacent triangles, (number=Nt) to form a Polygon(PG)
Triangle number i=1
Find ith triangle Ti's neighbour triangle Tj inside PG
Compare statistics properties of Ti & Tj (Mean and Varience)
Similar
Y
N
Change Tj's value as Ti's
Find next neighbour triangle Tj of Ti
N
Y
i = i+1
i > Nt
N
Y
N = N+1
N> Np
N
Y
Inside considered image, use emerged triangles to form polygons
N
Delete the inner nodes which locate at same lines
End

Figure 3.13 Flow chat of triangle fusion and node reduction.

The result of triangle fusion and node reduction is shown in Table 3.1. The image sequences to be used for simulation are Miss America and Foreman. The image size is 352 $\times$ 288. For selecting control nodes, the three thresholds are set as: $T_d = 4$, $T_e = 5$, $T_l = 24$. For triangle fusion, the thresholds are set as: $T_\mu=5$, $T_\sigma=200$, and $T_\mu=10$, $T_\sigma=300$ for Miss America and Foreman respectively. Because of triangle fusion, the average node number is reduced from 148 to 116 for Miss America, and from 226 to 176 for Foreman, so the reduction ratio is around 22%.

Table 3.1 Node number of Miss America and Foreman.

Miss America Seq. No.	Node number Before triangle fusion	Node number after triangle fusion		Foreman Seq. No.	Node number before triangle fusion	Node number after triangle fusion
1	155	117		23	222	172
2	153	120		24	235	185
3	144	115		25	228	185
4	156	121		26	226	181
5	138	101		27	225	180
6	153	121		28	219	164
7	142	108		29	222	175
8	151	117		30	221	170
9	153	119		31	226	177
10	157	120		32	225	170
Average	148	116		Average	226	176

Figure 3.14 illustrates the generated active mesh structure, the first row represents the original images of Miss America and Foreman with contour segments, and the second row represents the mesh structure before triangle fusion, the last row represents the mesh

structure after triangle fusion. From the figures of mesh structure, it can be clearly recognized that most triangles have a uniformly distributed luminance value, so the mesh is content-adaptive.

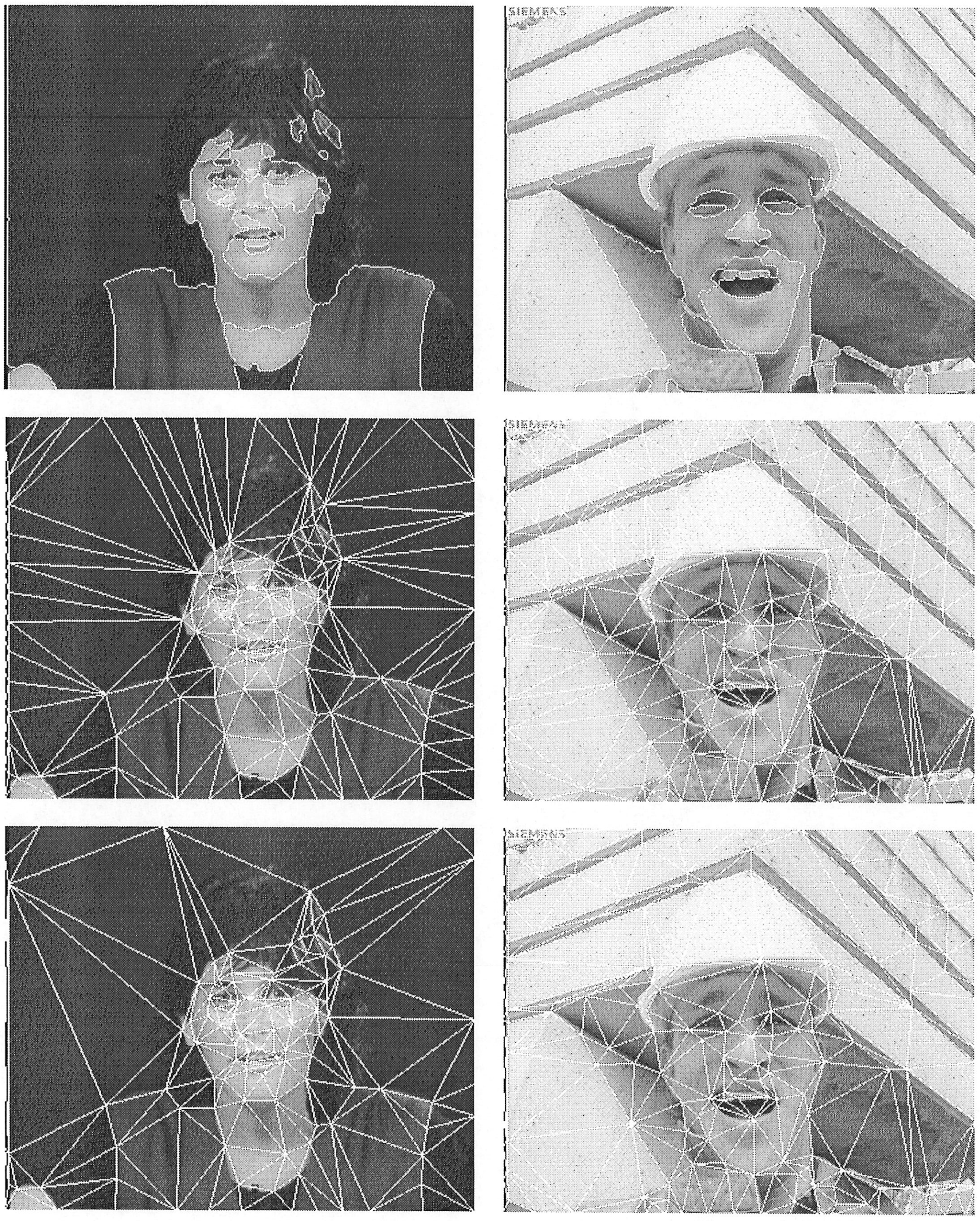


Figure 3.14 Results of triangulation for sequence Miss America and Foreman.

3.4. Spatial-Temporal Gradients Method of Mesh Design

Y. Altunbasak, and A. Murat Tekalp have proposed another content-based mesh design method [29-30], which is called the spatial-temporal gradient (STG) based approach. This algorithm also consists of a nonuniformly spaced node point selection procedure, followed by triangulation using the selected node points. The procedure aims to partition the image into triangles in such a way that a predefined function of the displaced frame difference (DFD) within each patch attains approximately the same value. An outline of the algorithm is as follows:

1. Estimate a 2-D dense motion field between the present and reference frames. Label all pixels as "unmarked";
2. Compute DFD_{avg} given by

$$DFD = \frac{\sum_{(x,y)} (DFD(x,y))^p}{K} \qquad (3.19)$$

   where the summation is over all unmarked points, K is a number of unmarked points, and p is a positive number.
3. Compute a cost function $C(x, y)$ associated with each unmarked pixel as a predefined function of spatial-temporal intensity gradients.
4. Find the unmarked pixel with the highest $C(x, y)$ which is not closer to any other previously selected node point than a prespecified distance. Label this point as a node point.
5. Grow a region about this node point until $\Sigma(DFD(x, y))^p$ in this region is greater than DFD_{avg}. Label all pixels within this region as "marked".
6. Go to 2 until a desired number of node points, N, are selected.
7. Given the selected node points, apply a triangulation procedure to obtain a content-based mesh.

Comparing the STG based approach and our intensity segmentation-based approach, we can see some differences. The STG approach uses motion intensity gradient and a cost function to determined node points. Although this method results in motion content-based triangular mesh, and the node number can be fixed, it also has some disadvantages. First, in many cases, the points with same gradient derivation may concentrate on some areas, which leads to many node points locate at some local regions. Since DFD_{avg} is computed on unmarked pixels and if image has a uniform background, after the high gradient points have been selected in the foreground area, many node points will also be detected in the background area (see result of [29]). In fact, these nodes are redundant since many triangles in background area have the same image intensity. To sum up, though the selected nodes have high gradient, but they may not lie on a region boundary (contour), the generated triangular patch may cover different regions and, as mentioned in last section, the motion vectors cannot be accurately estimated.

In contrast to the STG based method, using the proposed intensity segmentation-based mesh design method, the most selected nodes are image contour intersections and high curvature points on the contours. So after employing constrained triangulation the mesh structure nearly coincides with image contour, and each triangular patch consists of only one segmented region or part of one region. For one region, the translation and deformation can be accurately estimated, which can result in precise image prediction. Moreover, due to the triangle fusion algorithm, the node number can be reduced to a great extent. Less prediction error and small amount of motion vectors fit the low bit-rate requirement.

After mesh generation, the next step will be motion estimation for nodal points and image prediction based motion estimation and compensation, which will be described in the subsequent chapter.

Chapter 4. Motion Estimation

Successive images of a video sequence are strongly correlated and the motion estimation and compensation between consecutive frames aim at reducing the spatial-temporal redundancies by exploiting this strong correlation. Motion compensation using 2D mesh requires computation of the parameters of a spatial transformation within each mesh element (patch). It is well known that the parameters of an affine mapping can be uniquely estimated from three corresponding points (e.g. at the three vertices of a triangular patch, as described in section 3.2). Therefore, in order to estimate the affine mapping parameters for motion compensation of each patch, it suffices to estimate the motion vectors at the nodal points. If the displacements of nodal points can be precisely determined, the quality of image prediction would be perfect. So, here, the algorithm of block matching is initially applied to estimate the displacements of nodal points, then, an iteration algorithm corresponding to the refinement of the displacement vectors will be developed.

4.1. Initialization by Block Matching

The goal of this stage is to find the initial motion vector for each control node, which is obtained from the last chapter. The chosen algorithm is similar to the above-mentioned block matching algorithm (BMA).

Before starting motion estimation, we first introduce the classification of control nodes. We divide the control nodes into three kinds as follows (Figure 4.1 (a)):

1) Inner nodes (P_I). Nodes that are inside the image frame.
2) Boundary nodes (P_B). Nodes that are on the boundaries of the image frame.
3) Corner nodes (P_C). Nodes that are on the 4 corners of the image frame.

To coarsely estimate the motion vector, we first use a square block to enclose each node, and the node is located at the centre of this block. Then we search the optimal matching block in the reference image to get the translational motion vector. We also call it block matching algorithm since it is nearly the same as BMA in section 2.3. Since the block is centered at the control node, the block size must be an odd number. We take it as 15 x 15, and the search range is [-*p*, *p*], where we use *p* = 7.

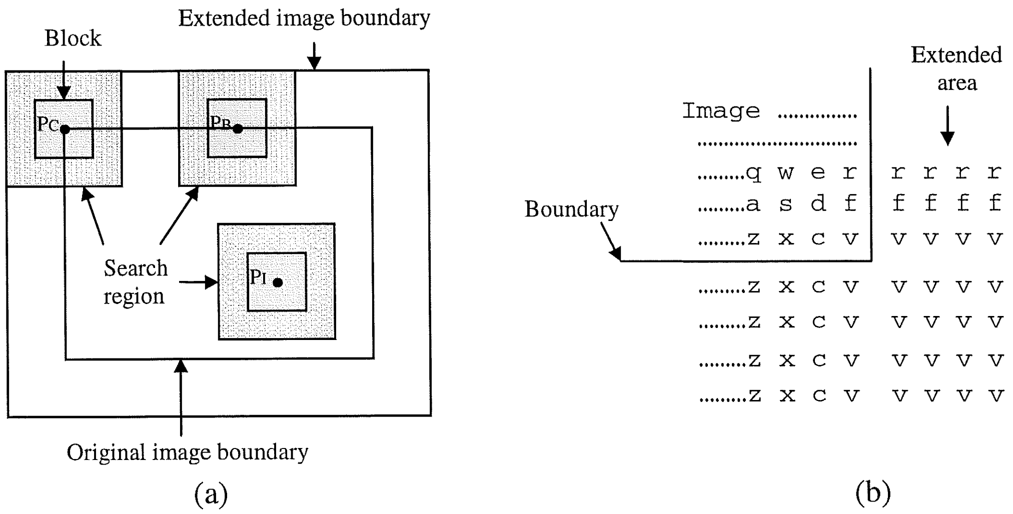


Figure 4.1 Image Enlargement.

It should be noticed that the block matching algorithm we applied is different from the traditional BMA. For conventional BMA, to keep the size and shape of the image framework, the corner nodes are assigned zero motion vectors, and boundary nodes can only move along image border, so they have only horizontal or vertical motion displacement. This method is not sufficient for some cases. For example, an object in the current frame of a video sequences, may move outside the image boundary in the future frame, or an object in a past frame may move inside a current frame. To cope with the problems of corner and boundary nodes, we propose an approach that is to enlarge the original reference image as illustrated in Figure 4.1. The extending method is shown in Figure 4.1 (b), the border image intensities are copied to the extended area, and the width of the extended area should be equal to (block size/2 + p). This way, any block in an image can find its best matching position in the reference image. Thus each node can have its most suitable displacement.

4.2. Avoiding Triangle Inconsistency

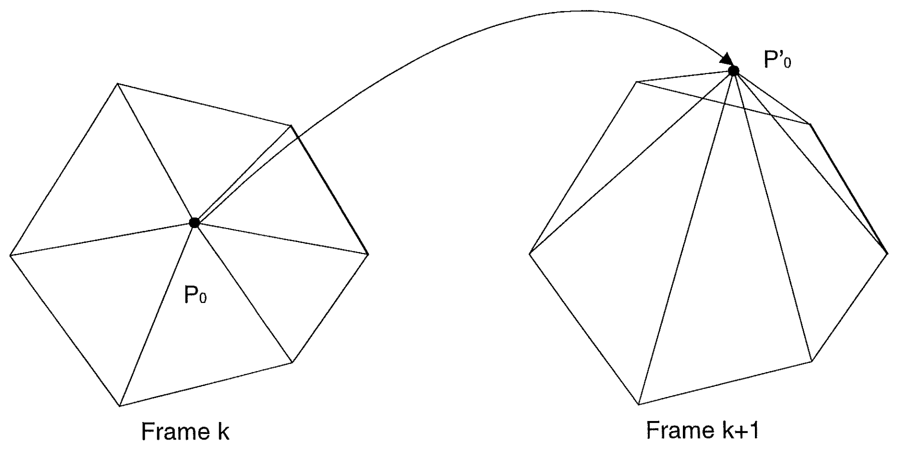


Figure 4.2 Illustration of inconsistent motion vectors.

After initialization of the motion vectors, we can reconstruct the image by compensating the obtained motion vectors. However, it is possible that motion vectors may be inconsistent in the sense that they do not preserve the connectivity of the mesh structure. This is illustrated in Figure 4.2, where the node P_0 in frame k is connected with its neighbour nodes which form the polygon. The motion vector at the node P_0 must move it inside the polygon in frame k+1. However, because the motion estimation method utilized to compute the node-point motion vector at the node P_0 does not employ such a constraint, this condition may be violated as shown in Figure 4.2. The displaced node P'_0 lies outside its displacement support region. When such a condition occurs, the estimated vector should be replaced by a motion vector that is interpolated from those of the surrounding nodes, or alternatively, a local search may be conducted to estimate the motion vectors at such nodes.

Therefore, we use a motion vector post-processing (after block matching) algorithm to preserve the connectivity of the triangle patches. In case of a motion vector crossover, it will be replaced by a motion vector that is interpolated from those of the surrounding N nodes, through the following procedure:

1. Process the nodes in the order of scan line to detect nodes with inconsistent motion vectors as follows: At each node P_0,
   (a) Find all the nodes connected to node P_0, and label them as P_i, $i = 1, \ldots, K$, where K is the number of nodes connected to P_0.
   (b) Find the motion compensated node locations P_i', $i = 1, \ldots, K$, in frame t+1 using the motion vectors at the nodes P_i. Form the polygon defined by P_i', $i = 1, \ldots, K$.
   (c) Motion compensate the node P_0 to find P_0'. If P_0' is inside the polygon, go to next node in order. Otherwise,
2. Interpolate the motion of the node from its neighbours as follows:

$$u = \frac{\sum_{i=1}^{K} \frac{u_i}{d_i}}{\sum_{i=1}^{K} \frac{1}{d_i}} \tag{4.1}$$

$$v = \frac{\sum_{i=1}^{K} \frac{v_i}{d_i}}{\sum_{i=1}^{K} \frac{1}{d_i}} \tag{4.2}$$

where (u_i, v_i) is the node-motion vector at the node P_i, and d_i is the distance between the node P_i and P_0.

This way, the connectivity can be kept by avoiding triangle inconsistency. Thus the procedure provides increased robustness to errors in motion estimation. In fact, throughout our experiment, this condition rarely occurs.

Until now, from these translation vectors, one can reconstruct a prediction. However, if the movement of the objects present in the image is not only translational, the movement will be described only coarsely and the quality of the reconstructed image will be poor. So we will use motion refine algorithm to get the accurate motion vectors of the nodal points.

4.3. Refinement by Deformation of the Triangulation

In the preceding paragraph, a grid of control nodes to which a triangulation of Delaunay is applied, was used to describe the reference image at time *t - dt*. Each node has a translation vector, found at the stage of initialization by block matching, to approximate the movement between the reference image and the image to be predicted. These vectors can model only translations and do not take into account the deformation of the objects. This is why they must be refined in order to possibly model the movement to maximize the image quality after motion compensation. The problem thus consists in estimating these displacements. The following proposes a method of searching optimal displacement of each node based on a local deformation of the triangulation.

4.3.1. Principle

In reference to Figure 4.3, which illustrates the mesh deformation according to a node displacement in the image plane, the main principle of the proposed mesh refinement can be described as the procedure which is composed of an iterative process as follows:

1) considering a given node, fix the location of the surrounding nodes;

2) move the considered node in a given neighbourhood inside the support region, and synthesize the predicted image in the polygon defined by the surrounding nodes according to this displacement;
3) calculate the prediction error inside the polygon: keep the displacement that minimizes this error,
4) update the displacement of the considered node with this refinement.
5) Considering the next node, go to step 1).

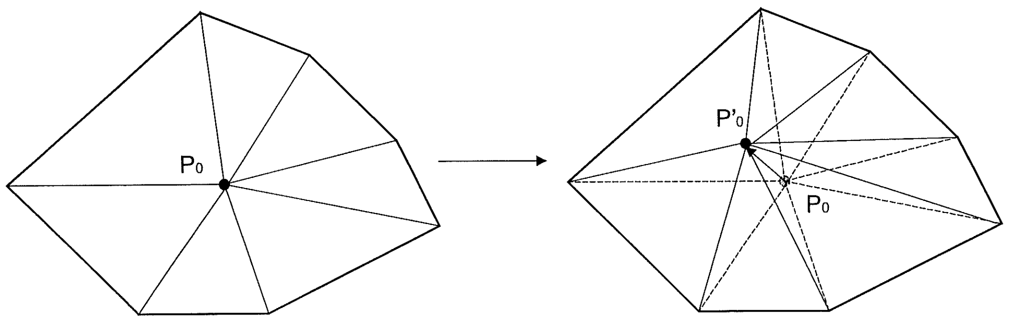


Figure 4.3 Refinement process: P_0 is the considered node, P_0' is its new location.

After each refinement, the difference between the current image and the predicted image either decreases or remains unchanged. Therefore, the refinement process can be iterated until all the control nodes converge to either local or global minima. The refinement process is applied to the considered node when its displacement was refined at the previous iteration or if the displacement of at least one of the surrounding nodes was refined at the previous iteration. Thus, in most cases the number of the mesh nodes to which the refinement process is applied decreases as the iteration goes on.

4.3.2. Support Region and Search Region for a Node

Before the location of a given node can be refined, the support region and search region associated with the given node are to be determined. The definition of support region and search region for a node, and the process for obtaining them are presented here for the inner, boundary and corner nodes.

Let P denote an arbitrary inner node in a triangular mesh, let K denote the number of patches that are connected to P, and let S_1, ..., S_K denote these patches. A search for the refined position of P is conducted within the region $S = \cup^{K}_{i=1} S_K$, which is referred as the "support region" of P, as shown in Figure 4.4 (a). The node P may be allowed to take on any location in the support region S. However, to simplify the computation, we define a square window that is centered around node P as the "search region" of node P (Figure 4.4 (a)). The constrained condition is that the search region must be enclosed in the support region. The maximum size of the search region, denoted as M_s, is predefined, and the maximum size of the square window inside the support region is denoted as M_w. Then the search range [-Sz, Sz] is defined as follows:

$$S_z = \begin{cases} M_s / 2 & \text{if } M_s \le M_w; \\ M_w / 2 & \text{if } M_s > M_w. \end{cases} \qquad (4.3)$$

Note that Sz is an integer.

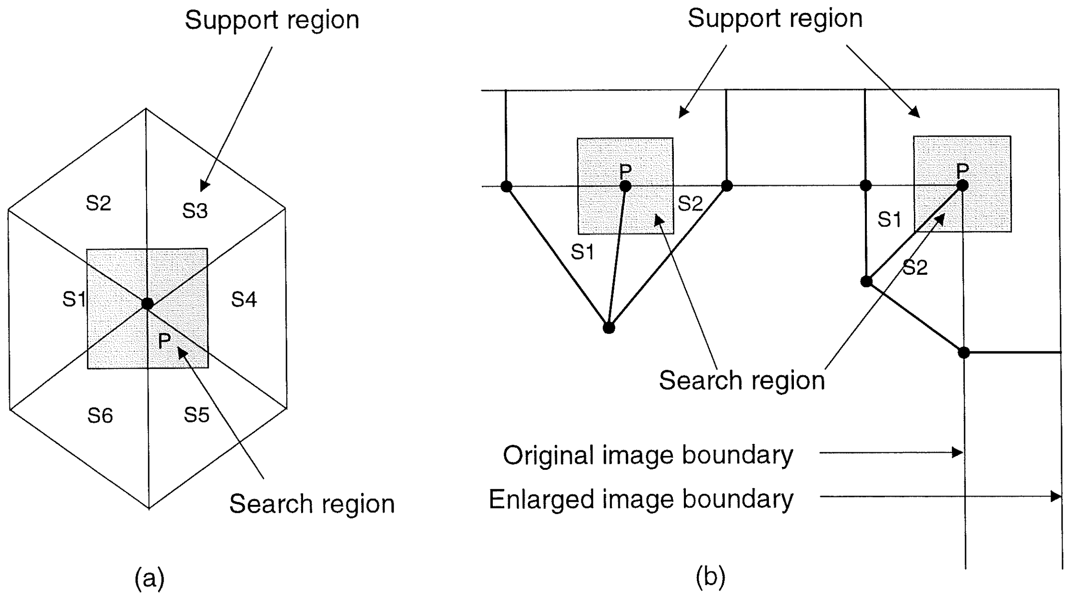
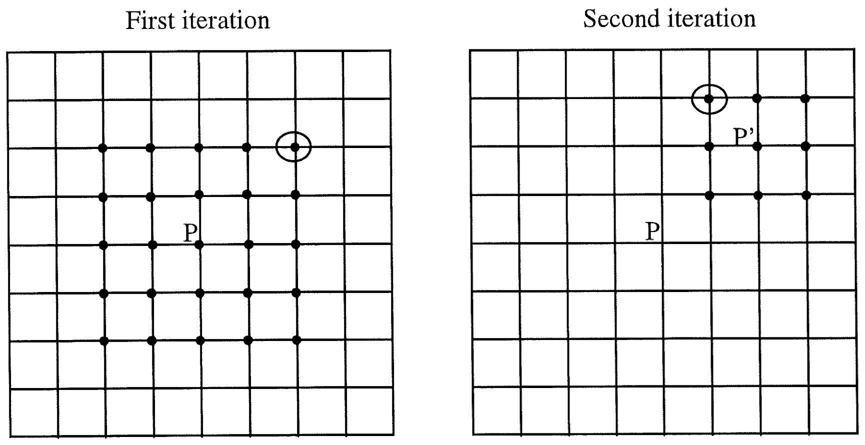


Figure 4.4 Support region and search space, (a) inner node; (b) corner and boundary nodes.

However, for a corner or boundary node, the situation is different. If we define the "support region" for these nodes as the union of associated triangular patches, which is the same as for inner nodes, then the considered nodes are located on the edge of this region (also the boundary of image). In some publications [14, 15], a search for a refined position of boundary nodes is constrained to only the corresponding image boundary segments, and for corner nodes, even zero motion is mandatorily applied to them. As said in the initialization part, the approach is not sufficient when an object moves out or inside the image boundary from frame to frame. So we will use the enlargement algorithm again to solve this problem. At first, we rebuild the image based on the obtained motion vectors. Because the corner and boundary nodes may move outside of the image border, the new image size *Sn* may be larger than the original image size *So*. We define the maximum search region size *Ms* as the same as for the inner nodes. Thus, the width of the extended region Re in Figure 4.4 (b) should be $(Sn - So + Ms)/2$. The support region for corner and boundary nodes can then be expressed as $S = \cup^{K}_{i=1} S_K \cup \mathrm{Re}$. The support region and search region for corner and boundary nodes are shown in Figure 4.4 (b).



● : Position to be tested;
○ : Optimal position;

P : Considered node;
P' : New node position.

Figure 4.5 Search range of a considered node.

As shown in Figure 4.4, the search space of a control node is defined as a square window. During motion refinement, the considered node may move to any location in the search space. So we use a full search algorithm to find the optimal displacement of the considered node within one iteration. However, a full search algorithm is computation expensive, and after one iteration, the nodes have moved towards their optimal position. So it is not necessary to keep the same search size during the subsequent iteration. Thus, we adopt a kind of hierarchical method, as shown in Figure 4.5, that is, after one iteration, the search size is divided by two. This way, each control node can rapidly converge to its local or global minima.

4.3.3. Refining

After determining a search space for a node, we can start to refine the location of this node. In order to find the optimum displacement of N_0, we employ the MSE criterion defined as:

$$\begin{aligned} E_S &= \frac{1}{N}\sum_{p\in S}(I(p+d(p),t-dt)-I(p,t))^2 \\ &= \frac{1}{N}\sum_{e\in S}\left[\sum_{p\in e}\left(I(p+\sum_{k=1}^{3}\varphi_k^e(p)d_{ver(e,k)},t-dt)-I(p,t)\right)^2\right] \qquad (4.4) \\ &= E_S(V) \end{aligned}$$

where N denotes the number of pixels within the support region of point $p=(x, y)^T$, S is support region of P. According to (3.9) which formulates the displacement of each point in a given element e as a function of the displacement of the considered element vertices, ver(e, k) corresponds to the kth vertex of the element e, and $V=(d_1, \ldots , d_n)^T$ to the complete nodal motion vector field.

The energy minimization is realized by estimating the optimal set of nodal motion vectors. The motion estimation approach described below considers successively the influences of each node displacement on the global error criterion. The nodal vector field V is determined iteratively, and the way to obtain $V^{(k+1)}$ from $V^{(k)}$ is as follows:

$d_i^{(k+1)}$ is estimated for i=1, then 2, …, until n such as:

$$E(d_1^{(k+1)}, \dots d_i^{(k+1)}, d_{i+1}^{(k)}, \dots, d_n^{(k)})$$
$$\leq E(d_1^{(k+1)}, \dots d_{i-1}^{(k+1)}, d_i^{(k)}+v_i, \dots, d_n^{(k)}) \ \forall v_i \in F \tag{4.5}$$

where F contains the set of acceptable vectors defined by the application context, i.e.

$$d_i^{(k+1)} = d_i^{(k)} + \arg\{\min E_{supp(i)}(v_i), \ (v_i \in S)\} \tag{4.6}$$

where

$$E_{\sup p(i)}(v_i) = \frac{1}{N} \sum_{e \in \sup p(i)} \left[\sum_{p \in e} \left(I(p + \varphi^e_{ind(e,i)}(p)(d^k + v_i, t - dt) - I(p,t) \right)^2 \right]$$

$ind(e, i) \in \{1, 2, 3\}$ indicates the index of the vertex of the triangle e whose index within the global triangulation is i, and $supp(i)$ is composed of the triangles which include the vertex i. This process is also a proof of principle of refinement mentioned-above.

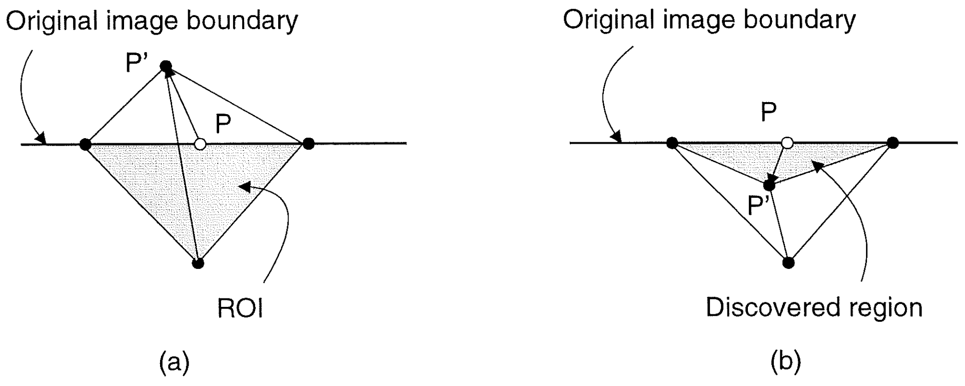


Figure 4.6 (a) Problem of ROI, (b) Problem of discovered region.

Note that when one calculates the error Es, only those spatial positions that map onto the ROI (region of interested) of frame at time t, are included in the calculation. This is illustrated in Figure 4.6. In our case, the ROI is inside the original image frame. The region mapped outside of image boundary is abandoned. On the other hand, if a boundary node of a past frame moves inside the current frame, it will cause a blank region in the ROI of the predicted image; we call it a discovered region. Figure 4.6. (b) illustrates the discovered region caused by inward motion of boundary node. When this is the case, we fill the discovered region with the corresponding pixel intensity of the reference image, or with the projection of the boundary pixel intensity of the reference image. It should be noticed that the discovered region is a part of ROI, that is to say, it is included for error calculation.

4.3.4. Application

Whole Procedure

As indicated and shown in the preceding paragraphs, refinement is an iterative procedure, and it will converge to steady state. Taking computation cost into consideration, a maximum search size is predefined and it will be divided by two after each iteration. To make the implementation efficient, we assign a marker to each node, to present the status of displacement of the treated node. For the start of the iteration, all markers are all initially set to zero. When the displacement of the treated node is updated, the markers of this node and all its adjacent nodes are set to one. Otherwise, if the motion vector remains unchanged, the marker is set to zero. The algorithm stops when all markers are zero or the search size is equal to one. The whole procedure of motion vector refinement is illustrated in Figure 4.7.

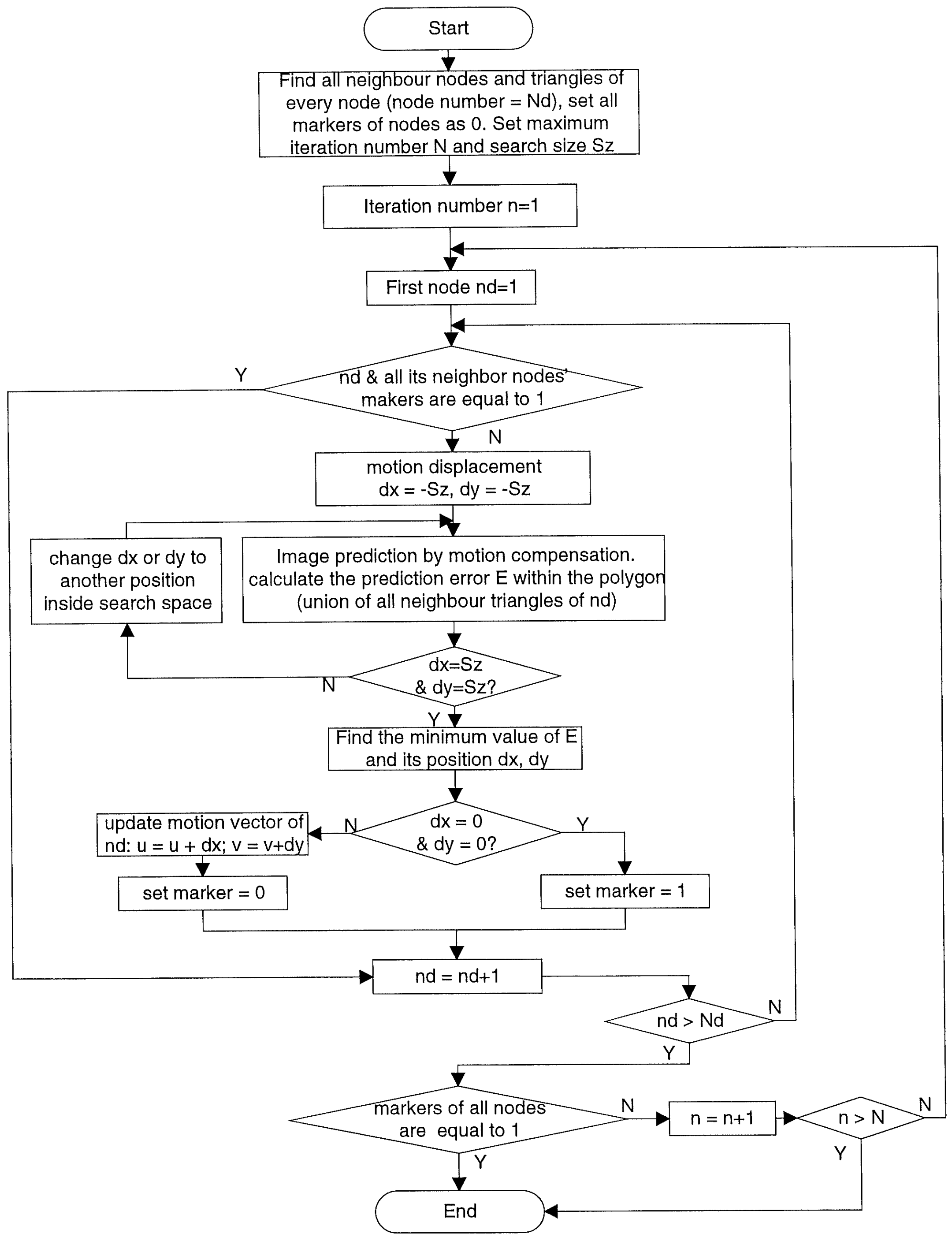

Start
Find all neighbour nodes and triangles of every node (node number = Nd), set all markers of nodes as 0. Set maximum iteration number N and search size Sz
Iteration number n=1
First node nd=1
nd & all its neighbor nodes' makers are equal to 1
Y
N
motion displacement dx = -Sz, dy = -Sz
change dx or dy to another position inside search space
Image prediction by motion compensation. calculate the prediction error E within the polygon (union of all neighbour triangles of nd)
dx=Sz & dy=Sz?
N
Y
Find the minimum value of E and its position dx, dy
dx = 0 & dy = 0?
N
Y
update motion vector of nd: u = u + dx; v = v+dy
set marker = 0
set marker = 1
nd = nd+1
nd > Nd
N
Y
markers of all nodes are equal to 1
N
Y
n = n+1
n > N
N
Y
End

Figure 4.7 Flow chat of motion vector refinement.

Motion Compensation

The prediction error in the polygon is calculated according to (4.4). From (4.4), we can see that the prediction error is the mean squared difference between the predicted (reconstructed) current image at time t and the reference image at time $t - dt$ within the treated polygon. The prediction of the current image is constructed by using motion compensation for the reference image:

$$\hat{I}(p, t) = I(p + d(p), t - dt) \qquad (4.7)$$

Where $d(p)$ is the motion vector of the point $p(x, y)$.

Based on the knowledge introduced in section 3.2, nodal motion vectors interpolation within motion compensation, described by (3.10), is used to compensate luminance values. Since the aim consists in determining a luminance value for each point of the sampling grid at time t, a backward motion compensation is considered, that is nodal location pi become $pi+di$ and nodal motion vectors di become $-di$. Then, since the scanning is done on the sampling grid at the time t, every pixel in the frame t has a corresponding vector and can be affected with a luminance value of time t-dt. In short, nodal point motion vectors establish a set of point correspondences from frame t-dt to frame t, which are used to determine a set of backward afine transformation from frame t to t-dt (see Figure 4.8).

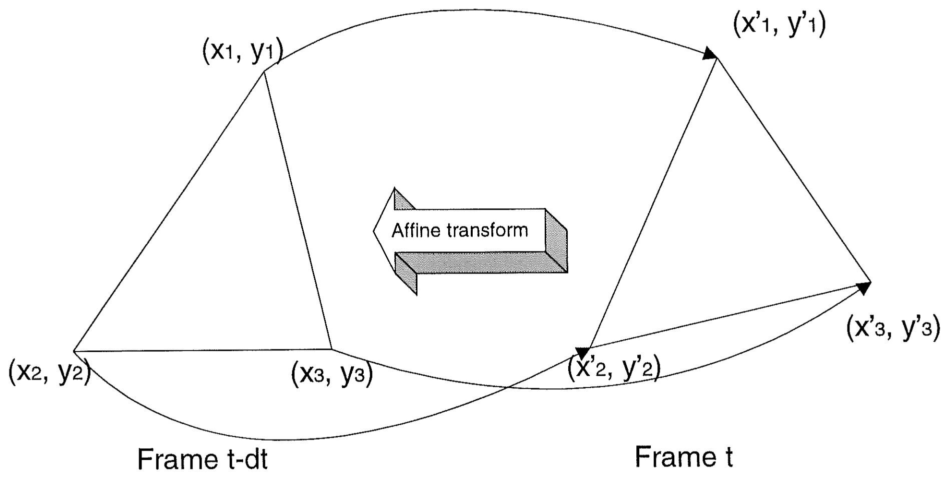


Figure 4.8 Backward Affine transformation.

Now knowing the motion vector of each point of a given triangle, the motion compensation can be carried out easily. Previously, each triangle should be marked by different values in order to build a correspondence between the image pixels and the motion model. And to increase speed, the processing of a considered triangle should be limited to a minimum rectangular region. The process proceeds in three stages:

1. Define a minimum rectangular region which can enclose this triangle. As shown in Figure 4.9, the coordinates of up-left and down-right corners of the rectangle, (x_{min}, y_{min}) and (x_{max}, y_{max}), are determined from the coordinates of three vertices of the considered triangle. Where $x_{min} = \min(x_1, x_2, x_3)$, $y_{min} = \min(y_1, y_2, y_3)$, $x_{max} = \max(x_1, x_2, x_3)$, and $y_{max} = (y_1, y_2, y_3)$.
2. For the considered triangle *T*, link its three vertices to get three edges. Fill the triangular area with a particular value using scan line algorithm. That is, first detect two edge points P1 and P2 by scanning order, then mark all pixels between P1 and P2 by a special value. (Figure 4.9)
3. Process this marked triangle inside the defined rectangle.

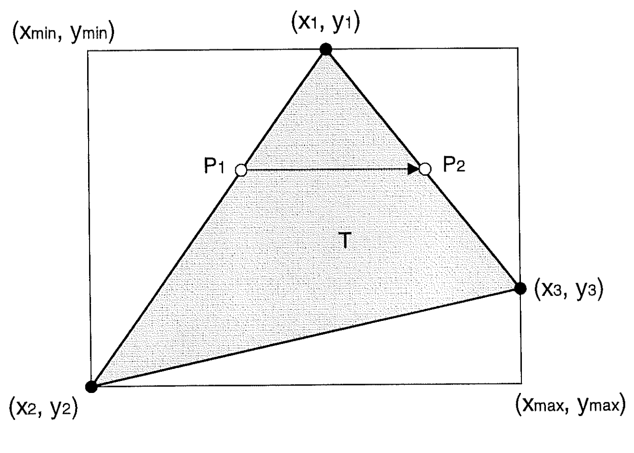


Figure 4.9 Detection of treated triangle.

Interpolation of Image Intensity

Notice that when the motion vector indicates a position which does not correspond to a point in the sampling grid, the reconstructed image intensity value is obtained from a bilinear interpolation of the four nearest pixels. With this, the intensity value at point (x', y') is obtained by:

$$I_n(x', y') = (1-\alpha)\,((1-\beta)I_{n-1}(x, y) + \beta\, I_{n-1}(x+1, y)) + \alpha((1-\beta)\, I_{n-1}(x, y+1) + \beta\, I_{n-1}(x+1, y+1)) \quad (4.5)$$

Where (x, y) is the integral part and (α, β) is the fractional part of the co-ordinate (x', y'). This method requires six multiplications for each pixel. Various fast algorithms to speed up the computation of (4.5) have been proposed.

4.5. Image Reconstruction

Image reconstruction is the final step of video coding and takes place at the encoder and decoder sides. The process is the same as the motion compensation mentioned above. During the motion estimation, motion compensation is implemented within the support region of the considered node. For image reconstruction, we have found the motion vectors of all the control nodes, so we compensate the previous image for all pixels triangle by triangle. The flow chat of image reconstruction is shown in Figure 4.10.

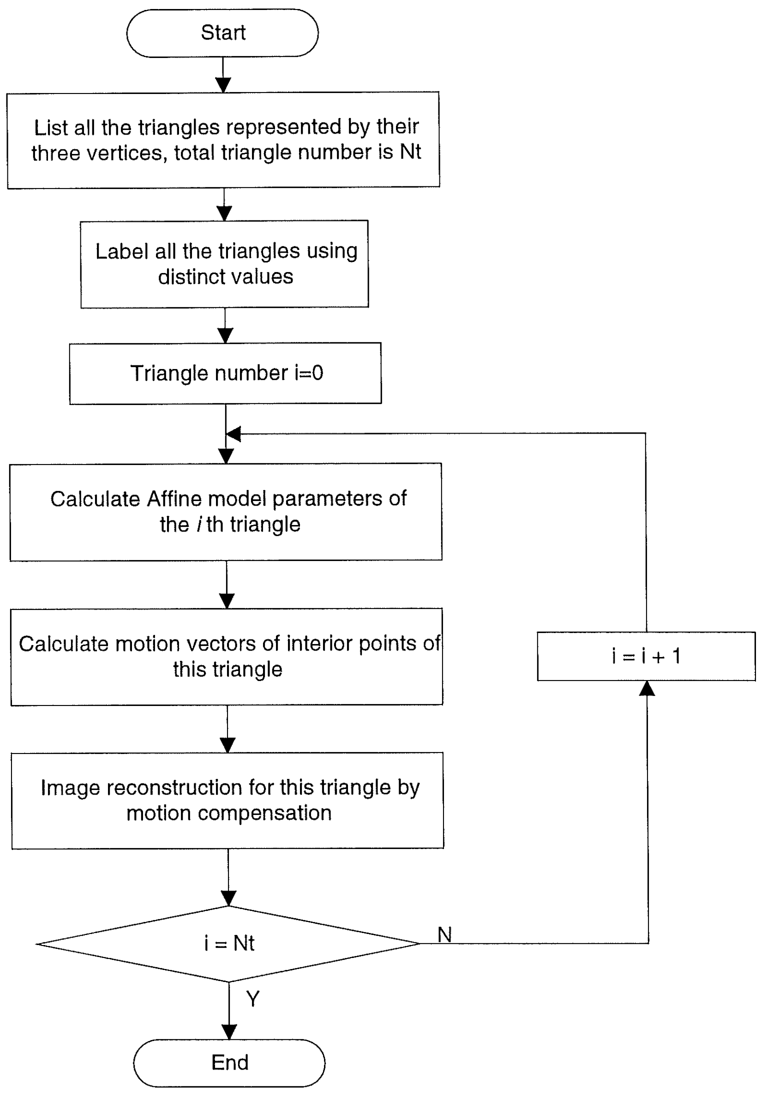

Start
List all the triangles represented by their three vertices, total triangle number is Nt
Label all the triangles using distinct values
Triangle number i=0
Calculate Affine model parameters of the *i* th triangle
Calculate motion vectors of interior points of this triangle
Image reconstruction for this triangle by motion compensation
i = Nt
N
Y
i = i + 1
End

Figure 4.10 Flow chat of image reconstruction.

We have described each step of a segmentation-based mesh design for motion estimation scheme so far. The next chapter will be the simulation results and conclusion.

Chapter 5. Simulation Results and Conclusion

In the previous chapters, a segmentation-based mesh design for motion estimation scheme has been proposed and all the phases have been described. The objective of this project is to get the reconstructed image by motion compensation and compare it with BMA. In this chapter, we present the simulation results and give a final conclusion.

5.1. Simulation Results

The simulations are performed using two test image sequences (Miss America and Foreman) in CIF format (352 × 288 pixels / frame). Only the luminance is processed. Backward prediction is simulated, i.e., each frame is predicted (reconstructed) from its previous frame. In order to evaluate purely the prediction precision (i.e., excluding the error propagation effect), the original image of the previous frame is used to predict the current frame. Image quality is evaluated by peak signal-to-noise ratio (PSNR), given by

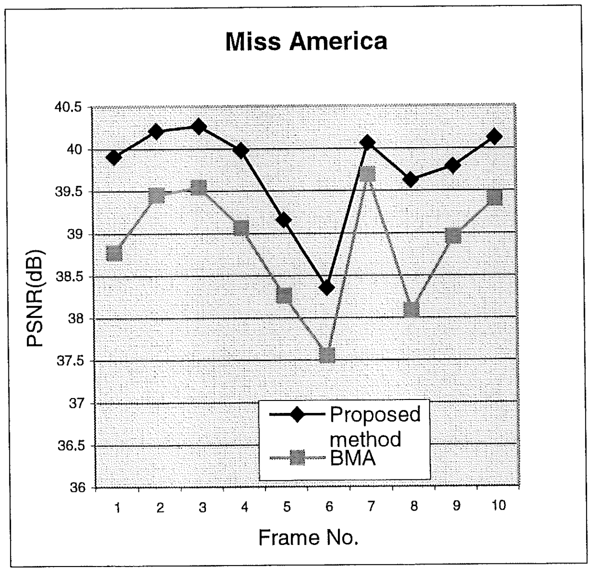
$$PSNR = 10\log_{10}\frac{255^2}{MSE} \qquad (5.1)$$

Where MSE represents the mean square error between the original and reconstructed frames, it is given by

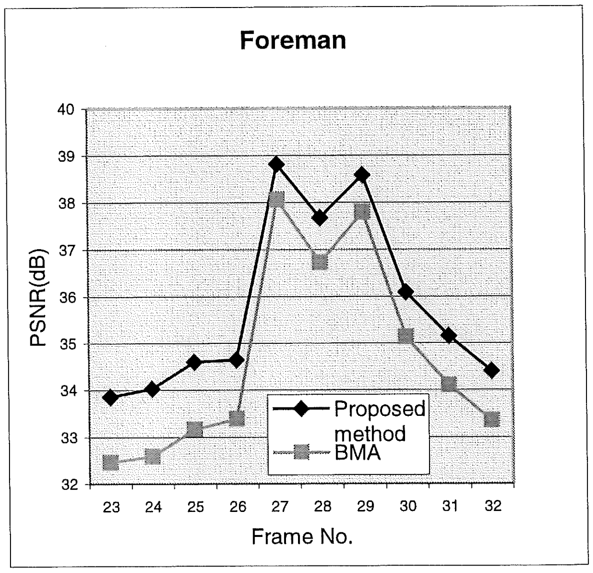
$$MSE = \frac{1}{I \times J}\sum_{i=0}^{I-1}\sum_{j=0}^{J-1}\left(O(i,j) - R(i,j)\right)^2 \qquad (5.2)$$

Where $I \times J$ represents the frame size, and $O(i, j)$ and $R(i, j)$ denote original and reconstructed images, respectively.

The performance of the segmentation-based mesh design for motion estimation and compensation is compared to that of conventional motion compensation using block matching. The full search algorithm with the mean absolute difference as the block matching error criterion is used. The block size of BMA is 16×16 pixels, and search range used by BMA is ± 7 pixels in both vertical and horizontal dimensions. With the proposed scheme, an initial estimate is first obtained by the BMA and then updated iteratively, as described in the previous chapter. For the BMA of initialization process, the block size is 15×15 pixels since the block is centered around a control node, and the search range is ± 7 pixels. For the refinement process, the search range is initialized as ± 7 pixels, and after each iteration, this range is divided by 2, so the iteration number is restricted to at most 3. Table 5.1 gives the PSNR results of BMA and proposed method. Figure 5.1 shows the PSNR curves.



(a)

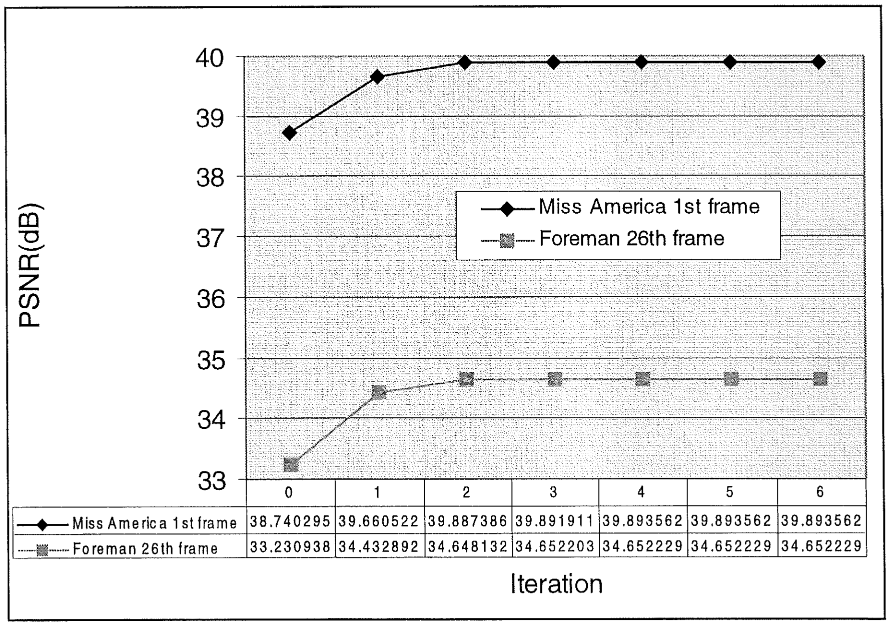


(b)

Figure 5.1 PSNR curves obtained by proposed method and BMA, (a) Miss America; (b) Forman.

Table 5.1 PSNR of BMA and proposed method.

Miss America Seq. No.	PSNR Of BMA	PSNR of Proposed method		Foreman Seq. No.	PSNR of BMA	PSNR of Proposed method
1	38.776302	39.886181		23	32.466370	33.856060
2	39.454369	40.210739		24	32.594524	34.028569
3	39.541801	40.270092		25	33.169453	34.601997
4	39.064316	39.981239		26	33.395699	34.651287
5	38.259865	39.157368		27	38.065998	38.821270
6	37.551235	38.357059		28	36.728539	37.674125
7	39.689461	40.066349		29	37.801094	38.585281
8	39.082512	39.623711		30	35.137424	36.091587
9	38.957756	39.787331		31	34.101692	35.159222
10	39.398514	40.124088		32	33.362087	34.403877



	0	1	2	3	4	5	6
Miss America 1st frame	38.740295	39.660522	39.887386	39.891911	39.893562	39.893562	39.893562
Foreman 26th frame	33.230938	34.432892	34.648132	34.652203	34.652229	34.652229	34.652229

Figure 5.2 Refinement algorithm convergence.

As mentioned in the last chapter, the iterative algorithm of motion refinement will always converge, at least to a local minimum. The rate of convergence of this algorithm is shown in Figure 5.2 for the 1st frame of Miss America and for the 26th frame of Forman; the search range is fixed to ±7 pixels. From Figure 5.2, we can see that the refinement normally converges in 4 iterations, the PSNR after the 3rd iteration is nearly same as after the 4th iteration, and the results of fix search range and varied search range (divided by 2) is almost the same. So we use the latter method because it can significantly reduce the computation cost and be implemented much faster.

Since the block size of BMA is 16×16, and the image size is 352×288, the block number of each frame is $18 \times 22= 396$. In contrast, by using the segmentation-base mesh design scheme, the average numbers of control nodes are 116 and 176 for Miss America and Foreman respectively (table 3.1). So, the node numbers are only about 29% and 44% of the block numbers of BMA, the number of transmitted motion vectors is thus greatly reduced.

From the first two chapters, we know that BMA can bring motion block effects, especially when motion is important and frame rate is low. Figure 5.3 shows the original images and reconstructed images by BMA and proposed method. The left and right columns are Miss America and Foreman respectively. The first row represents the original images (the 4th frame of Miss America and the 26th frame of Foreman). The second row represents the reconstructed images by BMA (Miss America is reconstructed from the 1st frame, Foreman is reconstructed from 23rd frame). The third row represents the reconstructed images by the proposed method. The images are zoomed out in order to have a better visual effect. From the Figure 5.3, it is clear that the proposed method can outperform BMA for the motion block artefacts and highly improves the subjective image quality.

From [29, 30], the PSNR of the reconstructed 4th frame of Miss America from the 1st frame is around 37.44, but using our algorithm, the PSNR is 39.06, so the proposed segmentation-based mesh design scheme for motion estimation is better than STG-based mesh design scheme.

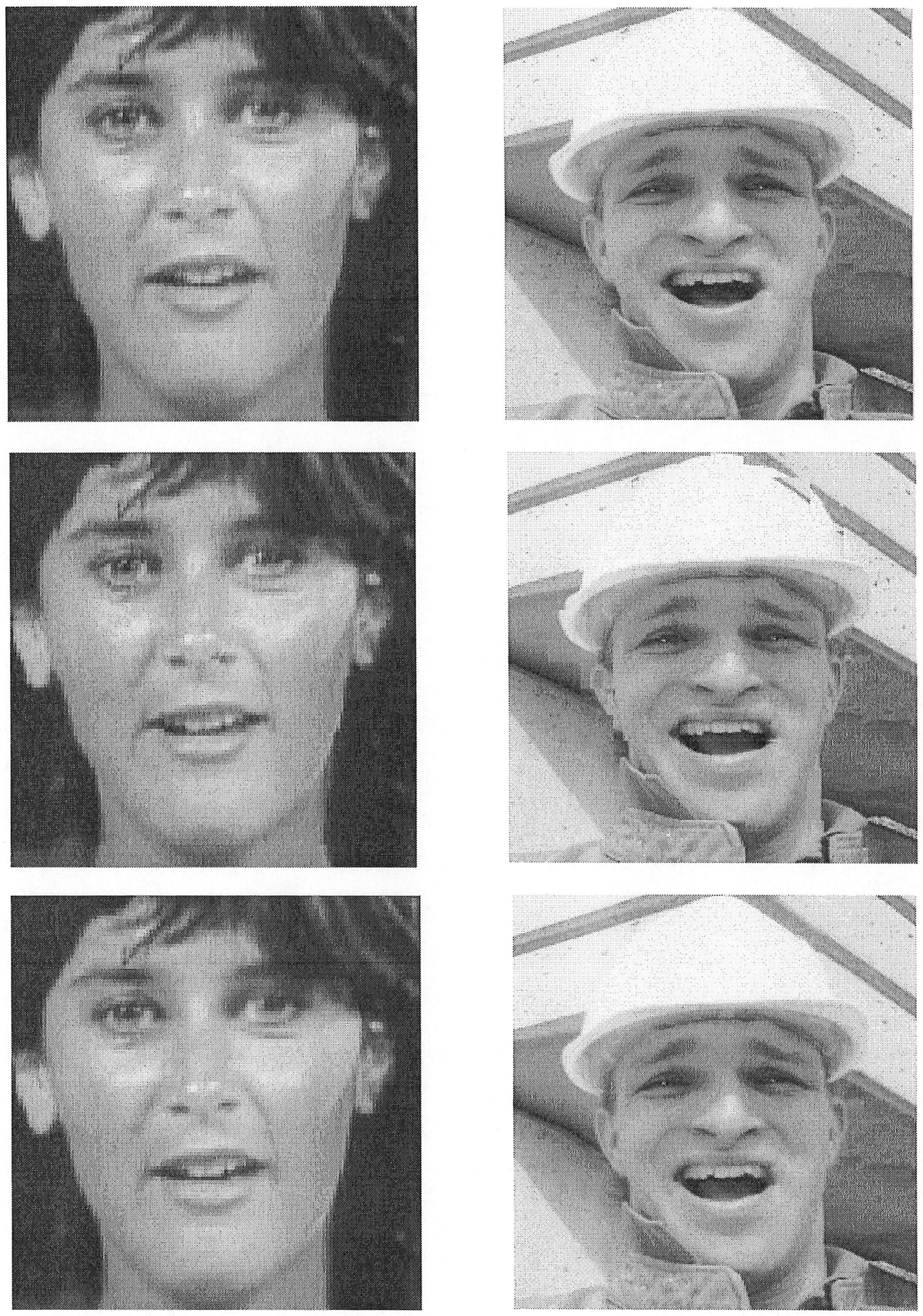


Figure 5.3 Original and reconstructed images, left: Miss America; right: Forman.

5.2. Conclusion

In this thesis, we have proposed a segmentation-based mesh design scheme for motion estimation between image frames. The current frame is reconstructed by motion compensating previous image based on a mapping of a selected set of image points (control nodes) between the previous and the current frames. Triangular mesh structure and affine motion model are chosen for representation of the motion of frames. The content-based triangular mesh generation is based on image segmentation followed by selection of control nodes. An important novel property of the proposed method is that it selects the control nodes on the image contours, after constrained Delaunay triangulation and triangle fusion algorithm, each triangular patch is nearly homogenous, so its motion vectors can be accurately estimated.

Simulation results indicate that this method performs better than classical motion estimation algorithm like BMA. Objectively, the PSNR is higher. Subjectively, this method can improve the motion block effects produced by BMA.

Since the generated mesh structure coincides with image contours, particular objects can be easily identified, so this method is suitable for object-oriented process. This will be the subject of future work.

References:

[1] P. Salembier, L. Torres, F. Meyer, and C. Gu, "Region-based video coding using mathematical morphology," *Proceedings of the IEEE,* Vol. 83, No. 6, pp. 843-857, 1995.

[2] G. H. Lee, J.-S. Kim, and R.-H. Park,, "Video coding using variable block-size segmentation by motion vectors," *Journal of Visual Communication and Image Representation,* Vol. 5, No. 4, pp. 342-355, 1994.

[3] D. Wang, C. Labit, and J. Ronsin, "Segmentation-based motion compensated video coding using morphological filters," *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 7, No. 3, pp.549-555, June 1997.

[4] E. Salari, and S.Lin, "Low-bit-rate segmentation-based image sequence coding," *Opt. Eng.* Vol. 34, No. 3, pp.829-833, 1995.

[5] G. J. Sullivan, and R. L. Baker, "Motion compensation for video compression using control grid interpolation," in *Proceedings of IEEE ICASSP,* Toronto, Canada, May 1991, pp. 2713-2716.

[6] Y. Wang, and O. Lee, "Active mesh - a feature seeking and tracking image sequence representation scheme," *IEEE Trans. Image Processing,* Vol. 3, No. 5, pp. 610-624, 1994.

[7] O. Lee, and Y. Wang, "Motion-compensated prediction using nodal-based deformable block matching," *J. Visual Comm. & Image Repres.* Vol. 6, No. 1, pp. 26-34, 1995.

[8] D. Wang, "A multiscale gradient algorithm for image segmentation using watersheds," *Pattern Recognition*, Vol. 30, No. 12, pp.2043-2052, 1997.

[9] H. Li, A. Lundmark and R. Forchheimer, "Image sequence coding at very low bitrates: a review," *IEEE Trans. Image Processing,* Vol. 3, No. 5, pp. 589-609, 1994.

[10] V. Bhaskaran, and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures,* Kluwer Academic Publishers, Boston, 1995.

[11] M. Dudon, O. Avaro, and C. Roux, "Triangular active mesh for motion estimation", *Signal Processing: Image Communication*, 10 (1997) pp. 21-41.

[12] M. Dudon, G. Eude, and C. Roux, "Optimal sharing between Motion and Prediction error information", *Picture Coding Symp.*, pp. 617-622, 1996.

[13] Yuichiro Nakaya, and Hiroshi Harashima, "Motion compensation based on spatial transformations", *IEEE Trans. Circuits and System for Video Tech.,* Vol. 4, No. 3, pp. 339-356, 1994.

[14] J. Nieweglowski, T. Geroge Campbell, and Petri Haavisto, "A novel video coding scheme based on temporal prediction using digital image warping", *IEEE Trans. Consumer Electronics,* Vol. 39, No. 3, pp. 141-150, 1993.

[15] J. Nieweglowski, and Petri Haavisto, "Temporal image sequence prediction using motion field interpolation", *Signal Processing: Image Communication*, 7 (1995) pp. 333-353.

[16] Yao Wang, Xia-Ming Hsieh, Jian-Hong Hu, and Ouseb Lee, "Region segmentation based on active mesh representation of motion: comparison of parallel sequential approaches", *Proceedings of the ICIP*, pp. 185-188, 1995.

[17] A. Murat Tekalp, Yucel Altunbasak, and Gozde Bozdagi, "Two- versus Three-Dimensional Object-based video compression", *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 7, No. 2, pp.391-397, June 1997.

[18] Y. Altunbasak, and A. Murat Tekalp, "Closed-form connectivity-preserving solutions for motion compensation using 2-D meshes", *IEEE Trans. Image Proc.*, vol. 6, no. 9, pp. 1255-1269, Sep. 1997.

[19] Y. Altunbasak, and A. Murat Tekalp, "Occlusion-adaptive, content-based mesh design and forward tracking", *IEEE Trans. Image Proc.*, vol. 6, no. 9, pp. 1270-1280, Sep. 1997.

[20] C. Toklu, A. T. Erdem, M. I. Sezan, and A. Murat Tekalp, "Tracking motion and intensity-variations using hierarchical 2-D mesh modeling for synthetic object transfiguration", *Graphical Models and Image Processing*, vol. 58, no. 6, pp. 553-573, Nov. 1996.

[21] M. Eckert, J. Villar, J. I. Ronda, F. Suarez, F. Jaureguizar, and N. Garcia, "Object oriented motion compensation for very low bit-rate coding applying content based triangle meshes", *Proceedings of the VCIP*, pp. 350-360, 1997.

[22] G. Wolberg, *Digital image warping*, Los Alamitos, CA, U.S.A.: IEEE Computer Society Press, 1990.

[23] Jonathan Richard Shewchuk, Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, *First Workshop on Applied Computational Geometry (Philadelphia, Pennsylvania)*, pp. 124-133, ACM, May 1996.

[24] Reinhard Klotte, and Piero Zamperoni, *Handbook of Image Processing Operators*, John Wiley & Sons Ltd., 1996.

[25] Borko Furht, Stephen W. Smoliarr, and Hongjiang Zhang, *Video and Image Processing in Multimedia Systems,* Kluwer Academic Publishers, 1995.

[26] Raymond Westwater, and Borko Furht, *Real-Time Video Compression – Techniques and Algorithms*, Kluwer Academic Publishers, 1997.

[27] Borko Furht, Joshua Greenberg, and Raymond Westwater, *Motion Estimation Algorithms for Video Compression*, Kluwer Academic Publishers, 1997.

[28] William I. Grosky, Ramesh Jain, and Rajiv Mehrotra, *The Handbook of Multimedia Information Management*, Prentice Hall PTR, 1997.

[29] Yucel Altunbasak, A. Murat Tekalp, and Gozde Bozdagi, "Two-dimensional Object-based Coding Using a Content-based Mesh and Affine Motion Parameterization", *Proceedings of IEEE*, pp. 394-397, 1995.

[30] Yucel Altunbasak, and A. Murat Tekalp, "Object-scalable Content-based 2-D Mesh Design for Object-based Video Coding", *Multimedia Communications and Video Coding*, Edited by Y. Wang et al., Plenum Press, New York, pp. 247-256, 1996.

[31] A. Murat Tekalp, Peter Van Beek, Candemir Toklu, and Bilge Günsel, "Two-dimensional Mesh-based Visual-object Representation for Interactive Synthetic/Natural Digital Video", *Proceedings of the IEEE*, pp. 1029-1051, Vol. 86, NO. 6, June 1998.

[32] K. Haris, S. N. Efstratiadis, N. Maglaveras, and C. Pappas, "Hybrid Image Segmentation Using Watersheds", *SPIE Proc. Visual Communications and Image Processing'96*, 2727, pp. 1140-1151, Orlando, Florida, U.S.A. (1996).

[33] L. Vincent, and P. Soille, "Watershed in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations", *IEEE Trans. Pattern Analysis Mach. Intell*, 13, pp. 583-598, 1991.

[34] A. K. Jain, *Fundamental of Digital Image Processing*, Printice-Hall, New Jersey, 1989.

[35] Borko Furht, "Multimedia systems and techniques", Kluwer Academic Publishers, 1996.