2-D MESH-BASED MOTION ESTIMATION AND VIDEO OBJECT MANIPULATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF
NATURAL AND APPLIED SCIENCES OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HÜSEYİN KAVAL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2007

Approval of the thesis:

2-D MESH-BASED MOTION ESTIMATION AND VIDEO OBJECT MANIPULATION

submitted by **HÜSEYİN KAVAL** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Assoc. Prof. Dr. Gözde Bozdağı Akar _____
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Mete Severcan _____
Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. Gözde Bozdağı Akar _____
Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. Aydın Alatan _____
Electrical and Electronics Engineering Dept., METU

Asst. Prof. Dr. Çağatay Candan _____
Electrical and Electronics Engineering Dept., METU

Dr. Şener Yılmaz _____
Image Processing Dept., ASELSAN

Date: (07.09.2007)

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Hüseyin KAVAL

Signature :

ABSTRACT

2-D MESH-BASED MOTION ESTIMATION AND
VIDEO OBJECT MANIPULATION

Kaval, Hüseyin

M.Sc., Department of Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Gözde Bozdağı AKAR

September 2007, 60 pages

Motion estimation and compensation plays an important role in video processing applications. Two-dimensional block-based and mesh-based models are widely used in this area. A 2-D mesh-based model provides a better representation of complex real world motion than a block-based model.

Mesh-based motion estimation algorithms are employed in both frame-based and object-based video compression and coding. A hierarchical mesh-based algorithm is applied to improve the motion field generated by a single-layer algorithm. 2-D mesh-based models also enable the manipulation of video objects which is included in the MPEG-4 standard. A video object in a video clip can be replaced by another object by the use of a dynamic mesh structure.

In this thesis, a comparative analysis of 2-D block-based and mesh-based motion estimation algorithms in both frame-based and object-based video representations is performed. The experimental results indicate that a mesh-based algorithm produces better motion compensation results than a block-based algorithm. Moreover, a two-layer mesh-based algorithm shows improvement over

a one-layer mesh-based algorithm. The application of mesh-based motion estimation and compensation to video object replacement and animation is also performed.

Keywords: Mesh-Based Motion Estimation, Video Object Manipulation, MPEG-4, 2-D Animation, Hexagonal Matching Algorithm.

ÖZ

2 BOYUTLU ÖRGÜYE DAYALI DEVİNİM KESTİRİMİ VE
VİDEO NESNE İŞLEME

Kaval, Hüseyin

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Gözde Bozdağı AKAR

Eylül 2007, 60 sayfa

Devinim kestirimi ve dengelemesi video işleme uygulamalarında önemli bir rol oynamaktadır. İki boyutlu bloğa dayalı ve örgüye dayalı modeller bu alanda yaygın olarak kullanılmaktadır. İki boyutlu örgüye dayalı bir model karmaşık gerçek dünya devinimleri için bloğa dayalı bir modelden daha iyi bir gösterim sağlamaktadır.

Örgüye dayalı devinim kestirimi algoritmaları çerçeveye ve nesneye dayalı video sıkıştırma ve kodlamasında kullanılmaktadır. Sıradüzensel örgüye dayalı bir algoritma, tek katmanlı bir algoritma tarafından üretilen devinim alanını iyileştirmek için uygulanmaktadır. İki boyutlu örgüye dayalı modeller MPEG-4 standardında yer alan video nesne işlemeye de olanak tanımaktadır. Dinamik bir örgü yapısı kullanılarak bir video nesnesi ile başka bir nesne yer değiştirilebilmektedir.

Bu tez çalışmasında, iki boyutlu bloğa ve örgüye dayalı devinim kestirimi algoritmalarının çerçeveye ve nesneye dayalı video gösterimlerinde karşılaştırmalı bir incelemesi gerçekleştirilmiştir. Deneysel sonuçlar örgüye dayalı

bir algoritmanın bloğa dayalı bir algoritmaya göre daha iyi devinim dengelemesi sonuçları ürettiğini göstermektedir. Ayrıca, iki katmanlı örgüye dayalı bir algoritma tek katmanlı örgüye dayalı bir algoritmaya göre iyileştirme sağlamaktadır. Örgüye dayalı devinim kestirimi ve dengelemesinin video nesne yer değişimi ve animasyona uygulaması da gerçekleştirilmiştir.

Anahtar Kelimeler: Örgüye Dayalı Devinim Kestirimi, Video Nesne İşleme, MPEG-4, İki Boyutlu Animasyon, Altıgene Dayalı Eşleştirme Algoritması.

To my mother, my father, and Burak…

ACKNOWLEDGMENTS

I would like to thank my supervisor Assoc. Prof. Dr. Gözde Bozdağı Akar for her guidance and support throughout this thesis study. This work could not be finished without her efforts and encouragements.

I would also like to thank my family and colleagues at Aselsan for their morale support, help and understanding.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1. Problem Definition

Motion estimation and compensation plays an important role in video processing applications. Motion estimation is the process of finding a set of motion vectors to represent the movement of video objects in an image sequence. Motion compensation uses this motion information to predict a frame from another frame. It is often employed in video compression. Since successive video frames contain similar objects, motion estimation and compensation yields good compression results by removing temporal redundancy.

Two-dimensional block-based and mesh-based models have traditionally been used in motion estimation and compensation. A 2-D mesh-based model is powerful for representing the complex real world motion. The main problems in a 2-D mesh-based model are designing the mesh and estimating the motion vectors of the nodes. Motion compensation using the estimated motion vectors is also required in video compression applications [3, 8].

The use of 2-D mesh-based models for object-based video representation is recommended in the MPEG-4 standard. MPEG-4 allows compression and manipulation of video objects in a scene separately. Thus, extraction of semantically meaningful video objects becomes a fundamental problem in object-based video representation. Shape and motion modeling of video objects are the other problems that need to be addressed [3, 4, 19].

2-D mesh-based models also enable the manipulation of video objects. Replacement of a video object in a video clip by another object, and merging computer generated images with real images to create enhanced display information are some examples of video object manipulation. In these

applications, the design and accurate tracking of a 2-D video object mesh are required [3].

1.2. Scope of the Thesis

The study described in this thesis is three-fold. First, several 2-D block-based and mesh-based motion estimation methods are compared in frame-based video representation. Then, the use of these methods in object-based video representation is investigated. Finally, application of 2-D mesh-based motion estimation and compensation to video object manipulation is performed.

1.3. Outline of the Thesis

This thesis consists of five chapters; and references are presented at the end of the thesis.

In Chapter 1, problem definition and scope of the thesis are presented.

In Chapter 2, two-dimensional motion estimation and compensation methods are described.

In Chapter 3, adaptation of 2-D mesh-based models to visual object representation is investigated.

In Chapter 4, results of the experiments for the presented algorithms and the applications are presented.

In Chapter 5, conclusions on this study and future work are discussed.

CHAPTER 2

2-D MOTION ESTIMATION AND COMPENSATION

2.1. Introduction

2-D motion estimation is often viewed as an initial step for 3-D motion analysis. Besides, it has other applications such as video filtering, video compression, etc. Depending on the type of application motion estimation techniques may differ. For instance, in a video compression application, the estimated motion vectors are used to predict a frame to be coded from a previously coded reference frame. In this case, the goal is to minimize the number of bits used in video coding while providing a good estimation of the real motion. Hence, there is a trade-off and a motion estimation method must be chosen accordingly [6].

2-D dense motion estimation methods can be classified as pel-recursive, optical flow equation-based, Bayesian and block-based methods [5]. Pel-recursive methods estimate the motion pixel-by-pixel. They are simple but they have low motion estimation accuracy. For this reason, they are rarely used in today's codecs [6]. Optical flow equation-based methods produce smooth motion fields but they have a bad performance in terms of the PSNR [8]. Bayesian methods employ probabilistic smoothness constraints to predict the motion vectors. Although they produce good motion estimation results, their main drawback is the high amount of computations required [5, 8].

Block-based motion estimation has been a popular motion estimation method due to its simplicity and satisfactory results in many video sequences. One problem with this approach is that the generated motion field is often not smooth because of the independent motion estimation of adjacent blocks. Hence, this results in blocking artifacts in low-bit-rate video compression [3, 6].

Mesh-based motion estimation overcomes the problem of blocking artifacts. It produces a continuous motion field by constraining the movement of adjacent patches. Mesh-based model also allows representation of complex motion such as zooming and rotation [2, 3, 6, 24].

2-D motion estimation methods can also be classified as backward or forward estimation. In the block-based motion estimation case, backward estimation refers to searching a block of pixels of the current frame in a previous reference frame to find the best matching block. In the mesh-based approach, backward estimation refers to searching the node points of the current frame in a previous reference frame to find the best matching positions. On the other hand, in forward estimation, node points are defined in the reference frame and searched for best matching positions in the current frame. Forward estimation is suitable for tracking image features through the video sequence but at the expense of complexity. Backward estimation is widely used due to the lower computational requirements of the mapping process [1-3].

In the following sections, first, block-based motion estimation is described. Then the mesh-based motion estimation approach is discussed. Finally, the hierarchical motion estimation concepts and a technique for performing hierarchical mesh-based motion estimation are presented.

2.2. Block-Based Motion Estimation

Block-based motion estimation is a popular motion estimation approach. It is widely used in digital video applications. Block-based motion estimation and compensation is also adopted in international video compression standards such as MPEG 1 and 2 [5].

In the block motion model, an image is partitioned into small blocks which are usually chosen as rectangles. In general, two types of motion are considered for these blocks; simple 2-D translation or various 2-D deformations. The advantage of the translational motion model is the low overhead to represent the motion field since one motion vector is sufficient for each block. However, it fails to represent complex motion such as zooming and rotation. For this reason, deformable motion models can be employed for better adaptation to those kinds of motion at the expense of higher overhead for motion representation [5, 6].

The block matching algorithm (BMA) is considered as the most popular block-based motion estimation method which adopts the translational motion model [14]. It is fast and simple requiring less complex hardware. For this reason, it is utilized in MPEG 1 and 2 codecs. The generalized block matching algorithm is a sophisticated alternative to classical block matching. It employs deformable blocks which provide a better representation of the underlying motion. However, this is achieved at the expense of an increase in the complexity of the motion estimation process. The hierarchical block matching algorithm also provides an improved motion representation by employing a layered motion estimation technique as described in Section 2.4 [5, 6].

In our studies, we have used block-based motion estimation to produce a fast initial estimation of the motion vectors which are then refined by a mesh-based motion estimation technique. As a result, the block matching algorithm has been employed due to its low computational requirements. The following section describes this algorithm.

2.2.1. Block Matching Algorithm

In the block matching algorithm, first, the current frame is segmented into fixed sized rectangular blocks. Then, each block is searched independently of each other in a reference frame to find a matching block. Finally, the displacement vectors for all blocks are obtained. The procedure is illustrated in Figure 2.1.

As shown in the figure, the search is performed only in a limited area called the search window. This limitation reduces the computational complexity. In the case of smooth transitions between the frames, a small search window can be applied to minimize the time required for computations.

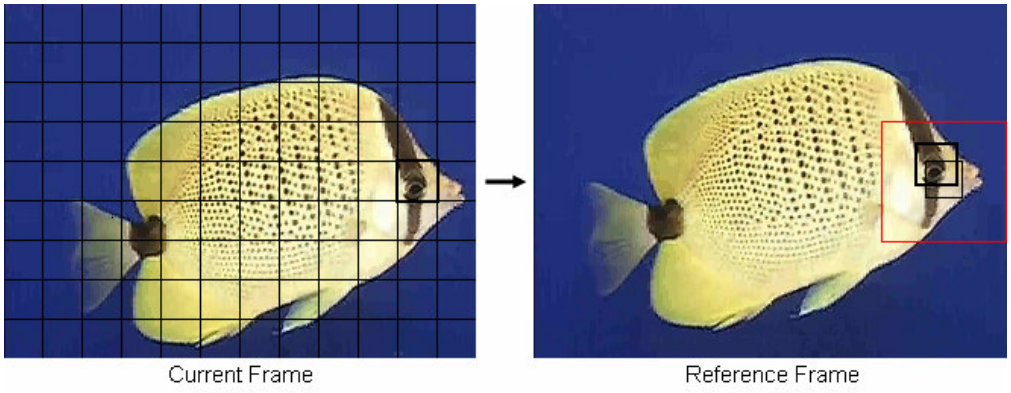


Figure 2.1 – Block Matching Algorithm

There exist several matching criteria for matching of blocks. Jain and Jain [14] used the mean square error (MSE). However, the MSE implementation is not simple due to the square operation included. Moreover, according to Srinivasan and Rao [16], the matching criterion has no significant effect on the search. Therefore, the mean absolute difference (MAD) is recommended due to its implementation simplicity [15]. The MSE and MAD are defined as

$$MSE(i,j) = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left[s_k(m,n) - s_{k-1}(m+i,n+j)\right]^2 \tag{2.1}$$

$$MAD(i,j) = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left|s_k(m,n) - s_{k-1}(m+i,n+j)\right| \tag{2.2}$$

where $s_k(m, n)$ is the luminance value of the pixel (m, n) in frame k, $(m+i, n+j)$ denotes a candidate block position in the search window, and $M \times N$ is the block size.

All candidate blocks in the search window are compared in terms of the MAD values and the one with the minimum MAD is selected as the best matching block. This matching procedure is applied to all of the blocks in the current frame. Finally, a motion field composed of each block's displacement vector is generated as shown in Figure 2.2.

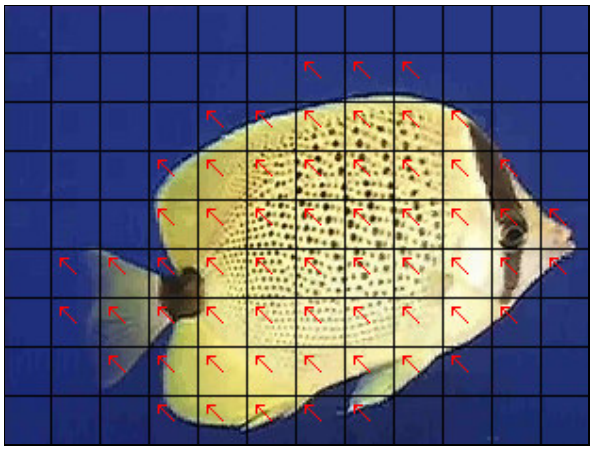


Figure 2.2 – Block Motion Field

In a video compression application, this motion field is used to synthesize a predicted image by motion compensation. The prediction is performed block-by-block, i.e. the intensities of pixels in a block are predicted from the corresponding pixels in the matching block.

The block matching algorithm is simple to implement and yields good results for many video sequences which contain translational motion. However, it suffers from representing rotational motion, zooming, etc. Besides, block-by-block prediction generates visual artifacts. Smaller blocks may be used for better motion modeling and to reduce the visual artifacts but this results in high bit rates in coding. A better alternative; mesh-based motion estimation is discussed in the next section.

In our implementations, we have used the block matching algorithm to generate a fast initial estimation of the motion field which is then refined by the mesh-based algorithms. In addition, we have applied motion compensation using the motion vectors produced by block matching to compare the algorithm with the mesh-based algorithms in terms of the PSNR, and entropy. These comparisons are presented in the Experimental Results Chapter in detail.

2.3. 2-D Mesh-Based Motion Estimation

Mesh-based motion estimation and compensation is an advanced video representation and processing technique. In a mesh-based representation, the underlying image is partitioned into non-overlapping polygonal patches. The motion field over the entire image is described by the motion vectors at the corners of the patches which are called the nodes of the mesh. The motion vectors of the interior points of the patches are interpolated from those node points. Mesh-based representation provides a continuous motion field over the entire image [6, 24]. Mesh-based motion estimation is illustrated in the figure below.

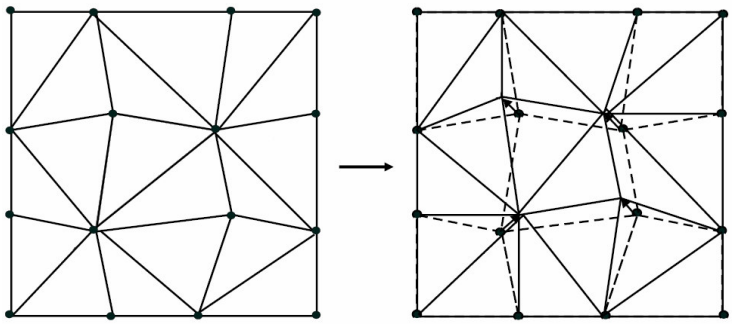


Figure 2.3 – Mesh-Based Motion Estimation.

In a 2-D mesh-based model, when a node location is updated through the motion estimation process, all of the patches which are connected to the node are deformed. The deformed patches eventually provides better adaptation to underlying motion than a block-based model which does not preserve the neighboring relations of the blocks [4, 7]. On the other hand, a 2-D mesh-based model may produce bad results in occlusion regions since the motion field is enforced to be continuous. An occlusion-adaptive mesh-based model was proposed by Altunbaşak and Tekalp [7] to overcome this problem. In the scope of this thesis, the occlusion problem is not handled.

In mesh-based motion estimation the primary problems that need to be solved are 2-D mesh design and motion vector estimation at the node points. In a video

compression application, motion compensation using the estimated motion vectors is also required. This chapter addresses these problems. First, 2-D mesh design methods are presented. It is followed by the description of an effective motion estimation method: the hexagonal matching algorithm. Finally, the motion compensation approach is given.

2.3.1. 2-D Mesh Design

Two-dimensional mesh design is usually performed by the use of either a regular or an irregular mesh structure. A regular mesh is composed of equal sized patches. In Figure 2.4 two examples of regular meshes are shown.

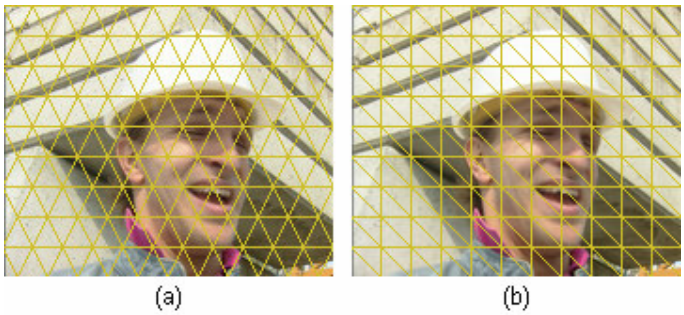


Figure 2.4 – Regular Mesh Examples. (a) An Equilateral Mesh, (b) A Right-Angled Mesh.

Nakaya and Harashima [1] and Park *et al.* [2] are some of the researchers who utilized regular meshes in their work. The main benefit of employing a regular mesh is that the number of bits used in coding is reduced since each node position does not need to be coded explicitly. On the other hand, a regular mesh suffers from representing the image content appropriately. For this reason, irregular meshes are also used. Servais *et al.* [4], Al-Regib *et al.* [8], and Altunbaşak and Tekalp [7] employed content-based irregular meshes. A content-based mesh represents the object boundaries and non-stationary regions of the image better. In Figure 2.5, a content-based mesh is illustrated [4].

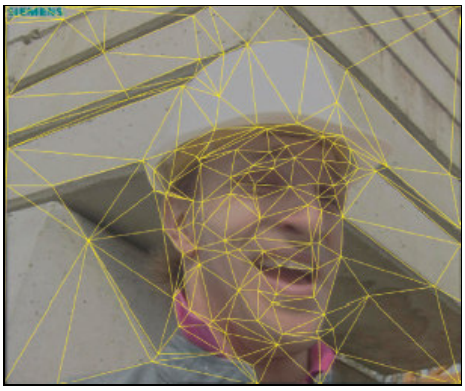


Figure 2.5 – A Content-Based Mesh

We have used the right-angled regular mesh with triangular patches (Figure 2.4 (b)) in our single-layer and two-layer hexagonal matching algorithm implementations. In object-based motion estimation studies, this mesh structure is employed in combination with non-uniform patches to obtain an accurate representation of video objects. Besides, a content-based irregular mesh is also applied to the same video objects for comparison purposes. The details of the object-based 2-D mesh design are presented in Chapter 3.

2.3.2. Hexagonal Matching Algorithm

The hexagonal matching algorithm (HMA) was developed by Nakaya and Harashima [1]. The algorithm performs motion estimation of the nodes (grid points) by iterative local minimization of the prediction error. This process is also called mesh refinement. The process may be applied immediately after the design of a two-dimensional regular mesh or after an initial estimation of the node motion vectors which is obtained by the block matching algorithm. We have chosen the latter since an initial estimation by a fast method like block matching reduces the time required for the grid points to converge to local or global minima.

As seen on the regular meshes of Figure 2.4, each grid point (except the ones on the frame boundary) is connected to six triangles, i.e. each grid point is one of the vertices of six triangles which form a hexagon surrounding the grid point. Mesh refinement updates the location of each grid point iteratively by minimizing the prediction error inside this surrounding hexagon. The steps of the algorithm are as follows:

i. A new grid point G_Z is selected as the node to be refined. Locations of the six surrounding grid points, G_A ~ G_F are kept fixed during the refinement process of G_Z (Figure 2.6).

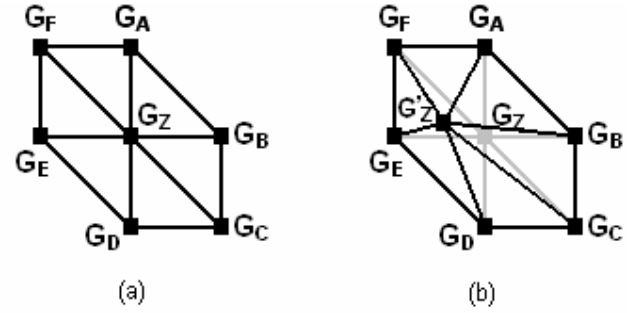


Figure 2.6 – Hexagonal Mesh Refinement

ii. G_Z is moved to an adjacent point G'_Z. For this candidate grid point location, we have deformed patches inside the hexagon as shown in Figure 2.6 (b).

iii. For each of these deformed patches of the reference frame, a transformation function is estimated to map the patch onto the corresponding undeformed patch of the current frame. An affine transformation is applied for this mapping which is described in Section 2.3.3.

iv. The predicted image inside the hexagon is synthesized by warping the six deformed patches onto the undeformed patches using the estimated transformation function.

v. The MAD between the predicted image and the current frame is calculated. This difference is assigned to the candidate grid point location G'_Z as its error value.

vi. Steps (ii) to (v) are repeated for each candidate grid point location G'_Z in the search range of G_Z.

vii. Among the candidate grid point locations, the one with the smallest error value is registered as the new location of G_Z.

viii. Steps (i) to (vii) are repeated for each grid point of the mesh.

The mesh refinement process described above is iterated until all grid points of the mesh converge to either local or global minima. An iteration of the process may not include refinement of some grid points. A grid point G_Z is refined in an iteration only if its location was refined in the previous iteration or location of at least one of its six surrounding grid points was refined after G_Z's last refinement. Hence, the number of grid points included in the refinement process decreases as the iterations continue and finally no improvement is observed in the prediction error. As a result, the iterative mesh refinement process is finished.

The hexagonal matching algorithm is an effective motion estimation method. Our simulation results indicate that the algorithm performs better than the block matching algorithm. In the Experimental Results Chapter, details of the experiments are presented. One drawback of the algorithm is its high computational cost. Faster algorithms as the ones proposed by Nakaya and Harashima [1] and Yu *et al.* [23] may be employed to reduce the computational cost.

2.3.3. Motion Compensation

Motion compensation can be performed to synthesize the content of a frame from another frame using the estimated grid point motion vectors. Mesh-based motion compensation is achieved by warping the undeformed mesh of the current frame onto the deformed mesh of the reference frame. The warping is done patch by patch to synthesize the frame content. Since the patches of the mesh are chosen as triangles, a six-parameter affine transformation is applied to warp the content of each triangle.

Affine Transformation

Affine transformation models translation, rotation and scaling of a patch in the current frame to the corresponding deformed patch in the reference frame [2]. The

intensity value of pixel (x, y) in the i^{th} synthesized patch $\hat{P}$ in the predicted frame K is calculated by the following formula

$$\hat{P}_i^k(x, y) = P_i^{k+1}(x', y') \tag{2.3}$$

and the affine transformation for the patch is given by

$$x' = a_{i1}x + a_{i2}y + a_{i3} \tag{2.4}$$

$$y' = b_{i1}x + b_{i2}y + b_{i3} \tag{2.5}$$

where (x', y') and (x, y) represent the pixel positions in the current and reference frames, respectively.

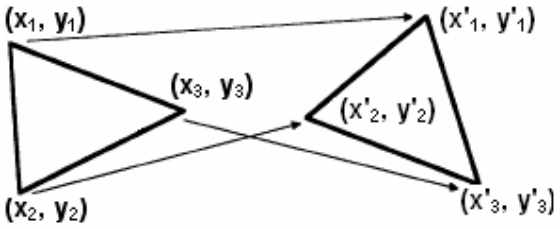


Figure 2.7 – Affine Mapping of Patches

Affine transformation has six parameters, a_{i1}, a_{i2}, a_{i3}, b_{i1}, b_{i2}, b_{i3}, and there exists a one-to-one matching between the vertices of a deformed patch and the vertices of an undeformed patch as shown in Figure 2.7. Therefore, the six parameters can be found by solving the equations 2.4 and 2.5 for three vertex positions of each patch. After finding the affine parameters of a deformed patch, for each pixel (x, y) in that patch, a matching pixel (x', y') in the current frame is calculated using these parameters. Then, intensity values of these matching pixels are assigned to the pixels of the synthesized patch. This procedure is applied to all patches of the deformed mesh. Finally, the contents of all patches are synthesized and the predicted reference frame is formed.

One problem related to the affine transformation is that the calculated position (x', y') does not always correspond to a valid pixel location. The transformation may produce non-integer x' or y' values. In this case, an interpolation is needed to determine the intensity value of pixel (x, y). We have employed the bilinear interpolation since it uses the nearest integer locations in both x-direction and y-direction to interpolate the intensity value.

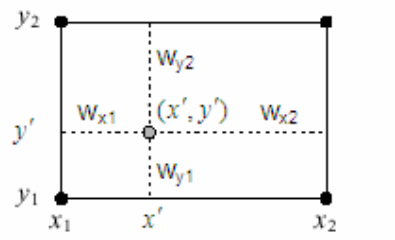


Figure 2.8 – Bilinear Interpolation

Assignment of Pixels to Triangles

One problem in motion compensation by affine transformation is the assignment of pixels to triangles to be able to perform warping. Although the assignment is simple in the case of undeformed regular triangles, more computational effort is required for the deformed triangles. A simple method to solve this problem is described below.

The triangle shown in Figure 2.9 has three vertices; $G_A(x_A, y_A)$, $G_B(x_B, y_B)$ and $G_C(x_C, y_C)$, and three line equations:

$$y = a_{AB}x + b_{AB} \qquad (\textit{Line } G_AG_B) \tag{2.6}$$

$$y = a_{AC}x + b_{AC} \qquad (\textit{Line } G_AG_C) \tag{2.7}$$

$$y = a_{BC}x + b_{BC} \qquad (\textit{Line } G_BG_C) \tag{2.8}$$

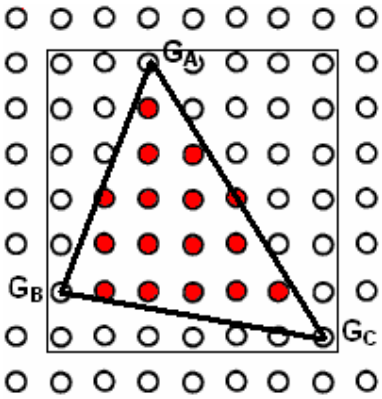


Figure 2.9 – Assignment of Pixels to Triangles

The pixel assignment procedure for this triangle is as follows:

i. Put the vertex $G_C(x_C, y_C)$ into the equation 2.6. ***If*** ($y_C > a_{AB}x_C + b_{AB}$), ***then*** label G_C as *"above"*, ***else*** label G_C as *"below"*.

Put the vertex $G_B(x_B, y_B)$ into the equation 2.7. ***If*** ($y_B > a_{AC}x_B + b_{AC}$), ***then*** label G_B as *"above"*, ***else*** label G_B as *"below"*.

Put the vertex $G_A(x_A, y_A)$ into the equation 2.8. ***If*** ($y_A > a_{BC}x_A + b_{BC}$), ***then*** label G_A as *"above"*, ***else*** label G_A as *"below"*.

ii. Select a pixel, $P_i(x_i, y_i)$, inside a previously defined window as shown in Figure 2.9.

iii. Put $P_i(x_i, y_i)$ into the equation 2.6. ***If*** ($y_i > a_{AB}x_i + b_{AB}$), ***then*** label the pixel as *"above"*, ***else*** label *the pixel* as *"below"*.

If the label of the pixel and the label of G_C are the same, then the pixel is on the same side of the line G_AG_B as G_C. Otherwise, it is on the other side.

Put $P_i(x_i, y_i)$ into the equation 2.7. ***If*** ($y_i > a_{AC}x_i + b_{AC}$), ***then*** label the pixel as *"above"*, ***else*** label *the pixel* as *"below"*.

If the label of the pixel and the label of G_B are the same, then the pixel is on the same side of the line G_AG_C as G_B. Otherwise, it is on the other side.

Put $P_i(x_i, y_i)$ into the equation 2.8. ***If*** ($y_i > a_{BC}x_i + b_{BC}$), ***then*** label the pixel as *"above"*, ***else*** label *the pixel* as *"below"*.

If the label of the pixel and the label of G_A are the same, then the pixel is on the same side of the line G_BG_C as G_A. Otherwise, it is on the other side.

iv. Assign the pixel to the triangle if the pixel is on the same side of the lines as the three vertices G_A, G_B, *and* G_C.

v. Repeat steps (ii) to (iv) until all pixels inside the window are evaluated.

This procedure assigns the pixels to the deformed triangles. The content of each deformed triangle is then synthesized by a warping operation based on affine transformation.

We have performed motion compensation to evaluate the performance of the motion estimation methods that have been implemented. The results are presented in detail in the Experimental Results Chapter.

2.4. Hierarchical Motion Estimation

Hierarchical motion estimation can be applied to obtain an improved motion field representation. There are different implementations of hierarchical motion estimation in both block-based and mesh-based approaches.

Hierarchical block matching algorithm performs motion estimation by searching the best matching blocks in each level starting with the lowest resolution image level. The estimated motion vectors of one level provide an initial prediction for the motion vectors of next higher resolution level. On the other hand, an alternative method, known as variable-size block matching uses the same original image resolution in all levels while dividing the blocks into smaller ones in each level. Although these hierarchical block matching methods can improve the

motion vectors resulting in higher PSNR values, the generated motion fields may still include some outliers [5-8].

In hierarchical mesh-based motion estimation, the common approach is to apply meshes with different densities in each layer of the estimation process. Toklu *et al.* [17] proposed to use hierarchical regular meshes. On the other hand, Van Beek *et al.* [18] employed coarse-to-fine content-based meshes in their hierarchical motion estimation process. Al-Regib *et al.* [8] extended both methods by using blurred and downsampled images in higher levels to reduce the computational complexity.

Progressive motion estimation is a hierarchical mesh-based motion estimation technique proposed by Park *et al.* [2]. This technique provides hierarchical partial mesh refinement according to the motion activities. The technique employs regular meshes in each layer but also features the advantages of a content-based mesh. It performs the hierarchical refinement without additional overhead to indicate the motion-active regions. Hence, progressive motion estimation technique achieves lower bit rates than the other mesh-based hierarchical motion estimation methods described above. The technique also shows improvement over the block-based methods by eliminating block distortions. The detailed information is presented in the next section.

2.4.1. Progressive Motion Estimation

As stated in Section 2.3.1, a regular mesh is not designed according to the image content. As a result, mesh refinement based on a regular mesh may not produce accurate motion fields especially for motion-active regions of the image. A solution for this problem is to apply a hierarchical mesh refinement procedure as proposed by Park *et al.* [2]. They developed a progressive motion estimation technique which uses layered regular meshes designed according to the motion activities in the image. This section describes the proposed technique.

Progressive motion estimation employs regular meshes in a layered manner. The technique starts with uniform patches covering the whole image in the first layer. Then, in the next layers, smaller patches are applied only to the regions which contain high motion activities. The process is illustrated in Figure 2.10.

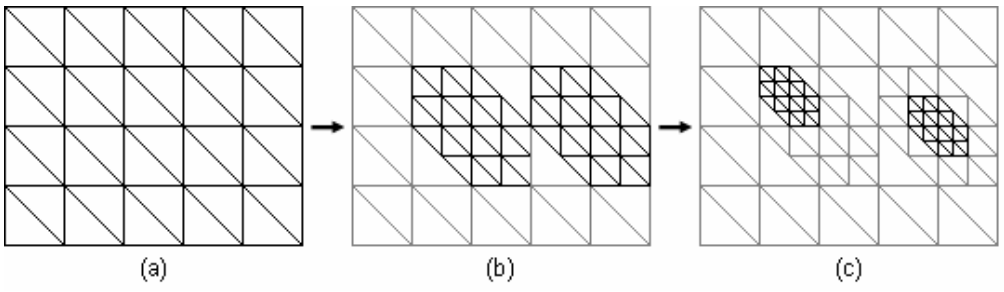


Figure 2.10 – Progressive Motion Estimation Layers

In the first step, the technique applies hexagonal matching algorithm as mentioned in Section 2.3.2. A 2-D mesh is designed for the current frame as seen in Figure 2.11 (a). Then, an image is synthesized by performing mesh refinement and motion compensation (Figure 2.11 (b)). This predicted image provides a coarse estimation of the reference frame.

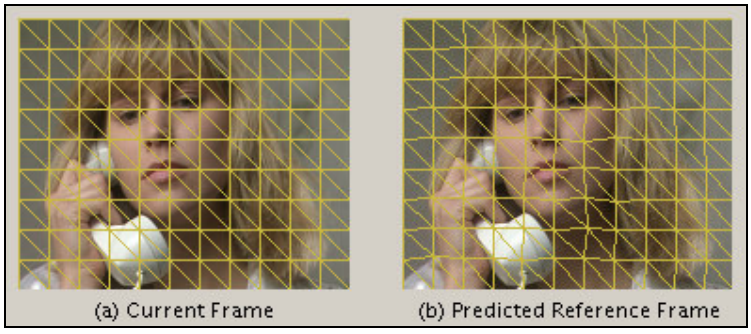


Figure 2.11 – Single-Layer Mesh Refinement

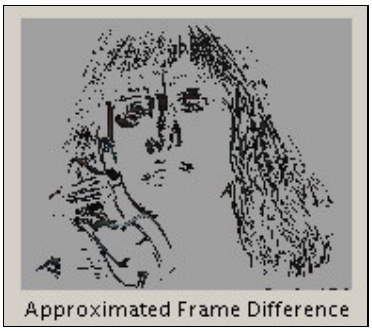


Figure 2.12 – Approximated Frame Difference

The next step starts with obtaining an approximated difference map. It is generated by subtracting the predicted reference frame from the current frame. Figure 2.12 shows an example of an approximated difference map. The approximated difference gives information about the motion-active regions. These regions are identified by a comparison based on the *variance*: If a patch difference has a greater variance, v_P, than the variance of the frame difference, v_F, then a motion discontinuity exists in that patch. The variance of the frame difference is given by

$$v_F = \frac{1}{MN}\sum_{j=1}^{M}\sum_{i=1}^{N}(f(i,j)-\bar{f})^2 \tag{2.9}$$

where M and N are the frame dimensions, $f(i,j)$ is the intensity value of pixel (i,j), and $\bar{f}$ refers to the mean frame difference. The variance of the patch difference is given by

$$v_P = \frac{1}{K}\sum_{i=1}^{K}(f_P(i)-\bar{f}_P)^2 \tag{2.10}$$

where K is the number of pixels in the patch, $f_P(i)$ is the intensity value of i^{th} pixel in the patch, and $\bar{f}_P$ denotes the mean patch difference.

Figure 2.13 (a) indicates the patches with motion discontinuities on the approximated difference map of Figure 2.12.

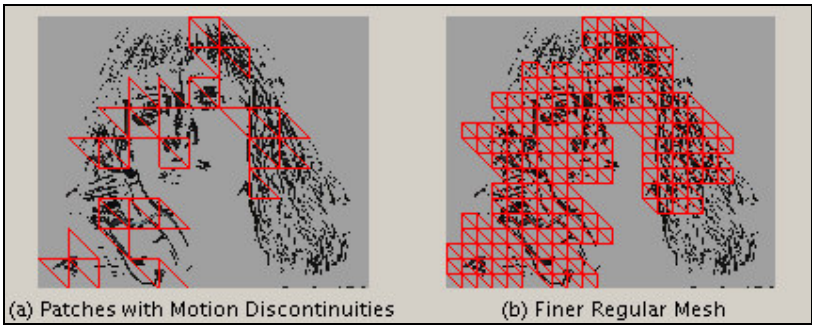


Figure 2.13 – Motion-Active Regions

In the following step, a finer regular mesh is applied to the regions containing motion discontinuities (Figure 2.13 (b)). A finer regular mesh refers to a mesh with smaller patches. In Figure 2.14, the current frame with the finer mesh is shown. On this second layer denser mesh, the refinement procedure is performed again as in the first step. Consequently, we have a hierarchical mesh refinement process.

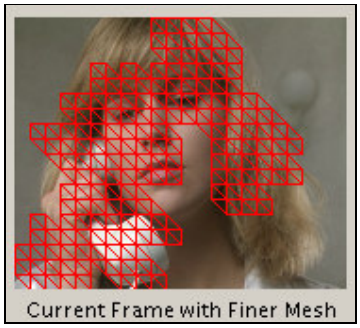


Figure 2.14 – Second Layer Denser Mesh

The progressive motion estimation technique provides a partial mesh refinement process as described. We have applied the technique using two-layer meshes. The results indicate that the technique shows improvement over the single-layer block matching and hexagonal matching algorithms.

CHAPTER 3

2-D MESH-BASED VISUAL OBJECT REPRESENTATION

3.1. Introduction

Object-based video representation which is recommended in the MPEG-4 standard [19] provides a solution for the problem with the 2-D frame-based motion estimation models: accurate representation of individual video objects in a scene.

MPEG-4 describes the video sequences as video objects. Segmentation of arbitrarily shaped video objects from the background and other objects is of primary importance for the effective compression and manipulation of these objects. This task is difficult especially in the presence of complicated backgrounds. After the segmentation, shape and motion of the objects need to be modeled. 2-D mesh-based models provide a good representation of shape and motion of video objects. A 2-D dynamic mesh also allows the animation of video objects [3, 12, 20-22].

This chapter focuses on the use of 2-D meshes for visual object representation. Section 3.2 gives an introductory information about the extraction (or segmentation) of video objects. Then, 2-D object-based mesh design methods are presented in Section 3.3. Section 3.4 describes object-based motion estimation techniques and finally, video object manipulation topic is presented in Section 3.5.

3.2. Video Object Extraction

Video object extraction (or segmentation) is one of the fundamental problems in object-based video representation. A simple solution for this problem is thresholding. Thresholding performs extraction on the basis of the difference in

colors (or intensities) of an object and background. It works well in the scenes as the one shown in Figure 3.1 (a).

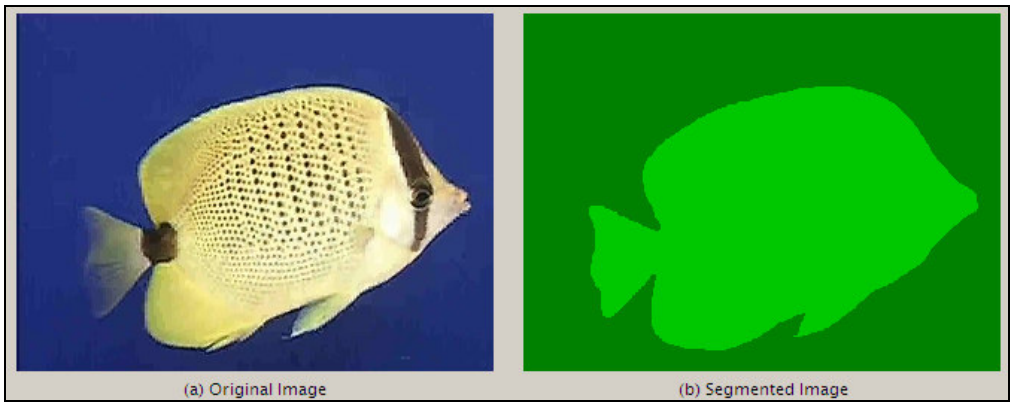


Figure 3.1 – Thresholding

In this scene, a color band is specified for background. Then, pixels inside this band are assigned a color value, and pixels outside the band are assigned another color value. This procedure generates the segmented image in Figure 3.1 (b).

On the other hand, segmentation gets complicated for some other scenes where a foreground object cannot be extracted from background by a simple color thresholding. An example for such scenes is shown in Figure 3.2 (a).

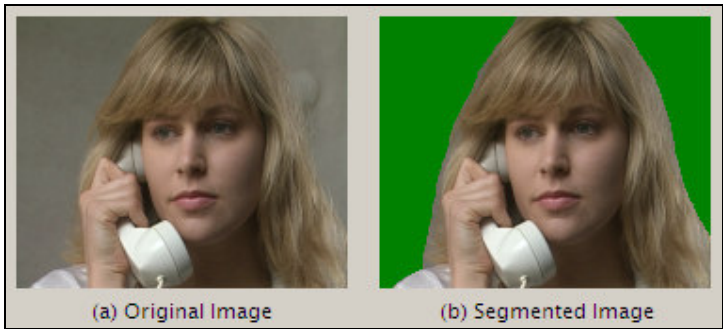


Figure 3.2 – Manual Object Extraction

In this case, object extraction can be done manually by the user or by some semi-automatic or automatic extraction methods as the ones proposed by Toklu *et al.* [25], Yang *et al.* [12] and Fan *et al.* [11]. Considering the difficulty of accurate object extraction, in our studies, we performed manual object extraction unless a simple color thresholding gives good results. Figure 3.2 (b) illustrates the image obtained by manual object extraction for the sample scene in Figure 3.2 (a).

3.3. Object-Based 2-D Mesh Design

Video object extraction is followed by the design of a 2-D object mesh. Similar to the full-frame mesh design (Chapter 2), regular or irregular meshes can be applied in the design of an object mesh. The difference is that this time the mesh generation is limited to the region covered by the video object (Figure 3.8 and Figure 3.9).

Object-based (2-D) mesh design process includes three stages: determination of nodes on the object boundary, determination of nodes inside the object, and triangulation between all nodes. These stages are described in detail in the following sections.

3.3.1. Boundary Node Selection

We have employed regular meshes in full-frame mesh-based motion estimation implementations as mentioned in Chapter 2. In a regular mesh, nodes are placed at fixed distances over the image (Figure 2.4). Hence, this kind of mesh cannot fit to the object boundary without modification. Our algorithm determines the boundary nodes by modifying the grid point positions of the regular mesh near the object boundary.

After performing object segmentation, the boundary node selection algorithm proceeds as follows:

i. Each grid point of the regular mesh is labeled as “inside” or “outside” of the object referring to the segmented image as shown in Figure 3.3.

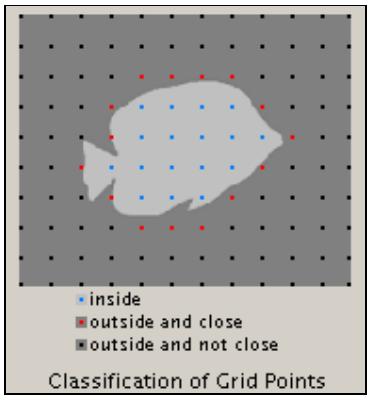


Figure 3.3 – Classification of Grid Points

ii. Among the grid points which are “outside”, the ones that are close to the object boundary are found and marked as “close” (Figure 3.3). The closeness criterion is:

> ***If*** *at least one neighbor of the grid point is “inside”,* ***then*** *the grid point is “close” to the object boundary*.

iii. A grid point which is “outside” and “close” is moved on the line towards one of its “inside” neighbors until the grid point reaches the object boundary (Figure 3.4). This candidate position and its distance to original grid point position are recorded. The grid point is put back to its original position. This procedure is repeated for all “inside” neighbors of the grid point and the candidate with the minimum distance is designated as the winner. Then, the grid point is moved to the winner position.

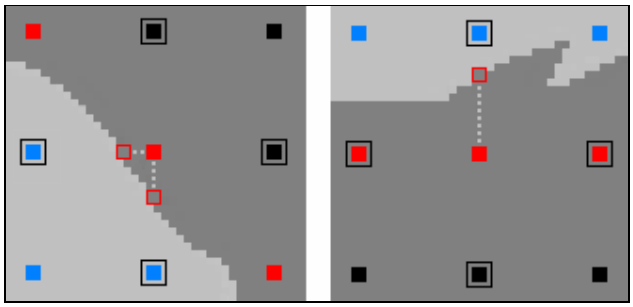


Figure 3.4 – Movement of "Close" Points

iv. Step (iii) is repeated for all grid points that are "outside" and "close". Finally, we obtain a boundary representation for the video object by determination of all boundary nodes as seen in Figure 3.5.

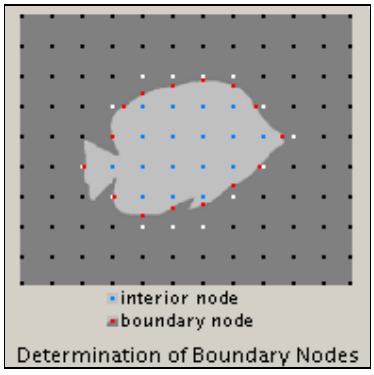


Figure 3.5 – Determination of Boundary Nodes

This algorithm does not make any modification on the "inside" points; rather it modifies the positions of the "outside" points near the boundary. The reason is that we want to preserve the uniform mesh inside the object to be able to perform object-based progressive motion estimation which is described in Section 3.4.3.

After the determination of boundary nodes, the boundary polygon is formed by sorting and connecting those nodes. The edges of this polygon are used as constraints for the generation of irregular patches inside the object as described in the triangulation stage (Section 3.3.3).

In our simulations, we have used the same boundary nodes and boundary polygon for both regular and irregular (content-based) mesh approaches for comparison purposes.

3.3.2. Interior Node Selection

Determination of the nodes inside a video object can be performed by one of the two popular approaches: the regular (uniform) mesh approach and the content-based mesh approach [3]. We have applied both of them in our object-based motion estimation and compensation studies. In the following sections, interior node selection by these two approaches is mentioned.

3.3.2.1. Regular Mesh Approach

In this approach, interior nodes are uniformly placed inside the video object as seen in Figure 3.5. Uniform distribution of the nodes provides lower bit rates than a non-uniform distribution in coding. Toklu *et al.* [10] applied this approach to synthetic object transfiguration.

3.3.2.2. Content-Based Mesh Approach

In the content-based mesh approach, interior nodes are placed according to some features of the region inside the video object such as spatial edges and motion activities. Content-based node selection algorithm proposed by Altunbasak *et al*. [7, 9] is used in our implementations. The algorithm for the object-based representation is as follows:

i. Estimate a 2-D dense motion field between the current and the reference frames. Label all pixels in the current frame as “unmarked”.

ii. Compute an average displaced frame difference, DFD_{avg}, inside the object as

$$DFD_{avg} = \frac{\sum_{(x,y)} (DFD(x,y))^p}{K} \tag{3.1}$$

where K is the number of unmarked pixels inside the object, p is a positive number (selected as 2), $DFD(x,y)$ denotes the displaced

frame difference at pixel (x, y), and the summation is over all unmarked pixels inside the object.

iii. For each unmarked pixel inside the object, compute a cost function $C(x, y)$ as

$$C(x, y) = \left|I_x(x, y)\right| + \left|I_y(x, y)\right| \tag{3.2}$$

where $I_x(x, y)$ and $I_y(x, y)$ denote the partials of the intensity with respect to x and y coordinates evaluated at pixel (x, y) (Figure 3.6). The cost is a function of spatial intensity gradient. This eventually results in that interior nodes are placed at the spatial edges.

Figure 3.6 – Image Derivatives

iv. Find the unmarked pixel with the highest $C(x, y)$ which is not closer to any other previously selected node point (including the boundary nodes) than a prespecified distance. Label this point as a node point.

v. Grow a circle around this node point until $\sum (DFD(x, y))^p$ in this circle is greater than DFD_{avg} or the circle exceeds the object boundaries. Label all pixels inside this circle as "marked". Set $DFD(x, y)$ at these pixels as zero. Figure 3.7 illustrates the procedure.

Figure 3.7 – Interior Node Selection

vi. Go to step (ii) until a desired number of node points, *N*, are selected or the distance criterion in step (iv) is violated.

The next stage is triangulation between the boundary and the interior nodes which is described in the following section. This algorithm places the nodes in such a way that after triangulation, each patch has approximately the same DFD value.

As shown in Figure 3.7, a high temporal activity is represented by a small circle, while a low temporal activity is represented by a large circle. The reason for labeling all pixels in a circle as marked is that only one node point is allowed in a circle. This constraint provides a node point distribution proportional to the temporal activity [7].

Node point distribution is directly affected by the parameters of the algorithm. *Minimum allowed distance between any two node points* and *desired number of node points* are some of those parameters. In our implementations, we tuned them for the best performance of the algorithm. In the Experimental Results Chapter, we discuss the effect of these parameters on the results.

3.3.3. Triangulation

In the scope of object-based mesh design, triangulation refers to generation of a mesh by partitioning the video object into triangular patches. This is achieved by connecting all boundary and interior nodes of the object by a selected method. Hence, these nodes become the vertices of the patches of the mesh (Figure 3.8 and Figure 3.9).

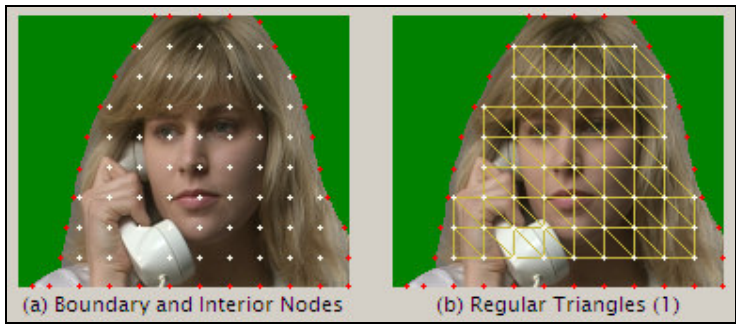
The regular and content-based mesh approaches also differ in the way of triangulation as described below.

3.3.3.1. Regular Mesh Approach

A 2-D regular mesh inside the object is generated in the same way as in the full-frame mesh design procedure (Chapter 2). However, this time, the regular mesh is generated between only the interior nodes as seen in Figure 3.8 (b). If the outside nodes were also used to cover the whole region inside the object, the mesh would exceed the object boundaries as shown in Figure 3.8 (c). Hence, it would result in an inaccurate object representation around the boundary.

In order to solve the problem near the boundary, another triangulation method called the Constrained Delaunay Triangulation is applied. The details of the method are given in the next section. Constrained Delaunay Triangulation produces irregular patches in the region between the boundary nodes and the nearest interior nodes which is not covered by the regular mesh. Hence, the whole region inside the object is well approximated by the combination of regular and irregular patches (Figure 3.8 (d)).

We have used this combination to adapt the single layer hexagonal matching and progressive motion estimation methods to object-based motion estimation as described in Section 3.4.



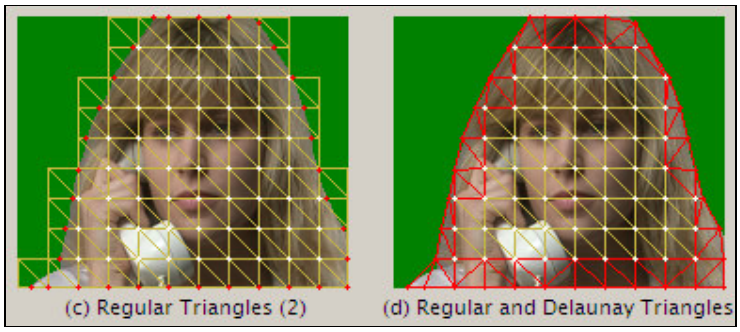


Figure 3.8 – Triangulation – Regular Mesh Approach

3.3.3.2. Content-Based Mesh Approach

Content-based triangulation follows the selection of boundary and content-based interior nodes. The Constrained Delaunay Triangulation [13] is employed to generate a content-based object mesh. The triangulation is performed between all boundary and interior nodes resulting in a non-uniform structure.

This method constructs the object mesh using the edges of the boundary polygon as constraints. Using these constraints guarantees that all triangles are constructed inside the object. One nice property of Delaunay Triangulation is that it eliminates too small angles between the triangle edges [3, 13].

We have used the software, *Triangle*, provided by Shewchuk [13] to construct a content-based Delaunay mesh inside the object. In Figure 3.9 (b), a Delaunay mesh generated by this software is illustrated.

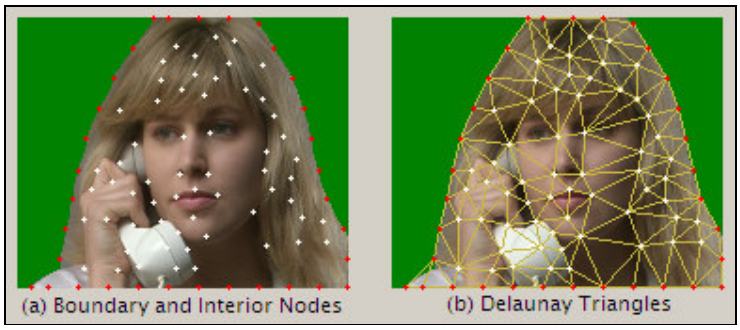


Figure 3.9 – Triangulation – Content-Based Mesh Approach

3.4. Object-Based Motion Estimation

Motion estimation in object-based video processing refers to generating a motion field by updating the positions of the boundary and the interior nodes of an object. Altunbasak *et al*. [7] stated that there are two approaches for node motion estimation: i) estimating the motion vectors at each node independently of each other (e.g., performing block matching at each node); ii) estimating motion vectors optimized for warping transformations (e.g., applying a constrained search procedure such as hexagonal matching algorithm [1]). They employed the former approach in their paper due to the high computational requirements of the latter approach.

In object-based motion estimation implementations, we have employed both approaches mentioned above; block matching is used for a coarse estimation of node motion vectors, and then, single layer hexagonal matching (or generally polygonal matching) is applied to refine these motion vectors. Besides, for the specific case of object-based progressive motion estimation, we have also employed a second layer mesh refinement procedure as in the full-frame progressive motion estimation process.

The following sections give information about the application of those motion estimation techniques to object-based video representation.

3.4.1. Initial Estimation by Block Matching

A block matching procedure similar to the classical block matching algorithm described in Chapter 2 is used to obtain an initial prediction of the 2-D motion field representing the object motion. The main difference of this procedure is in the way of locating the blocks. While in full-frame motion estimation whole image is segmented into non-overlapping rectangular blocks independently of the image content, this time we need to consider the shape of the object as a constraint. The procedure determines the locations of blocks on the basis of boundary and interior node positions.

Figure 3.10 illustrates the possible blocks for different types of nodes. For a node, four or five (depending on the type of node) candidate search blocks are formed, each having the node as one of its corners or on its center. Then, for each

candidate, the number of pixels which belong to the region inside the object is calculated. The candidate with the maximum number of pixels inside the object is selected as the block to be searched in the reference frame. This selection procedure is repeated for all boundary and interior points to locate their search blocks in the object. These blocks are then searched in the reference frame to find the best matches as in the classical block matching algorithm.

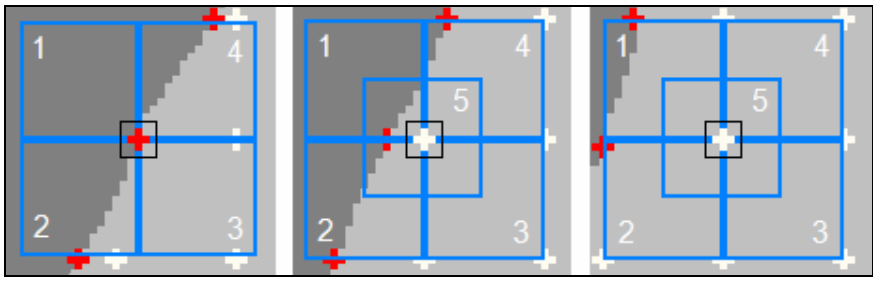


Figure 3.10 – Candidate Search Blocks for Different Node Locations

As shown in Figure 3.10, five candidate blocks are formed for an interior node while four is sufficient for a boundary node. The reason for forming a fifth candidate for interior nodes is that an interior node may be far from object boundary. In this case, this center block remains completely inside the object as seen in Figure 3.10 (c). Hence, this block is selected for the search in the reference frame since it provides a better representation of the motion field around the node than the other four candidates.

The search in the reference frame is limited to a search window and the MAD is used as the matching criterion as in the classical block matching algorithm.

The procedure given in this section is employed as the first step of motion estimation and provides a coarse estimation of the motion vectors at the nodes. It is followed by a mesh refinement procedure to find the optimal node vectors as described in the next section.

3.4.2. Mesh Refinement by Polygonal Matching

Refinement of the 2-D object mesh is realized by a polygonal matching algorithm (PMA) which is based on the hexagonal matching algorithm (HMA) described in Chapter 2. In fact, PMA is a generalized version of HMA for different kinds of polygons. Hence, mesh refinement proceeds very similar to the refinement process proposed in HMA.

In a right-angled regular mesh covering whole frame as shown in Figure 2.4, a node (grid point) is connected to six triangles unless it is on the frame boundary. On the other hand, a node point in an object mesh can be connected to different number of triangles due to the irregular structure of the mesh. Figure 3.11 shows two examples for different node placements in both the regular mesh and the content-based mesh approaches.

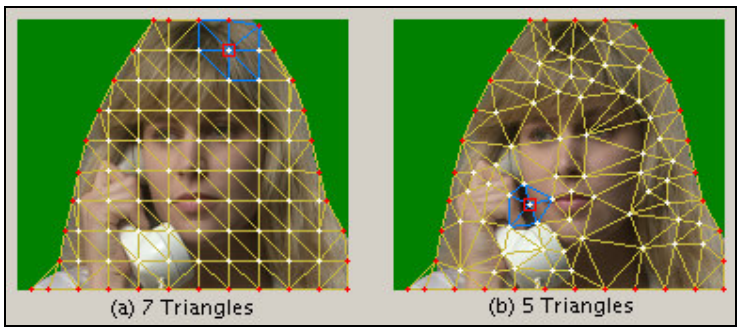


Figure 3.11 – Grid Point Examples

Although each node shown in the figure is connected to different number of triangles, the same mesh refinement procedure is applied to all of them: iterative local minimization of the prediction error within the bounding polygon of the node. The procedure follows the same steps as the mesh refinement procedure of HMA. All of the surrounding nodes of G_Z are kept fixed and G_Z is sequentially moved to adjacent positions in a limited search region to find the optimum displacement which yields the minimum prediction error (Figure 3.12). The procedure is repeated iteratively for all nodes of the mesh until all of them converge to either local or global minima.

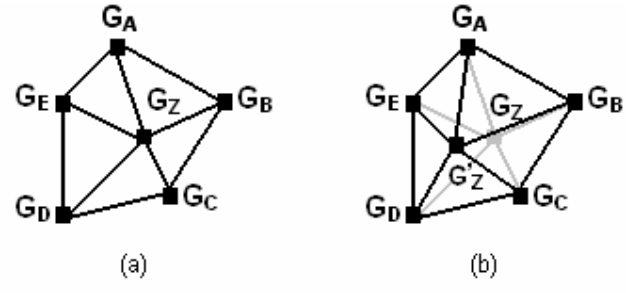


Figure 3.12 – Polygonal Mesh Refinement

In the content-based mesh approach, motion estimation phase is completed with this refinement procedure. However, in the regular mesh implementations, we refine the object motion field further by applying a second layer mesh as in the progressive motion estimation technique.

3.4.3. Progressive Motion Estimation

This technique, as described in Chapter 2, provides a hierarchical partial refinement. In each layer, denser regular meshes designed according to motion activity are employed (Figure 2.10). Park *et al.* proposed the technique with applications to full-frame regular meshes in their paper [2]. We have used the technique in object-based motion estimation by making some modifications.

Following the polygonal mesh refinement given in the previous section, we apply partial refinement. This time, motion-active regions inside the object are identified from the approximated difference map (Figure 3.13 (a)). Then, a denser mesh is applied inside the region covered by the regular mesh of first layer (Figure 3.13 (b)).

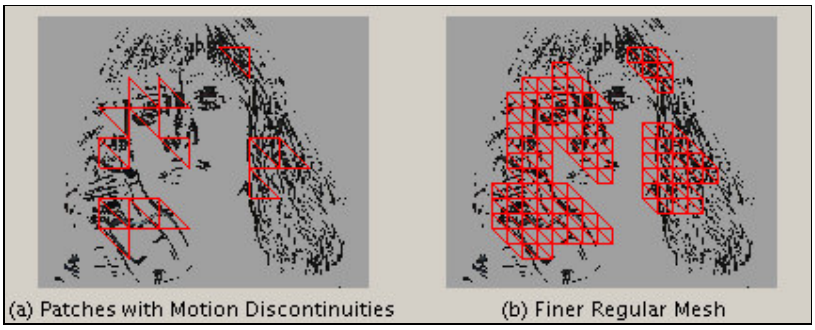


Figure 3.13 – Approximated Frame Difference

While identifying motion-active regions which are shown in Figure 3.13 (a), individual patch variances (given by Equation 2.10) are compared with the variance of the object difference given by

$$v_O = \frac{1}{K}\sum_{i=1}^{K}(f_O(i) - \bar{f}_O)^2 \tag{3.3}$$

where K is the number of pixels inside the object, $f_O(i)$ is the intensity value of i^{th} pixel inside the object, and $\bar{f}_O$ denotes the mean object difference. The patches whose variances are greater than the variance of object difference are included in the partial refinement process as described in Chapter 2.

Progressive motion estimation technique with the modifications described in this section provides a hierarchical refinement for object-based video processing. It produces better results than single-layer hexagonal matching algorithm and block matching algorithm according to our experiments. (See the Experimental Results Chapter).

3.5. Video Object Manipulation

2-D mesh-based models enable several functionalities for manipulation of video objects such as animating a still image of one object by motion parameters of another similar object and combining natural and synthetic objects interactively within a unified framework. In addition, the mesh structure allows spatio-temporal video interpolation [3].

In this section, video object manipulation is described in detail.

3.5.1. Object Transfiguration

Object transfiguration refers to synthesizing an animated video object from a still image of the same or a replacement object by texture mapping based on a dynamic 2-D mesh representation. Object transfiguration is classified as self-transfiguration and synthetic transfiguration. Self-transfiguration refers to animating a video object from a still image of the same object using its dynamic mesh representation. It is employed in object-based video compression. Transmitting texture maps only at selected key frames and animating these texture maps to obtain the intermediate frames may improve the efficiency of compression [3, 10, 17].

Synthetic transfiguration refers to replacing a natural video object in a video clip by another video object. The replacement video object may be animated from one of its still images using the motion vectors of the object to be replaced. This requires accurate motion tracking of the existing object. The 2-D dynamic mesh representation which is included in the MPEG-4 standard allows tracking of object motion through the video sequence [3, 19].

3.5.2. Augmented Reality

Augmented reality refers to merging synthetic (computer generated) video, images, text and/or graphics with natural moving images (video) to create enhanced display information. The synthetic content must remain in perfect registration with the moving real images [3, 19].

Merging synthetic or natural content with a moving object can be performed by first tracking the motion of the existing object using 2-D dynamic meshes. Then, registration of the augmentation object with the initial appearance of the object to be augmented is performed. Finally, 2-D mesh-based texture mapping is applied to obtain the new merged sequence. The 2-D mesh-based texture mapping can be performed in a way similar to self-transfiguration: the animation parameters obtained by motion tracking of the existing natural object are applied to the composite object [3, 19].

In the Experimental Results Chapter, we present an application in which a text is augmented onto a moving fish.

3.5.3. Spatio-Temporal Interpolation

Spatio-temporal interpolation is another functionality enabled by 2-D mesh-based models. Mesh-based spatial interpolation is achieved by first scaling the object mesh, and then applying texture mapping onto the scaled mesh. This technique may be applied for performing zooming in and out on an object through the video sequence [3]. In the Experimental Results Chapter, a mesh-based spatial interpolation application for the moving fish sequence is presented.

Mesh-based temporal interpolation is performed by computing an intermediate mesh geometry between the two given meshes and then applying forward and/or backward texture mapping from one or both of the given object texture maps. Linear interpolation of node points is a way of computing the intermediate mesh geometry in the presence of two original meshes. Mesh-based temporal interpolation may be applied in the case of frame rate up-conversion [3].

CHAPTER 4

EXPERIMENTAL RESULTS

4.1. Introduction

In this chapter, the experimental results of the algorithms described in the previous chapters are given. The algorithms are implemented in C++ using Borland C++ Builder 6.0 IDE and the OpenCV library is used for the implementation of some image processing algorithms. The graphical user interface is shown in the following figure.

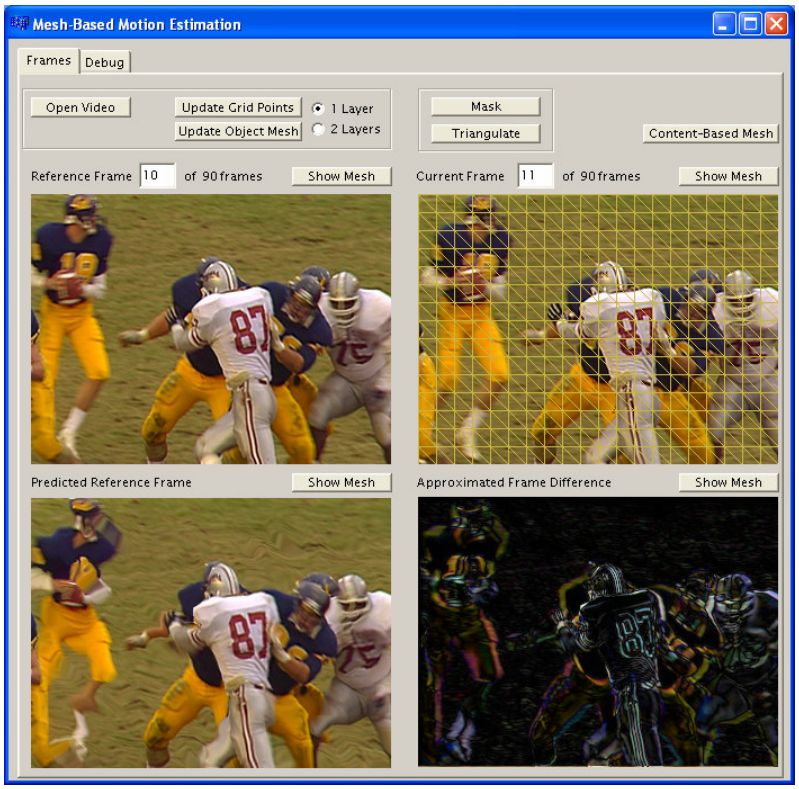


Figure 4.1 – The Graphical User Interface

This interface enables the user to perform several tasks related to motion estimation and compensation. The user can select a frame of a video sequence for displaying, generate a regular or an irregular mesh on a selected frame or start the motion estimation process between two frames. The user is also informed about the results of the experiments.

We have conducted the experiments on the selected video sequences by using this user interface. The following section gives information about the details of the experiments and results.

4.2. The Experiments

The first two experiments were conducted to compare the performance of the algorithms described in the previous chapters. The tests were performed in both frame-based and object-based video representations. The algorithms were compared in terms of peak signal-to-noise ratio (PSNR) and entropy.

The PSNR is commonly used as a measure of the quality of a predicted image in video compression. It is defined via the mean square error given by

$$MSE = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left[s_I(m,n) - s_P(m,n)\right]^2 \qquad (4.1)$$

where *M* and *N* are the frame dimensions, $s_I(m, n)$ is the luminance value of the pixel *(m, n)* in the original frame, and $s_P(m, n)$ denotes the luminance value of the pixel *(m, n)* in the predicted frame. The PSNR is defined as

$$PSNR = 20\log_{10}\left(\frac{I_{MAX}}{\sqrt{MSE}}\right) \qquad (4.2)$$

where I_{MAX} is the maximum intensity value of the pixels in the image. I_{MAX} is 255 for our test sequences.

Entropy is a measure of average information. Entropy of the difference between the original and predicted frames can be used to measure the quality of motion compensation. Entropy of an 8-bit image is given by

$$H_X = -\sum_{i=0}^{255} p(i) \log_2 p(i) \tag{4.3}$$

where $p(i)$ denotes the probability of the i^{th} intensity level in the image. A high entropy value indicates a less compressible image.

In the third experiment, application of mesh-based modeling to video object manipulation was performed.

The “Suzie” sequence used in the first two experiments has 150 frames of 176 x 144 size and the “Bream” sequence employed in the third experiment has 60 frames of 352 x 288 size.

4.2.1. Experiment 1: 2-D Frame-Based Motion Estimation and Compensation for “Suzie” Sequence

The aim of this experiment is to compare the block-based and regular mesh-based methods in frame-based video representation. The experiment was performed using one-layer block matching, and both one-layer and two-layer hexagonal matching algorithms. Two-layer hexagonal matching was realized by the use of the progressive motion estimation technique.

In the experiment, current and reference frames were selected as consecutive frames. For instance, when the second frame of the sequence was selected as the current frame, the first frame was used as the reference frame. Motion estimation was performed in the backward direction for all of the three algorithms. The resulting motion vectors were then used to generate the motion-compensated reference frame. The PSNR values were calculated based on the difference between the original and motion-compensated reference frames.

In the following table, motion estimation parameters for the three algorithms are listed.

Table 4.1 – Frame-Based Motion Estimation Parameters for the Block-Based and Regular Mesh-Based Algorithms

	1-Layer Block Matching	1-Layer Hexagonal Matching	2-Layer Hexagonal Matching
Block / Patch Size (pixels)	16 x 16	16 x 16	8 x 8
Search Range (pixels)	± 3	± 3	± 3
Number of Iterations	1	Up to Local or Global Minima	Up to Local or Global Minima
Mesh Connectivity Range (pixels)	Not Applicable	± 7	± 3

Figure 4.2 shows the PSNR values over all frames of the “Suzie” sequence for the three algorithms. In Table 4.2, the average PSNR values are also given. The results indicate that the two mesh-based algorithms perform more accurate motion compensation than the block matching algorithm. Moreover, the two-layer hexagonal matching algorithm realized by the progressive motion estimation technique shows improvement over the one-layer hexagonal matching algorithm. The average improvement is 0.16 dB which is close to the improvement achieved by Park *et al.* [2].

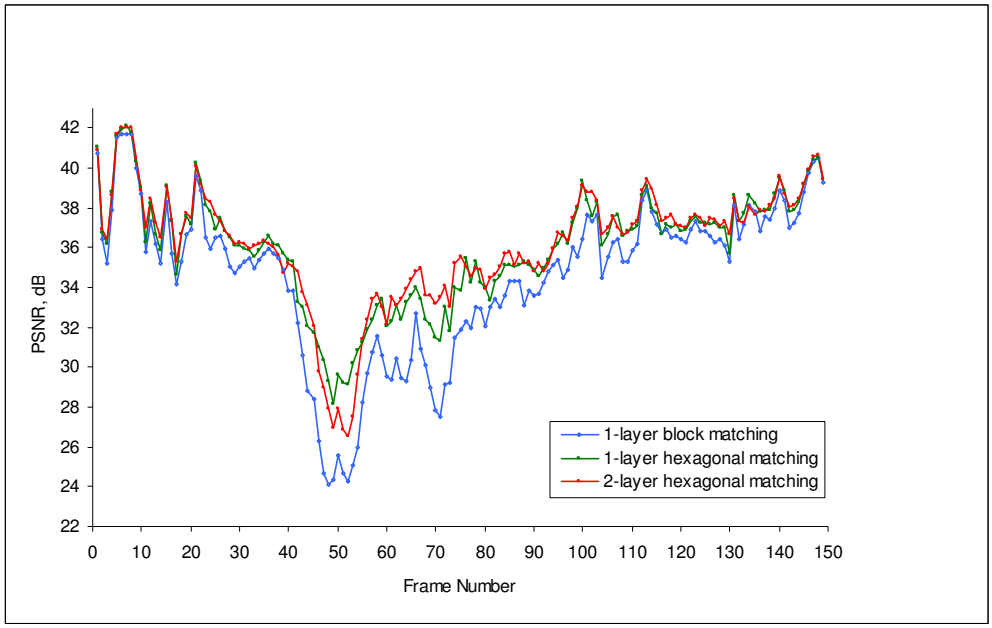


Figure 4.2 – Comparison of PSNR Values for All 150 Frames of "Suzie" Sequence

Table 4.2 – Average PSNR Values for All 150 Frames of "Suzie" Sequence

	1-Layer Block Matching	1-Layer Hexagonal Matching	2-Layer Hexagonal Matching
PSNR (dB)	34,60	35,99	36,15

The "Suzie" sequence contains different types of motion. Performance of the algorithms heavily depends on the type of motion. For this reason, we have also evaluated the algorithms for two specific cases.

In the first case, frames from 11 to 30 of the sequence were considered. In this range, smooth transitions (small changes in pixel positions) occur between the consecutive frames. Hence, all of the algorithms yielded high PSNR values as shown in Figure 4.3 and Table 4.3. Another feature of the frames in this range is that they contain "blinking", an example of local motion. This motion is limited to a small region. In this case, the two-layer hexagonal matching algorithm performed

better than the single-layer hexagonal matching algorithm since smaller patches employed in the second layer of the mesh refinement procedure represent the local motion better. As a result, the difference between the average PSNR values of the two algorithms (0,24 dB) was larger than the difference for the entire sequence (0,16 dB).

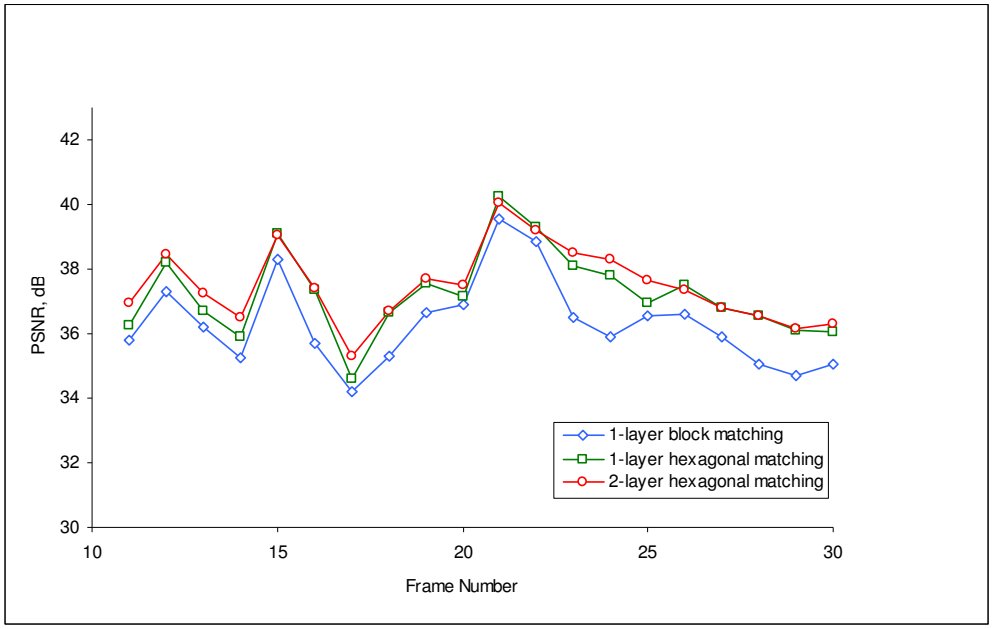


Figure 4.3 – Comparison of PSNR Values for Frames 11 to 30 of "Suzie" Sequence

Table 4.3 – Average PSNR Values for Frames 11 to 30 of "Suzie" Sequence

	1-Layer Block Matching	1-Layer Hexagonal Matching	2-Layer Hexagonal Matching
PSNR (dB)	36,31	37,25	37,49

In the second case, frames from 41 to 60 were used. These frames contain occluded and discovered areas. As a result, the algorithms produced lower PSNR

values compared to the average values of the entire sequence. This result was as expected since we did not employ any occlusion handling mechanism.

In this range, transitions between the consecutive frames are generally not smooth. As a result, grid point motion vectors tend to be large. However, because of the grid point connectivity constraint, each grid point was allowed to move in a small region (±3 pixels) in the second-layer of the hierarchical mesh refinement procedure. Hence, the two-layer hexagonal matching algorithm yielded less accurate motion compensation than the one-layer hexagonal matching algorithm. Comparison of the algorithms in terms of the PSNR values is given in Figure 4.4 and Table 4.4.

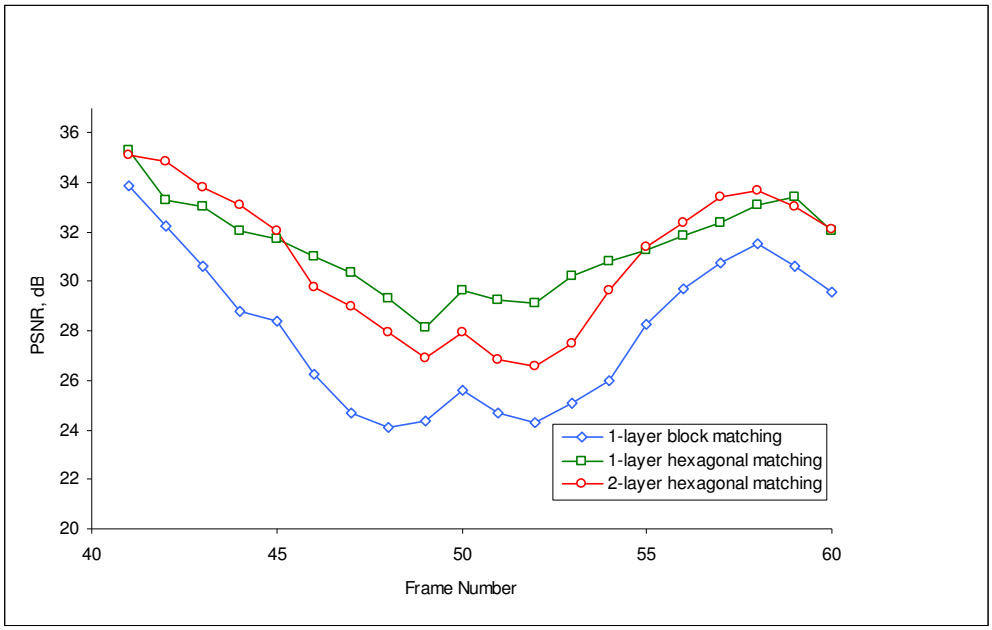


Figure 4.4 – Comparison of PSNR Values for Frames 41 to 60 of "Suzie" Sequence

Table 4.4 – Average PSNR Values for Frames 41 to 60 of "Suzie" Sequence

	1-Layer Block Matching	1-Layer Hexagonal Matching	2-Layer Hexagonal Matching
PSNR (dB)	27,97	31,37	30,85

4.2.2. Experiment 2: 2-D Object-Based Motion Estimation and Compensation for "Suzie" Sequence

In this experiment, performances of the block-based and mesh-based algorithms were measured and compared in object-based video representation. Backward motion estimation was applied again. This time, motion vectors were calculated only for the segmented video object instead of the entire frame. Then, these motion vectors were used to predict an object in the reference frame. Comparisons were performed in terms of the PSNR and entropy based on the difference between the original and predicted objects of the reference frame.

4.2.2.1. Comparison of Block-Based and Regular Mesh-Based Algorithms

In the first part of the experiment, one-layer block matching and both one-layer and two-layer hexagonal matching algorithms were compared as in experiment 1. Two-layer hexagonal matching was realized by using the progressive motion estimation technique. The same motion estimation parameters were used for the three algorithms as the ones in experiment 1.

Comparisons have been performed for the two specific cases described in experiment 1. The results were very similar to the results of the 2-D frame-based video representation experiment. In the first case, two-layer hexagonal matching algorithm showed improvement over the other two algorithms due to the motion in small regions inside the video object and smooth transitions between the consecutive frames. On the other hand, in the second case where occluded and discovered areas exist, one-layer hexagonal matching algorithm yielded better motion compensation results in terms of the PSNR and entropy due to the small search region associated with the second-layer mesh refinement procedure.

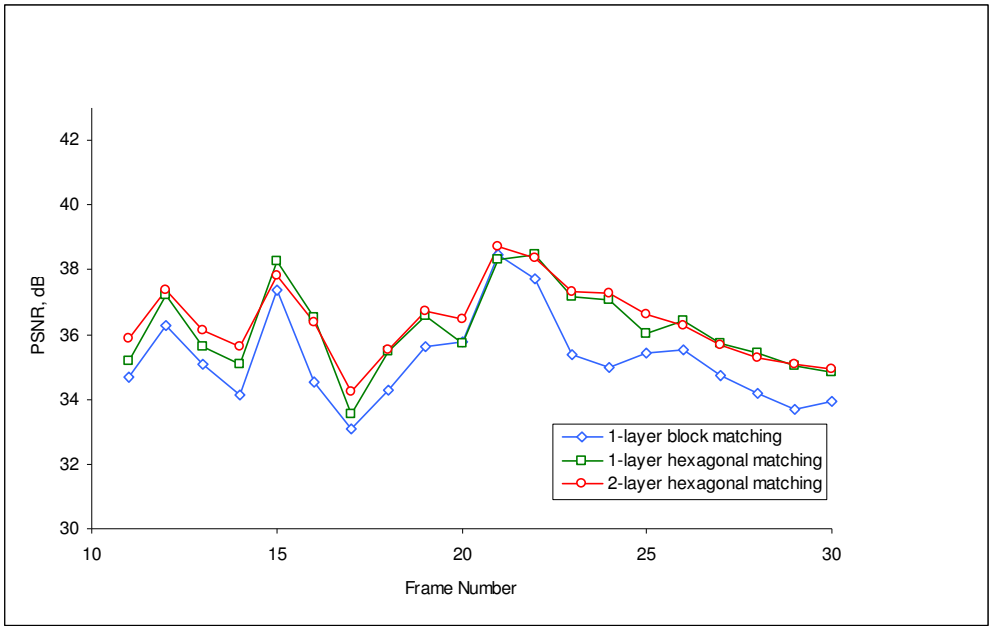


Figure 4.5 – Comparison of PSNR Values for Frames 11 to 30 of "Suzie" Sequence

Figure 4.6 – Comparison of Entropy Values for Frames 11 to 30 of "Suzie" Sequence

Table 4.5 – Average PSNR & Entropy Values for Frames 11 to 30 of "Suzie" Sequence

	1-Layer Block Matching	1-Layer Hexagonal Matching	2-Layer Hexagonal Matching
PSNR (dB)	35,25	36,19	36,39
Entropy (bits/symbol)	2,90	2,83	2,82

Figure 4.7 – Comparison of PSNR Values for Frames 41 to 60 of "Suzie" Sequence

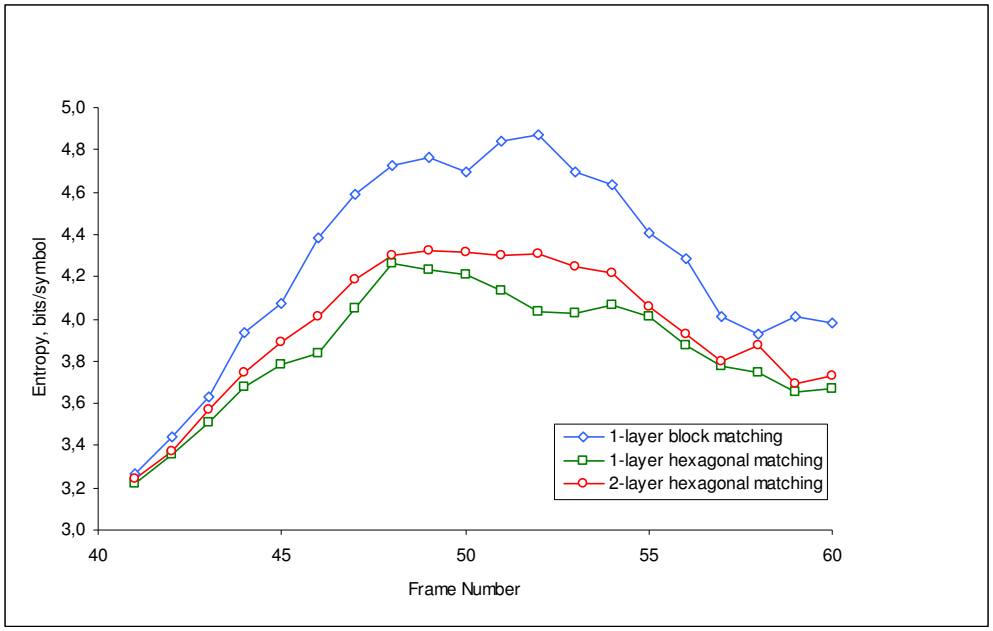


Figure 4.8 – Comparison of Entropy Values for Frames 41 to 60 of "Suzie" Sequence

Table 4.6 – Average PSNR & Entropy Values for Frames 41 to 60 of "Suzie" Sequence

	1-Layer Block Matching	1-Layer Hexagonal Matching	2-Layer Hexagonal Matching
PSNR (dB)	26,09	29,07	28,50
Entropy (bits/symbol)	4,26	3,86	3,96

4.2.2.2. Comparison of Content-Based and Regular Mesh-Based Algorithms

The second part of the experiment was performed to compare the one-layer regular mesh-based algorithm (the hexagonal matching algorithm) with a one-layer content-based algorithm. This algorithm is based on the content-based

interior node selection and triangulation approaches that are described in Chapter 3.

The performance of the content-based algorithm depends on several parameters associated with the interior node selection algorithm. One of the parameters is *minimum allowed distance between any two node points*. This distance affects the distribution of node points. A small value of the parameter is preferred since a large value may limit the number of node points and reduce the dependency on the image content. The distance was selected as 10 pixels in the tests.

Desired number of node points is another parameter. This parameter controls the number of nodes and triangles in the generated mesh. In the tests, this parameter was tuned for each frame to obtain the same number of triangles as the regular mesh used in one-layer hexagonal matching.

In the experiment, the content-based algorithm performed better than the one-layer hexagonal matching algorithm. This was as expected since the content-based mesh is designed according to the image content while the one-layer regular mesh is designed independently of the underlying image. As a result, the content-based algorithm yielded a more accurate representation of the motion field.

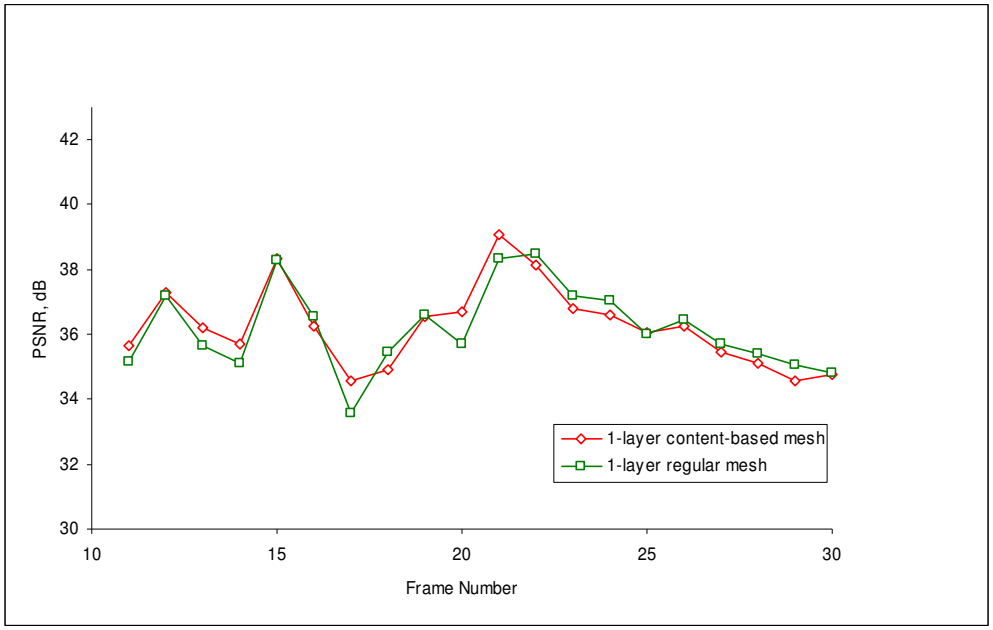


Figure 4.9 – Comparison of PSNR Values for Frames 11 to 30 of "Suzie" Sequence

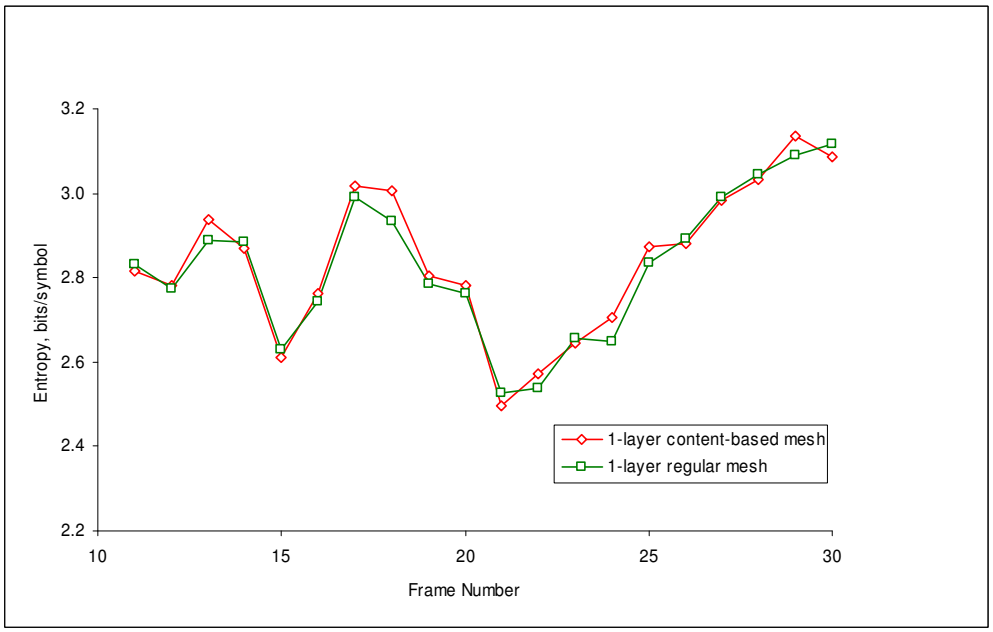


Figure 4.10 – Comparison of Entropy Values for Frames 11 to 30 of "Suzie" Sequence

Table 4.7 – Average PSNR & Entropy Values for Frames 11 to 30 of "Suzie" Sequence

	1-Layer Content-Based Mesh	1-Layer Regular Mesh
PSNR (dB)	36,25	36,19
Entropy (bits/symbol)	2,84	2,83

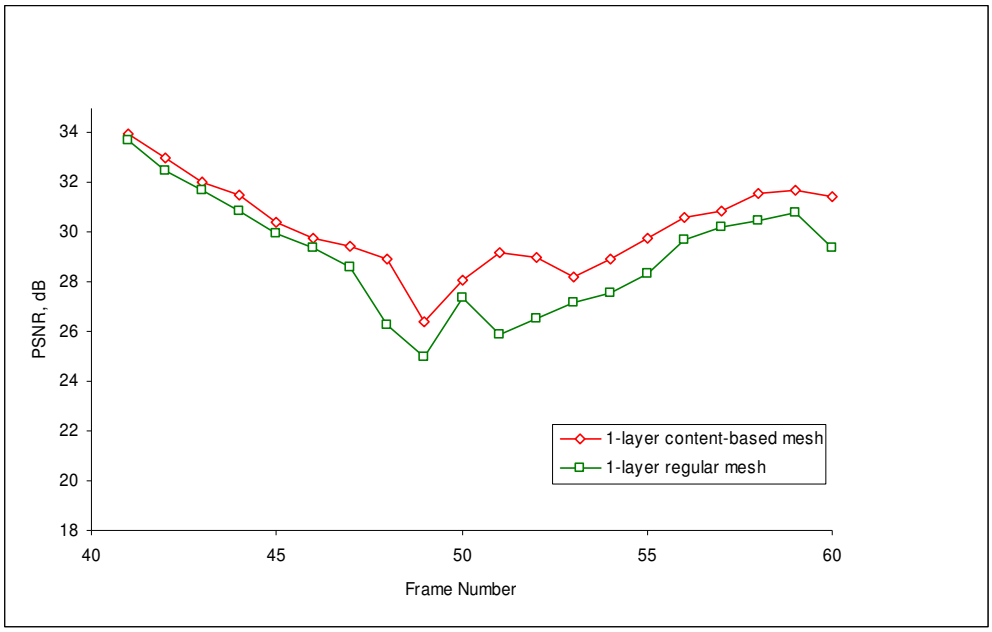


Figure 4.11 – Comparison of PSNR Values for Frames 41 to 60 of "Suzie" Sequence

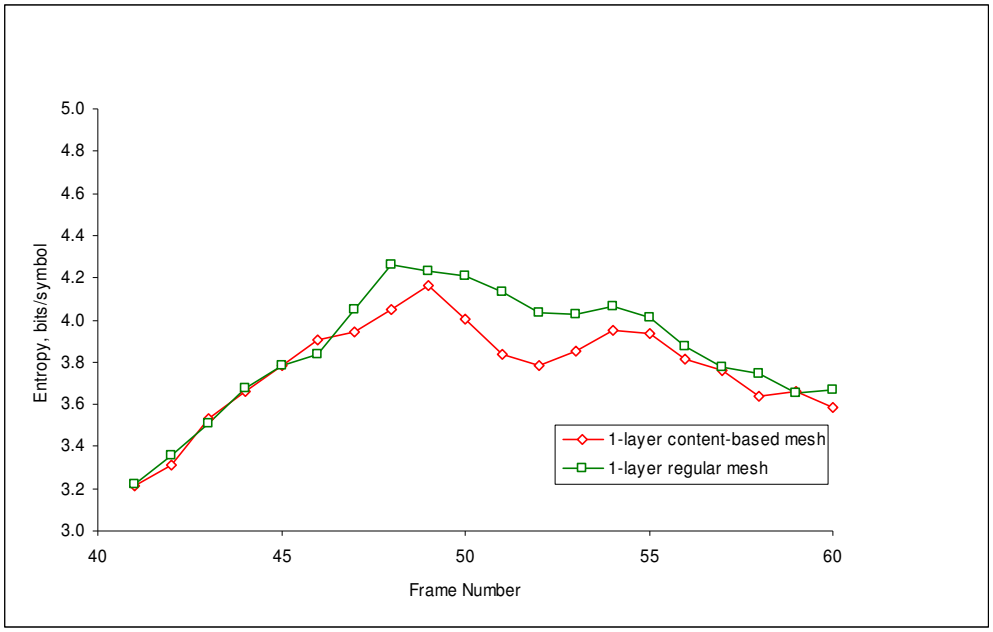


Figure 4.12 – Comparison of Entropy Values for Frames 41 to 60 of "Suzie" Sequence

Table 4.8 – Average PSNR & Entropy Values for Frames 41 to 60 of "Suzie" Sequence

	1-Layer Content-Based Mesh	1-Layer Hexagonal Matching
PSNR (dB)	30,25	29,07
Entropy (bits/symbol)	3,77	3,86

4.2.3. Experiment 3: Video Object Manipulation for "Bream" Sequence

In this experiment, application of 2-D mesh-based video representation to video object manipulation was performed. Video object manipulation was realized by an augmented reality and a spatial interpolation application on the "Bream" sequence.

In the two applications, the content-based irregular mesh approach, as described in the previous chapter, was employed. An initial 2-D content-based mesh was designed for the first appearance of the video object (the moving fish in the "Bream" sequence) which is assumed to be planar as seen in Figure 4.13 (a). Then, this mesh was tracked in the following frames to obtain the animation parameters for the object (Figure 4.13 (b)). Motion estimation at the nodes of the mesh was performed by the polygonal matching algorithm as described in Chapter 3.

Connectivity of the mesh was preserved by using some constraints on the displacement of the mesh nodes such as the *minimum allowed patch size*, etc. While enabling continuous tracking of the initial mesh structure, these constraints caused some of the nodes to converge to incorrect positions (Figure 4.13 (b)). The future work on mesh tracking may include the update of the mesh structure by node addition and deletion particularly in the presence of occluded or discovered areas.

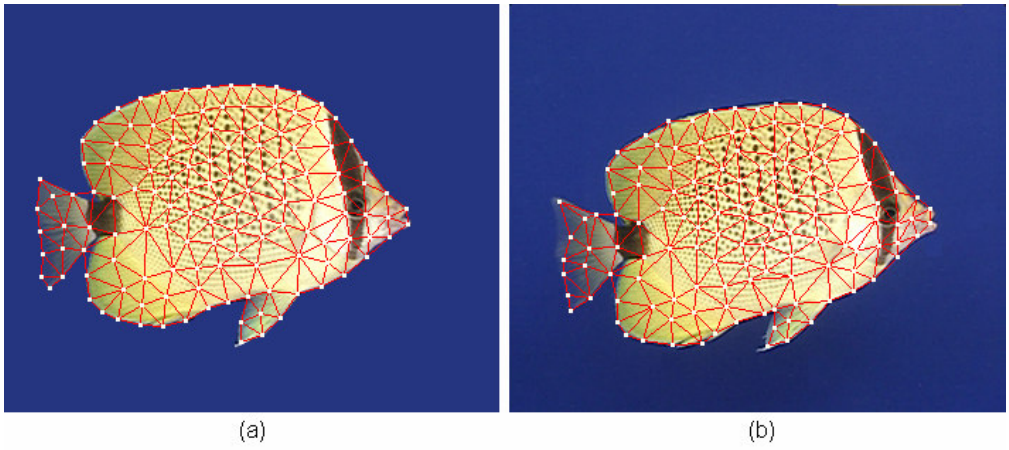


Figure 4.13 – Mesh Tracking on "Bream" Sequence. (a) Initial Mesh in Frame 1, (b) Tracked Mesh in Frame 25.

4.2.3.1. The Augmented Reality Application

Augmented reality, as described in the previous chapter, can be realized by merging synthetic images with real moving images. In this application, the text

"Fish" was used as the synthetic augmentation image as illustrated in Figure 4.14. The animation parameters obtained by motion tracking of the original fish object were applied to synthesize a merged sequence. In Figure 4.14 (b), a synthesized composite object is illustrated.

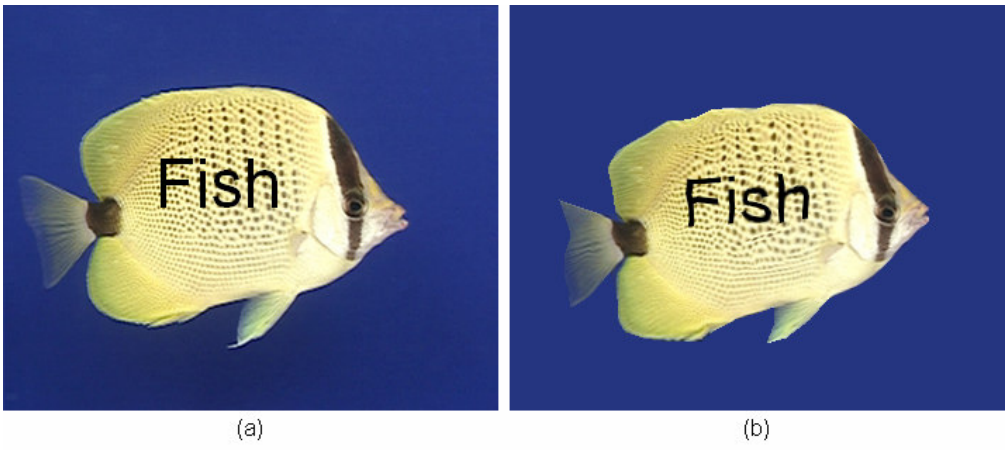


Figure 4.14 – Augmentation on "Bream" Sequence. (a) Original Composite Object in Frame 1, (b) Synthesized Composite Object in Frame 25.

4.2.3.2. The Spatial Interpolation Application

Mesh-based spatial interpolation can be used for performing zooming in and out on video objects or replacing a single object with several different sized objects through the video sequence. In the mesh-based representation, changing the size of a video object requires scaling of the 2-D object mesh and applying texture mapping on the new scaled mesh. In this application, the original fish object was replaced with its three different sized copies. All of the three objects had the same motion as the original object.

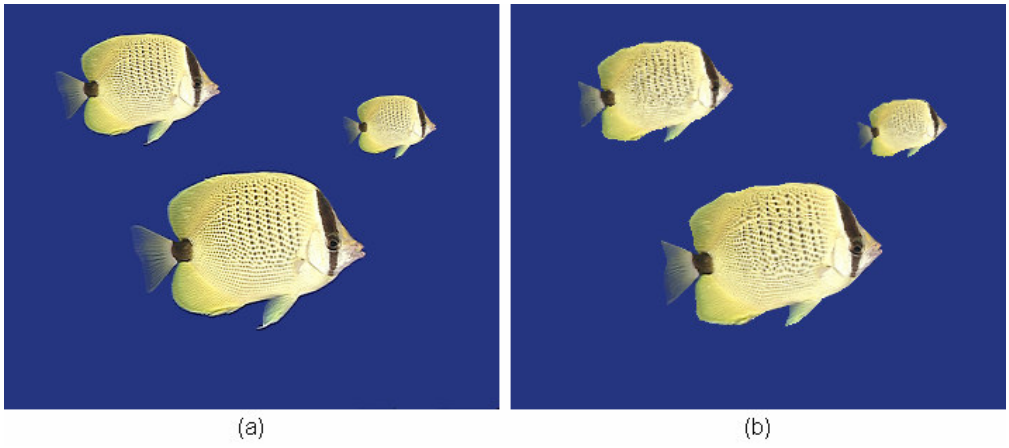


Figure 4.15 – Spatial Interpolation on "Bream" Sequence. (a) Original Objects in Frame 1, (b) Synthesized Objects in Frame 25.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Block-based and mesh-based models are extensively employed in 2-D motion estimation and compensation which has great importance for video compression applications. In this thesis, several block-based and mesh-based motion estimation methods have been investigated. The advantages and drawbacks of these methods have been stated and their performances have been evaluated by a number of experiments on both frame-based and object-based video representations. In addition, application of a 2-D mesh-based model to video object manipulation has been performed.

Block-based motion estimation has been realized by the use of the block matching algorithm. The algorithm is fast and simple to implement. However, the algorithm performed worse than the mesh-based algorithms in terms of the PSNR and entropy in both frame-based and object-based experiments. The reason is that the block matching algorithm adopts a 2-D translational motion model which suffers from representing complex motion.

Mesh-based models provide a better representation of complex real world motion than block-based models. A 2-D mesh-based model produces a continuous motion field due to the constraints on the movement of mesh nodes. Hence, such a model is most suitable in the presence of mild deformations. This kind of model fails to represent motion discontinuities. High computational complexity is another drawback of mesh-based models.

In the frame-based experiment, mesh-based motion estimation was realized by both one-layer and two-layer hexagonal matching algorithms. The two layer hexagonal matching algorithm was implemented by using the progressive motion estimation technique. The technique showed improvement over the one-layer

hexagonal matching algorithm particularly in the presence of smooth transitions and local motion which is represented better by the smaller triangles of the second layer. The one-layer hexagonal matching algorithm performed better in the occlusion regions due to the small search range used in the second layer of the two-layer hexagonal matching algorithm.

In the object-based experiment, the results were similar to the frame-based experiment but this time all of the algorithms produced low PSNR values. The reason is that the segmented object had higher motion activities than the background. In this experiment, a content-based algorithm was also compared with the single-layer hexagonal matching algorithm. The content-based algorithm yielded more accurate motion compensation results since it represents the object boundaries and non-stationary regions of the image better.

2-D mesh-based models also enable the tracking of video objects. Two video object manipulation applications utilizing the object tracking by 2-D dynamic meshes were performed in this thesis. In the first application, augmented reality was realized by merging a synthetic image with a natural video object. The second application involved the replacement of the same video object with its three different sized copies.

The future work on mesh-based motion estimation covers the application of the presented algorithms to video coding. Research can be conducted to further improve the coding efficiency of the algorithms. Moreover, reduction of computational complexity of the mesh-based algorithms can be examined for real-time applications. Lastly, 3-D mesh-based modeling can be investigated for the manipulation of video objects.

REFERENCES

[1] Y. Nakaya and H. Harashima, “Motion Compensation Based on Spatial Transformations”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 4, No. 3, pp. 339-356, June 1994.

[2] H. Park, A. C. Yu, and G. R. Martin, “Progressive Mesh-Based Motion Estimation Using Partial Refinement”, Proceedings of International Workshop on Very Low Bit-rate Video (VLBV) 2005, 4 pp., September 2005.

[3] A. M. Tekalp, P. J. L. Van Beek, C. Toklu, and B. Günsel, “Two-Dimensional Mesh-Based Visual-Object Representation for Interactive Synthetic/Natural Digital Video”, Proceedings of the IEEE, Vol. 86, No. 6, pp. 1029-1051, June 1998.

[4] M. Servais, T. Vlachos, and T. Davies, “Affine Motion Compensation Using a Content-Based Mesh”, IEE Proceedings on Vision, Image and Signal Processing, Vol. 152, No. 4, pp. 415-423, August 2005.

[5] A. M. Tekalp, Digital Video Processing. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[6] Y. Wang, J. Ostermann, and Y. Q. Zhang, Digital Video Processing and Communications. Prentice-Hall 2001.

[7] Y. Altunbasak and A. M. Tekalp, “Occlusion-Adaptive, Content-Based Mesh Design and Forward Tracking”, IEEE Transactions on Image Processing, Vol. 6, No. 9, pp. 1270-1280, September 1997.

[8] G. Al-Regib, Y. Altunbasak, and R. M. Mersereau, “Hierarchical Motion Estimation with Content-Based Meshes”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 10, pp. 1000-1005, October 2003.

[9] Y. Altunbasak, A. M. Tekalp, and G. Bozdagi, “Two-Dimensional Object-Based Coding Using a Content-Based Mesh and Affine Motion Parameterization”, IEEE International Conference on Image Processing, Vol. 2, pp. 394-397, October 1995.

[10] C. Toklu, A. T. Erdem, M. I. Sezan, and A. M. Tekalp, “2-D Mesh Tracking for Synthetic Transfiguration”, IEEE International Conference on Image Processing, Vol. 3, pp. 536-539, October 1995.

[11] J. Fan, X. Zhu, and L. Wu, “Automatic Model-Based Semantic Object Extraction Algorithm”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 10, pp. 1073-1084, October 2001.

[12] W. Yang, J. Liu, and H. Chen, “Automatic Extraction of Moving Objects in Video Sequences Based on Spatio-Temporal Information”, 31th Annual Conference of the IEEE Industrial Electronics Society, pp. 369-372, November 2005.

[13] J. R. Shewchuk, “Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator”, Proceedings of the 1st Workshop on Applied Computational Geometry, pp. 124–133, May 1996.

[14] J. R. Jain and A. K. Jain, “Displacement Measurement and Its Application in Interframe Image Coding”, IEEE Transactions on Communications, Vol. COM-29, No. 12, pp. 1799-1808, December 1981.

[15] H. G. Musmann, P. Pirsch, and H. J. Grallert, “Advances in Picture Coding”, Proceedings of the IEEE, Vol. 73, No. 4, pp. 523-548, April 1985.

[16] R. Srinivasan and K. R. Rao, “Predictive Coding Based on Efficient Motion Estimation”, IEEE Transactions on Communications, Vol. COM-33, No. 8, pp. 888-896, August 1985.

[17] C. Toklu, A. Erdem, M. Sezan, and A. M. Tekalp, “Tracking Motion and Intensity Variations Using Hierarchical 2-D Mesh Modelling for Synthetic Object Transfiguration”, Graphical Models and Image Processing, Vol. 58, No. 6, pp. 553-573, November 1996.

[18] P. V. Beek, A. M. Tekalp, N. Zhuang, I. Celasun, and M. Xia, “Hierarchical 2D Mesh Representation, Tracking, and Compression for Object-Based Video”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, pp. 353-369, March 1999.

[19] “MPEG-4 Overview”, ISO/IEC JTC1/SC29/WG11 N4668, March 2002.

[20] I. Celasun and A. M. Tekalp “Optimal 2-D Hierarchical Content-Based Mesh Design and Update for Object-Based Video”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, No. 7, pp. 1135-1153, October 2000.

[21] C. Toklu, A. M. Tekalp, A. T. Erdem, and M. I. Sezan, “2-D Mesh-Based Tracking of Deformable Objects with Occlusion”, IEEE International Conference on Image Processing, Vol. 1, pp. 933-936, September 1996.

[22] A. Mahboubi, J. Benois-Pineau, and D. Barba, “Tracking of Hierarchical Active Meshes for Object-Based Manipulation of Video Content”, IEEE TENCON 2000. Proceedings, Vol. 1, pp. 53-58, September 2000.

[23] A. C. Yu, H. Park, and G. R. Martin, “Fast Mesh-Based Motion Estimation Employing an Embedded Block Model”, IEEE International Symposium on Circuits and Systems 2006, pp. 4703-4706, May 2006.

[24] A. Nosratinia, “New Kernels for Fast Mesh-Based Motion Estimation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 1, pp. 40-51, January 2001.

[25] C. Toklu, A. M. Tekalp, and A. T. Erdem, “Semi-Automatic Video Object Segmentation in the Presence of Occlusion”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, No. 4, pp. 624-629, June 2000.