İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

NEW FRAME RATE UP-CONVERSION METHOD BASED ON FOREGROUND/BACKGROUND SEGMENTATION

M.Sc. Thesis by
Mehmet Mutlu ÇEKİÇ

Department : Computer Engineering

Programme : Computer Engineering

Thesis Supervisor: Assoc. Prof. Dr. Uluğ BAYAZIT

APRIL 2011

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

NEW FRAME RATE UP-CONVERSION METHOD BASED ON FOREGROUND/BACKGROUND SEGMENTATION

M.Sc. Thesis by
Mehmet Mutlu ÇEKİÇ
(504071548)

Date of submission : 20 April 2011
Date of defence examination: 01 April 2011

Supervisor (Chairman) : Assoc. Prof. Dr. Uluğ Bayazıt (ITU)
Members of the Examining Committee : Asst. Prof. Dr. Mustafa Kamaşak (ITU)
Asst. Prof. Dr. Hasan Fehmi Ateş (Işık Universiy)

APRIL 2011

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÖN YÜZ/ARKA YÜZ SEGMENTASYONUNA BAĞLI YENİ ÇERÇEVE HIZ ARTIRIMI

YÜKSEK LİSANS TEZİ

Mehmet Mutlu ÇEKİÇ
(504071548)

Tezin Enstitüye Verildiği Tarih : 20 Nisan 2011
Tezin Savunulduğu Tarih : 01 Nisan 2011

Tez Danışmanı : Doç. Dr. Uluğ BAYAZIT (İTÜ)
Diğer Jüri Üyeleri : Yrd. Doç. Dr. Mustafa KAMAŞAK (İTÜ)
Yrd. Doç. Dr. Hasan Fehmi ATEŞ (Işık Üniversitesi)

NİSAN 2011

FOREWORD

I would like to thank my supervisor, Assoc. Prof. Dr. Uluğ Bayazıt for his impressive help. He always gave me motivation of this work and allowed time for discussing every development of this thesis. Having the opportunity to work with him was rewarding and fullfilling and I was glad to work with him. I also want to thank Burak Çizmeci and Assist. Prof. Hasan Fehmi Ateş who helped me to start this work and shared their experiences and knowledge on this area. I want to thank my graduate school colleague Birkan Polatoğlu who always helped me all through the years full of class work and exams. I also want to thank my sister and also my housemate Yurdanur Çekiç for her help and giving motivation of work. Finally, I want to thank my mother Mahi Çekiç and my father Rüştü Çekiç. They always give their support for my educational life. I feel lucky since I have these parents.

Nisan 2011

Mehmet Mutlu ÇEKİÇ

Computer Engineer

TABLE OF CONTENTS

Page

ABBREVIATIONS

FRUC	**:** Frame Rate Up-Conversion
MC	**:** Motion Compensation
PSNR	**:** Peak Signal to Noise Ratio
HD	**:** High Definition
FPS	**:** Frame per Second
MCFRUC	**:** Motion Compensated Frame Rate Up-Conversion
ME	**:** Motion Estimation
MV	**:** Motion Vector
BM	**:** Block Matching
SAD	**:** Sum of Absolute Difference
GMM	**:** Gaussian Mixture Model
SSE	**:** Sum of Square Error
3-D	**:** 3 Dimensions
3DRS	**:** 3-D Recursive Search block matching algorithm
TSS	**:** Three Step Search
HBM	**:** Hierarchical Block Matching
BG	**:** Background
FG	**:** Foreground
OBMC	**:** Overlapped Block Motion Compensation
MSE	**:** Mean Squared Error
SSIM	**:** Structural Similarity
dB	**:** Decibel
QCIF	**:** Common Intermediate Format
CIF	**:** Quarter Common Intermediate Format

LIST OF TABLES

Page

LIST OF FIGURES

Page

NEW FRAME RATE UP-CONVERSION METHOD BASED ON FOREGROUND/BACKGROUND SEGMENTATION

SUMMARY

Frame rate up-conversion (FRUC) increases the quality of a video by increasing its temporal frequency. Motion compensated (MC) and non-motion compensated frame rate up conversion techniques make up the two main classes of techniques used in this area. Halo artifacts and jaggy edges cause the quality of video to be reduced both subjectively and objectively in these techniques. These are the main problem of FRUC techniques. In this work, we introduce a new method of motion compensated FRUC that uses Foreground/Background segmentation to address the occlusion problem. It scales motion vectors of existent frames to obtain motion vectors of interpolated frames. The gap pixels for which no motion vectors have been assigned are interpolated from their neighbors based on their foreground or background identity. A background subtraction algorithm was integrated to this work to segment the frames as background and foreground. The current work improves the quality of video and obtains better PSNR results than the proposed methods of [9], [13] and [14].

ÖN YÜZ/ARKA YÜZ SEGMENTASYONUNA BAĞLI YENİ ÇERÇEVE HIZ ARTIRIMI

ÖZET

Video çerçeve hız artırımı bir saniyeye düşen çerçeve sayısını artırarak bir videonun kalitesini artırmayı sağlar. Hareket dengeleyici ve hareket dengeleyici olmayan çerçeve hız artırımı bu alanı oluşturan başlıca iki tekniktir. Her iki teknikte de yapaylıklar ve tırtıklı kenarlar öznel ve nesnel olarak videonun kalitesinin düşmesine yol açar. Bu iki tip bozunum çerçeve hız artırımının iki ana problemidir. Bu problemleri ön plan ve arka plan segmentasyonunu kullanarak çözen yeni bir hareket dengeleyici çerçeve hız artırımı tekniğini bu çalışmada sunuyoruz. Bu çalışma elimizdeki çerçevelerin hız vektörlerini ölçekleyerek aradeğerlenmiş çerçevenin hareket vektörlerini elde eder. Hiç bir vektör atanmamış olan boş pikseller komşu piksellerin ön yüz/arka yüz etiketlerine bağlı olarak aradeğerlenme yöntemi ile oluşturulurlar. Çerçeveleri ön yüz ve arka yüz diye ayırabilmek için bir arka yüz çıkarıcı algoritması bu çalışmaya entegre edilmiştir. Önerilen yöntem ile videonun kalitesi artırıldığı gibi [9], [13] ve [14] yayınlarında önerilen metotlardan çok daha iyi tepe sinyal gürültü oranı elde edilmiştir.

1. INTRODUCTION

Advances in display technologies of televisions or monitors take place almost every day. Recently, the popularity of high definition (HD) TV's has made the 100Hz-120Hz video display technology a desirable feature. Due to limited bandwidth of the transmission network (wireless, cable etc.), video is generally captured and encoded at 25Hz progressive, 50Hz interlaced for PAL before the broadcast. In order to convey a sense of smoothness of motion, the temporal resolution is increased at receiver side. For instance, a received video with 25 frames per second (fps) can be converted into a video with 50 fps or 100 fps. This technique is called frame rate up conversion (FRUC).

1.1 Frame Rate Up-Conversion

There are a lot of methods used for FRUC in the past and new methods are still going on to be introduced. Recently, Nguyen has introduced new FRUC methods such as [20], [21] and [22]. He proposed correlation based vector processing method. Unreliable motion vectors are detected and corrected by this method [20]. Also, an adaptive motion compensation method is proposed for the occlusion areas based on the analysis of their surrounding motion distribution by this method. Alatan and Turetken introduced a novel method which is based on the segmented motion layers with planar perspective motion models [23]. Moreover, a method using bilateral motion estimation and adaptive motion compensation is introduced via [25], an adaptive FRUC based on motion classification is proposed by [9], a new FRUC method with edge weighted motion estimation and trilateral interpolation is introduced by [24].

There are two types of FRUC method used in the video processing area. These are non-motion compensated and motion compensated video frame rate up conversion (MCFRUC). Non-motion compensated FRUC does not consider the motion of the objects. It is implemented as either frame repetition or linear interpolation of the frames [10]. Due to the lack of adequate computational resources, this technique was

popular in the past. On the down side, it causes artifacts in the video like motion blur or motion judder.

In the last ten years, motion compensated video frame rate up conversion methods with higher complexity could be supported due to the developments in the microchip technology for receiver end processors. When compared to the non-motion compensated methods, these methods efficiently reduce the occurrence of the aforementioned artifacts resulting in higher interpolated frame quality. Motion estimation (ME) and motion compensation (MC) are the two main stages of motion compensated frame rate up conversion.

In the ME stage, motion vectors (MV) of the objects are calculated according to two sequential frames that are available. These motion vectors are then used for the reconstruction of a new (interpolated) frame between these two sequential frames. Generally, the block matching (BM) method is used for motion estimation. This method divides a frame into pixel blocks and displacement estimation is performed by comparing this block with displaced blocks in the other frame. Similarity of blocks can be decided according to sum of absolute difference (SAD) measure. The displaced block with the minimum SAD is the most similar one.

Block matching yields acceptable video coding performance. However, it does not yield true motion vectors which are very important for reconstructing higher quality interpolated frames. 3-D Recursive Search Block Matching introduced by De Haan addresses this deficiency of block matching [5]. Some other techniques like ME for overlapped motion vectors [6], motion vector classification [9], correlation of motion vectors [7] and adaptive true motion estimation [8] are also used to overcome ME problems in obtaining true motion vectors.

In the MC stage, a new frame between two successive frames is reconstructed via estimated MV's. While this method is being applied some problems can be encountered like halo artifacts or jaggy edges on occlusion areas. Occlusion area contains the covered/uncovered parts of the video. Handling occlusion artifacts is not a trivial problem and many solutions have been proposed to decrease these artifacts. The most popular one of these is based on using occlusion map as suggested by [3] and [4]. Another approach is constructing color segmentation maps obtained by Markovian approach [2]. Ince [1] proposed a new method called geometry based estimation which overcomes occlusion artifacts by using geometric mismatches.

1.2 Foreground/Background Subtraction

Identifying moving objects is a very popular problem in the computer vision area and the most popular solution is background subtraction. This technique finds stationary pixels (background pixels) and then it subtracts them from the whole frame. Thereby, pixels of moving objects (foreground) are determined. One of the methods used for background subtraction is Zivkovic's algorithm ([11]) that employs Gaussian mixture model (GMM) for estimating probability density. Another approach is based on color segmentation or background modeling. This method proposes using color information obtained from background subtraction and shadow detection to improve object segmentation and background update [12].

In this thesis, we propose a method for enhancing the quality of the interpolated frame by using foreground data obtained from the background subtraction method. Gap pixels, to which no motion vector has been assigned are typically encountered while reconstructing the interpolated frame. With this technique, we construct gap pixels successfully for block based estimated motion vectors. Next section describes the motion estimation methods used for FRUC. In section 3, we give a brief information about segmentation and section 4 presents the proposed algorithm. Section 5 demonstrates test results of the proposed algorithm with many test sequences. Finally, last section contains conclusion of this work.

2. MOTION ESTIMATION

Motion estimation is used to extract motion information of the video sequence. In this technique, motion information is denoted generally by a motion vector $\vec{V}(x, y)$. It can be said that ME is the main part of the frame rate up conversion, because new frames are reconstructed based on motion estimation information. So, if a poor motion estimator is used in the FRUC system, this results in poorly interpolated frames. ME forms the core of FRUC. It is also used in some other applications such as video compression, motion tracking, video stabilization etc. Some of the variants of ME used for FRUC are listed below. They are:

- *Pel-Recursive Techniques*: A motion vector is derived for each pixel.
- *Optical Flow Techniques*: Motion vectors are determined by the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and the scene.
- *Phase Plane Correlation Techniques*: Motion vectors are determined according to correlation between current frame and reference frame.
- *Block Matching Techniques*: Since we have used this technique in our thesis, detailed information is presented below for this technique.

2.1 The Block Matching Algorithm

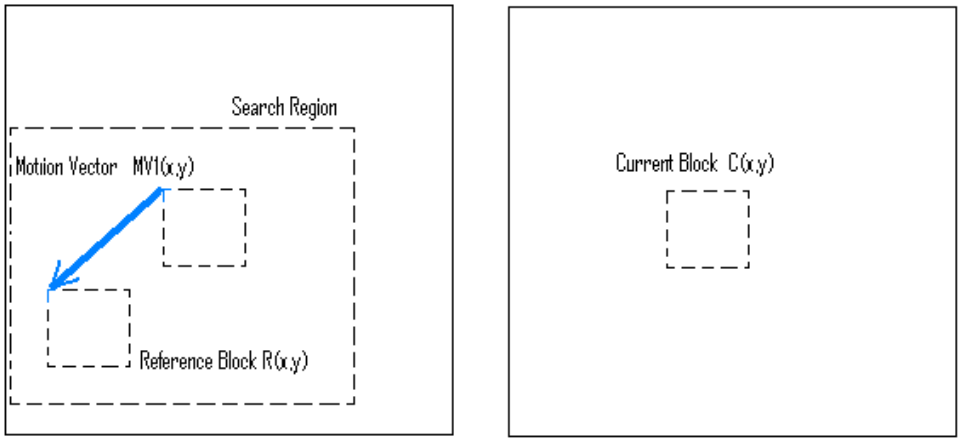


Figure 2.1 : Reference and Current Frame respectively

Block matching algorithm is the most widely used and popular ME technique. Instead of working on individual pixels, it calculates the motion vectors of block of pixels. Using blocks helps to reduce computational requirement.

As shown in Figure 2.1, the current and reference frame is divided into blocks and current block is searched for a best matching block in the reference frame. When best matching block is found, motion vector of the current block is determined. As it is seen, this vector is derived according to displacement from the current block's place in reference frame, and thus MV1(x,y) is obtained. There are several matching criterias minimized to yield a best match. The most widely used are sum of square error (SSE) and sum of absolute difference (SAD). They are obtained by the following equations. x and y denotes coordinates of the blocks.

$$\text{SSE} = \sum_{x=1}^{x=N} \sum_{y=1}^{y=N} (C(x,y) - R(x,y))^2 \quad \textbf{(2.1)}$$

$$\text{SAD} = \sum_{x=1}^{x=N} \sum_{y=1}^{y=N} \left| C(x,y) - R(x,y) \right| \quad \textbf{(2.2)}$$

SAD is the most widely used matching criterion because its computational requirement is lower than SSE's.

Block searching is not applied to the whole reference frame. A search region is defined according to motion in the video sequence and available computing resources. Generally, search region does not have a shape of square, because most of the video sequences make panning motions. So many applications use a wide search region.

2.1.1 Full Search Block Matching

This method has very basic and easy logic. It compares every possible block in the search region with current block; therefore it finds the best motion vector of the current block. On the down side, this algorithm requires many computations. Hence, this is not a popular method used in video processing area. Some other techniques that do not require so much computation are introduced below.

2.1.2 Three Step Search (TSS)

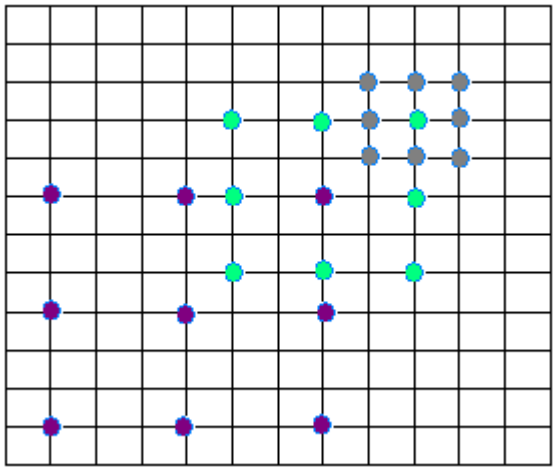


Figure 2.2 : Three Step Search

This is a greedy algorithm used for ME. In three step search algorithm, nine candidates are processed at the first iteration. The candidates are placed around the current block. As it is seen, each step narrows the search region. When, best matching candidate is found next search region is determined by taking this candidate as center. The iterations continue as long as the step size is not equal to one pixel. Thus, final candidate that is found in the last iteration becomes the best matching candidate and motion vector of the current block is determined as the displacement of this candidate. This algorithm requires very low computational complexity compared to full search while giving successful matching result but it does not guarantee that the found solution is optimum.

2.1.3 3-D Recursive Search

For video coding, the aim is to get a motion vector obtained from a best block match. The ideal ME algorithm is the one that yields motion vector estimates with small matching criterion and low computational cost. True estimated vectors which are obtained by low computational requirement can be thought as the best ME algorithm. On the other hand, motion vectors should represent true motion of the objects for frame rate up-conversion applications instead of providing best block match. Post processing algorithms can be used to derive more smooth motion vectors from motion vectors output by motion estimation algorithms such as block matching used in video coding.

3-D recursive search method is a motion estimation algorithm that directly yields true motion vectors. It was firstly proposed by De Haan [5]. The algorithm is based on two important assumptions. They are:

- Objects are larger than block size
- Objects have inertia

The first assumption advises that motion vectors of any neighboring block of the current block can be thought as a candidate of current block, but there is a gap in this assumption. Motion vectors of the neighboring block may not have been estimated yet. At this point, second assumption helps us to handle this problem. According to this assumption, motion vectors of previous frame are used for these blocks (neighboring blocks). So, it is assumed that some of the neighboring blocks are stationary. Also, it should be noticed that spatial and temporal neighboring motion vectors are used in this algorithm.

To start the algorithm a random vector needs to be used as a candidate. Figure 2.3 illustrates the sample candidates. Candidates are updated by a random vector that is randomly taken from a fixed set. This set can be like following.

$$RS = \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \begin{pmatrix} -2 \\ -2 \end{pmatrix}, \begin{pmatrix} 3 \\ 0 \end{pmatrix} \right\} \quad \textbf{(2.3)}$$

Therefore, new candidates are obtained by adding random vectors taken from RS to spatial neighbors. So, candidate set can be demonstrated as following.

$$CS = \begin{cases} \vec{v}(i-1, j-1, t), \vec{v}(i-1, j+1, t) \\ \vec{v}(i-2, j-2, t), \vec{v}(i-2, j+2, t) \\ \vec{v}(i-1, j-1, t) + \vec{V}_C, \vec{v}(i-1, j+1, t) + \vec{V}_C \end{cases} \quad \textbf{(2.4)}$$

Where $\vec{V}_C$ is the member of RS.

The algorithm goes on until finding the motion vector taken from candidate set that yields the smallest SAD as described in previous sections.

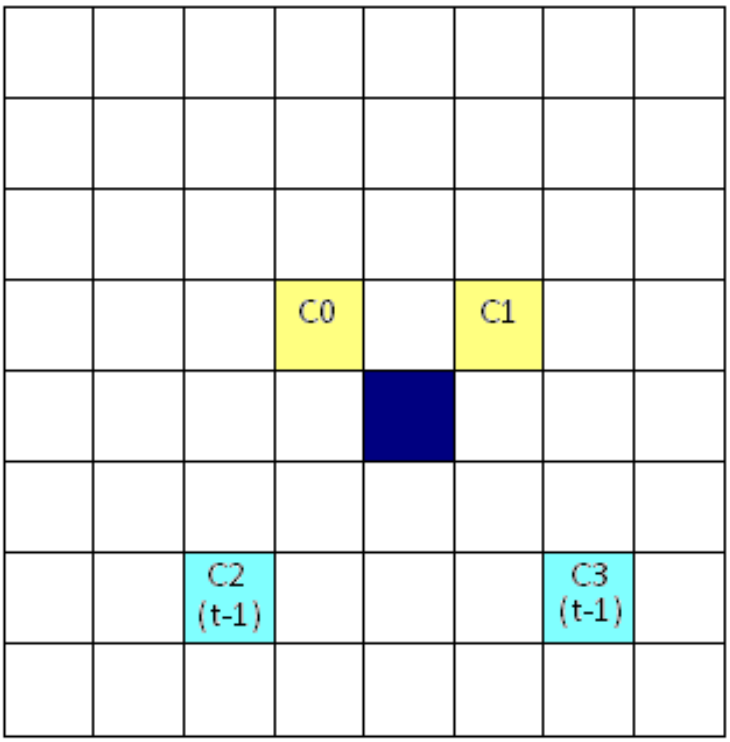


Figure 2.3 : 3D Recursive Search Candidates (t-1 represents previous frame)

2.1.4 Hierarchical Block Matching

Hierarchical Block Matching (HBM) is a different matching algorithm which provides very reliable motion vectors. This method is based on refining the motion at different resolutions [18]. In this method, a frame is taken and its resolution reduced gradually and thus a pyramid of frame in different resolutions is obtained. The original frame is located at the bottom of this pyramid and other members of the pyramid are obtained by down sampling the original frame. Let's say resolution of the original frame is NxN, therefore next resolution level is (N/2)x(N/2). In the same manner, search range of the algorithm is reduced too.

The algorithm starts from top level of the pyramid and compares all of the blocks in the search range as in 2.1.1. Since search range is reduced, it doesn't require so many computations. The motion vectors obtained in lowest level are scaled to the next level and they are considered as initial motion vectors of each block. The motion vectors are refined by carrying out a search region in the smaller search area around the initial motion vector, at the next level. Figure 2.4 shows how the algorithm works.

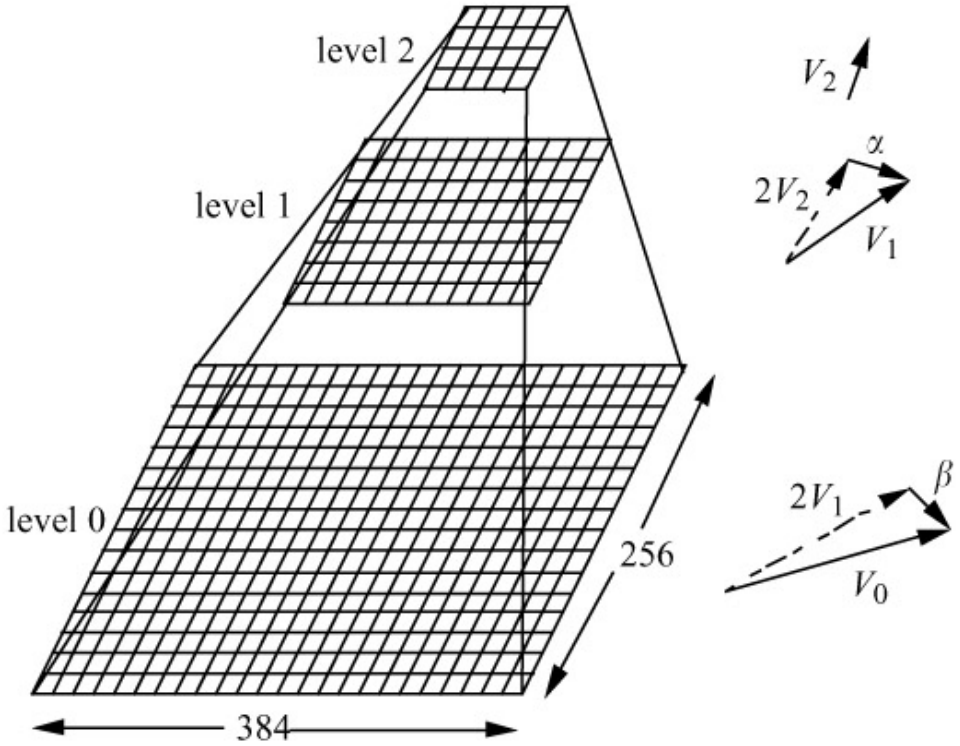


Figure 2.4 : Demonstration of Hierarchical Block Matching Method

2.1.5 Hexagonal Search Block Matching

Hexagonal Search Block Matching algorithm consists of seven candidate points with the center candidate surrounded by six candidates of the hexagon with up and down points being excluded [19]. Figure 2.5 shows that two horizontal candidates are away from center candidate with distance 2 and other four candidate are away from center with distance $\sqrt{5}$. Hexagonal search follows the pattern that center candidate is moved to any of the end candidates at the end of each step. In this pattern, there are always three new candidates and the other three candidates overlap previous candidates. This algorithm has three main steps as illustrated in Figure 2.5.

1) A large hexagon with the seven candidates is centered at (0,0) which is center of the predefined search window. If the center point gives the minimum SAD then step 3 is processed, otherwise step 2 is processed.
2) A new hexagon is placed with the new center at the candidate that gave the minimum SAD in the previous iteration. SAD calculation is again done for

three new candidates and then new center is defined. If center did not move minimum SAD then step 3 is applied, otherwise this step is repeated.

3) At this step, a small hexagon with center candidate surrounded with four candidates is formed. SAD of new four points are calculated and then the point which gives the minimum SAD is choosen as the best match. The final motion vector is the displacement vector of the final center from (0,0).

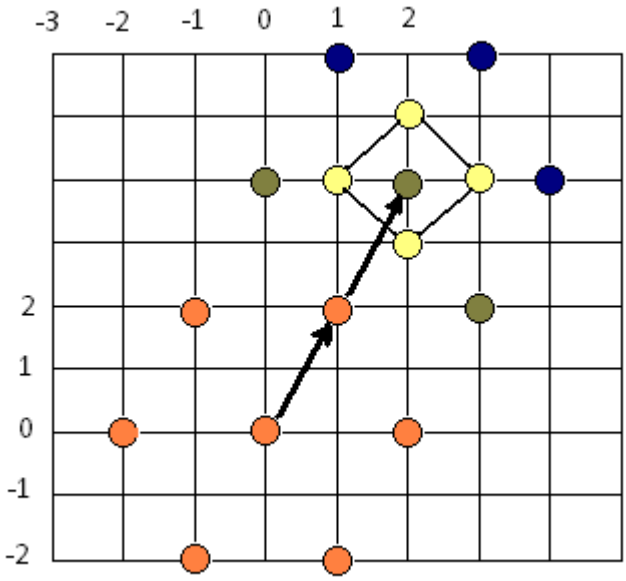


Figure 2.5 : Hexagonal Search Block Algorithm

3. FOREGROUND/BACKGROUND SEGMENTATION

Especially in computer vision area, foreground/background segmentation has a very important role for many applications such as visual surveillance, behavior analysis, etc. Generally, segmentation of foreground region can be achieved by subtracting learned background model from each frame. This method is also called background subtraction. However, this method doesn't generate successful results for some kind of dynamic scenes containing illumination change, shadow movement, and swaying objects. To overcome these issues, some methods have been proposed like adaptive background model [11]. In [15], a model that utilizes spatial and temporal statistics for detecting shadows and highlights is proposed. Furthermore, a probabilistic method based on edge measurement is introduced in [16].

We have used the adaptive background subtraction method in our work and integrated it to our algorithm as described in section 4. Following section explains briefly how this segmentation is done.

3.1 Segmentation Using Gaussian Mixture Model

In GMM segmentation model, every pixel is modeled by a mixture of K Gaussisans.

$$P(I_t) = \sum_{i=1}^{K} \omega_{i,t} \eta\left(I_t; \mu_{i,t}; \sum_{i,t}\right) \quad \textbf{(3.1)}$$

Where K is the component number and it is assumed that $\sum_{i,t} = {\sigma_{i,t}}^2 I$. I_t is the pixel's current value at time t.

In this technique, background is updated before foreground is detected. Following three steps show how background is updated.

1. If I_t matches component i, I_t is within standard deviations of $\mu_{i,t}$.

   $$\left(I_t - \mu_{i,t}\right) / \sigma_{i,t} > 2.5 \quad \textbf{(3.2)}$$

   Then the ith component is updated as following.

$$\omega_{i,t} = \omega_{i,t-1} \quad \textbf{(3.3)}$$

$$\mu_{i,t} = (1-\rho)\mu_{i,t-1} + \rho I_t \quad \textbf{(3.4)}$$

$$\sigma_{i,t}^{\ 2} = (1-\rho)\sigma_{i,t}^{\ 2} + \rho(I_t - \mu_{i,t})(I_t - \mu_{i,t}) \quad \textbf{(3.5)}$$

Where

$$\rho = \alpha \Pr(I_t | \mu_{i,t-1}, \sum_{i,t-1}) \quad \textbf{(3.6)}$$

2. Components which I_t don't match are updated by following.

$$\omega_{i,t} = (1-\alpha)\omega_{i,t-1} \quad \textbf{(3.7)}$$

$$\mu_{i,t} = \mu_{i,t-1} \quad \textbf{(3.8)}$$

$$\sigma_{i,t}^{\ 2} = \sigma_{i,t-1}^{\ 2} \quad \textbf{(3.9)}$$

3. If I_t does not match any component, then the least likely component is replaced with a new one which has $\mu_{i,t} = I_t$, $\sum_{i,t}$ large, and $\omega_{i,t}$ low.

After all of the updates are completed, the weights $\omega_{i,t}$ are renormalised.

Then foreground is detected as follows [26]. All components in the mixture are sorted into the order of decreasing $\omega_{i,t} / \left\| \sum_{i,t} \right\|$. So, higher importance gets placed on components with the most evidence and lowest variance, which are assumed to be background.

$$B = \arg\min_b \left(\frac{\sum_{i=1}^{b} \omega_{i,t}}{\sum_{i=1}^{K} \omega_{i,t}} > T \right) \quad \textbf{(3.10)}$$

Where T is a threshold. Therefore, the components 1…..B are assumed to be background. Thus, if I_t does not match one of these components, the pixel is marked as foreground. Finally, foreground pixels are segmented into regions using connected component labeling.

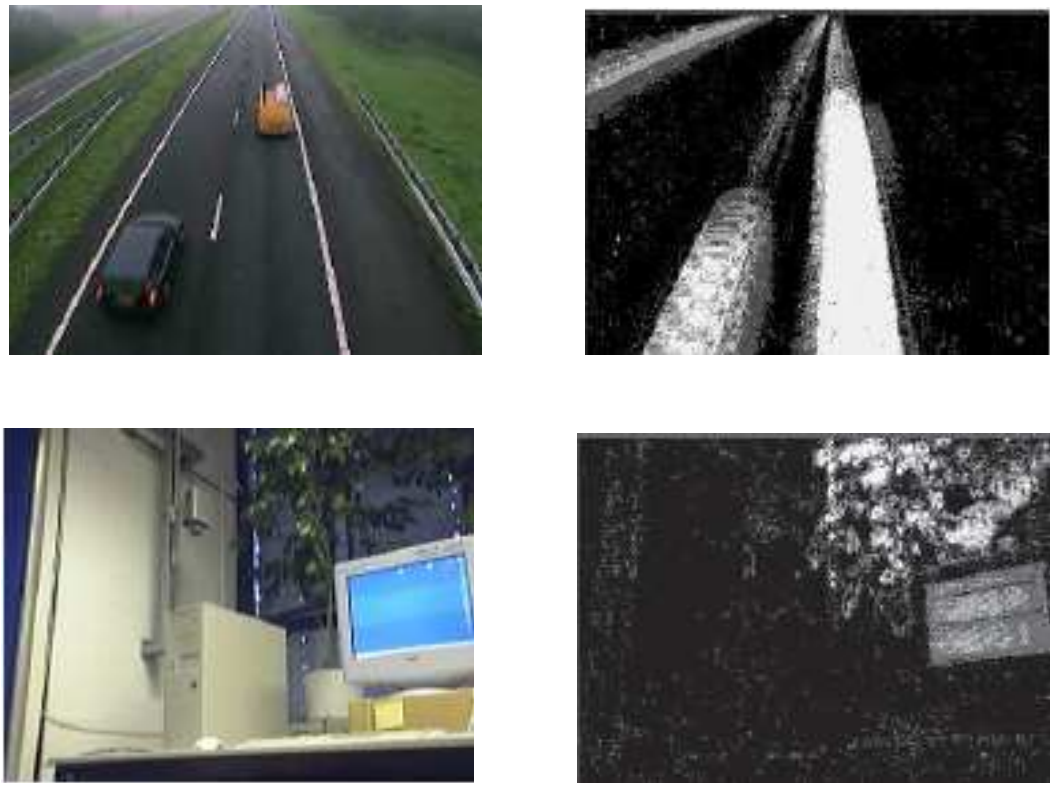


Figure 3.1 : Shows segmented pictures obtained by GMM [11]

4. MOTION COMPENSATION BASED ON FOREGROUND/BACKGROUND SEGMENTATION

Let subscript 0 denote the previous frame and subscript 1 denote the current frame of the two successive frames of a video sequence. The interpolated frame midway between (in time) these two frames is denoted by subscript ½. The notation $mv_{i,j}(x,y)$ denotes a motion vector at (x,y) that points from frame i to frame j. The initial motion field between frames 1 and 0 is obtained by using Hexagonal block matching algorithm.

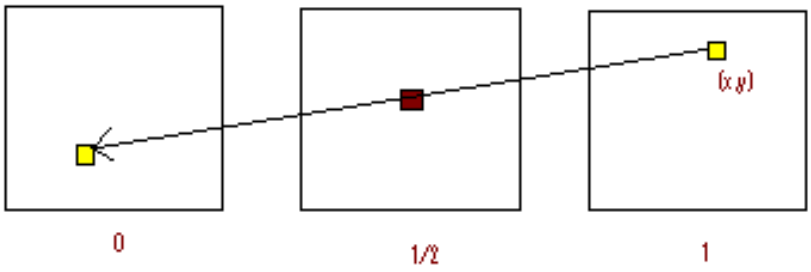


Figure 4.1 : Shows motion vector $mv_{1,0}(x,y)$

4.1 Background Subtraction

During the first stage of our algorithm, the foreground region of current frame is determined by the background subtraction method. Later, when the interpolated frame is constructed, this information, that indicates the positions of the moving object pixels, is employed.

Figure 4.2 : Foreground/Background Segmentation of Akiyo Sequence (Frame60)

4.2 Scaling Motion Vectors

The second stage is the scaling of the motion vectors from $mv_{1,0}(x,y)$ to $mv_{1,1/2}(x,y)$ for each pixel. Since the motion vector $mv_{1,1/2}(x,y)$ is obtained by scaling, it does not have full pixel accuracy. Let $\tilde{x}$ and $\tilde{y}$ be used to denote the coordinates of the half pixel accurate point in frame ½ which can be found via the following equations.

$$\tilde{x} = x - mv_{x,1,1/2}(x,y) \quad \textbf{(4.1)}$$

$$\tilde{y} = y - mv_{y,1,1/2}(x,y) \quad \textbf{(4.2)}$$

The pixel positions in frame ½ in a half pixel neighborhood around $\tilde{x}$ and $\tilde{y}$ are then determined. The coordinates of such a pixel satisfies

$$(|x'-\tilde{x}| \le 0.5)\, AND\, (|y'-\tilde{y}| \le 0.5) \quad \textbf{(4.3)}$$

Motion vectors of frame ½ can now be expressed by using the motion vectors from frame 1 to frame 1/2.

$$mv_{1/2,1}(x',y') = -mv_{1,1/2}(x,y) \quad \textbf{(4.4)}$$

$$mv_{1/2,0}(x',y') = mv_{1,1/2}(x,y) \quad \textbf{(4.5)}$$

4.3 Motion Segmentation Using Background/Foreground Segmentation

Before we can start to reconstruct frame ½ one important step is needed to apply the reconstruction successfully. The background/foreground identities of pixels in frame ½ need to be determined. Since $mv_{1/2,1}(x', y') = -mv_{1,1/2}(x, y)$, if pixel at coordinates (x, y) belongs to foreground then pixel at coordinates (x', y') belongs to foreground, too. Otherwise pixel at coordinates (x', y') belongs to background.

As a consequence of the many-to-one motion vector mapping from frame 1 to frame ½, pixels defined by multiple motion vectors (termed overlapped pixels) or pixels not defined by any motion vector (gap pixels) as well as pixels defined by a single motion vector can occur in frame ½. Each of the three types is handled differently in the reconstruction stage.

4.3.1 Overlapped Pixels

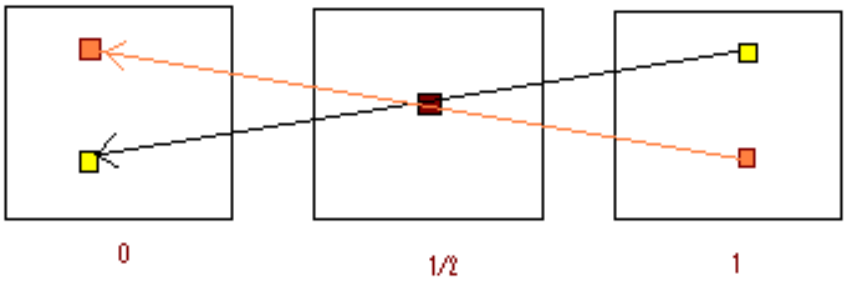


Figure 4.3 : Illustration of overlapped pixel

For the overlapped pixel type as shown by Figure 4.3, more than one motion vector from frame 1 to frame ½ maps to the same location and the most reliable one needs to be determined. We propose that the motion vector $mv_{1,1/2}(x_i, y_i)$ with the minimum value of $(x'-\tilde{x}_i)^2 + (y'-\tilde{y}_i)^2$ be used. If there is more than one such motion vector, we use the one for which $mv_{1,0}(x_i, y_i)$ has the least block matching error (SAD) between frames 1 and 0.

$$mv_{1,1/2}(x', y') = 0.5 \times (\arg\min_{mv_{1,0}(x,y)}(SAD(mv_{1,0}(x_i, y_i)))) \quad \textbf{(4.6)}$$

Then, we use $mv_{1/2,1}(x', y') = -mv_{1,1/2}(x_i, y_i)$ to transfer pixel value from frame 1 to frame ½ by bilinear interpolation of pixel values. Since we have used half pixel

accurate pixels we need to interpolate two pixel values that are close to half pixel accurate point in frame 1 via following equations.

$$x'' = x' + mv_{1/2,1,x}(x', y') \quad \textbf{(4.7)}$$

$$y'' = y' + mv_{1/2,1,y}(x', y') \quad \textbf{(4.8)}$$

$$x_1 = \lfloor x'' \rfloor;\ x_2 = \lceil x'' \rceil \quad \textbf{(4.9)}$$

$$y_1 = \lfloor y'' \rfloor;\ y_2 = \lceil y'' \rceil \quad \textbf{(4.10)}$$

$$f_{1/2}(x', y') = \begin{pmatrix} f_1(x_1, y_1) + f_1(x_2, y_2) + \\ f_1(x_1, y_2) + f_1(x_2, y_1) \end{pmatrix} \times 0.25 \quad \textbf{(4.11)}$$

4.3.2 Gap Pixels

A gap pixel does not have any motion vector assigned to it. A motion vector for it is determined by interpolative filtering of the assigned motion vectors of nearby pixels. Three types gap pixels at coordinates (x', y') can be identified:

- Gap pixel at coordinates (x', y') is surrounded on all sides by foreground pixels as shown in Figure 4.4 (vertically when going up or down or horizontally when going left or right in frame ½ all four nearest pixels with known motion vectors belong to foreground region). In this case, we determine $mv_{1/2,0}(x', y')$ by bilinear interpolation of these four motion vectors of the nearest foreground pixels.

$$mv_{1/2,0}(x', y') = \begin{pmatrix} (\delta/(\delta+\alpha)) \times mv_{1/2,0}(x' - \alpha, y') + \\ (\sigma/(\sigma+\beta)) \times mv_{1/2,0}(x', y' - \beta) + \\ (\alpha/(\delta+\alpha)) \times mv_{1/2,0}(x' + \delta, y') + \\ (\beta/(\sigma+\beta)) \times mv_{1/2,0}(x', y' + \sigma) \end{pmatrix} \times 0.5 \quad \textbf{(4.12)}$$

  α, β, δ, and σ are the distances to the neighboring pixels that are not gap pixels and they are illustrated in Figure 4.4. Then, we use $mv_{1/2,0}(x', y')$ to unidirectionally transfer bilinear interpolated pixel value from frame 0 to frame ½.

$$f_{1/2}(x', y') = \begin{pmatrix} f_0(x_1, y_1) + f_0(x_2, y_2) + \\ f_0(x_1, y_2) + f_0(x_2, y_1) \end{pmatrix} \times 0.25 \quad \textbf{(4.13)}$$

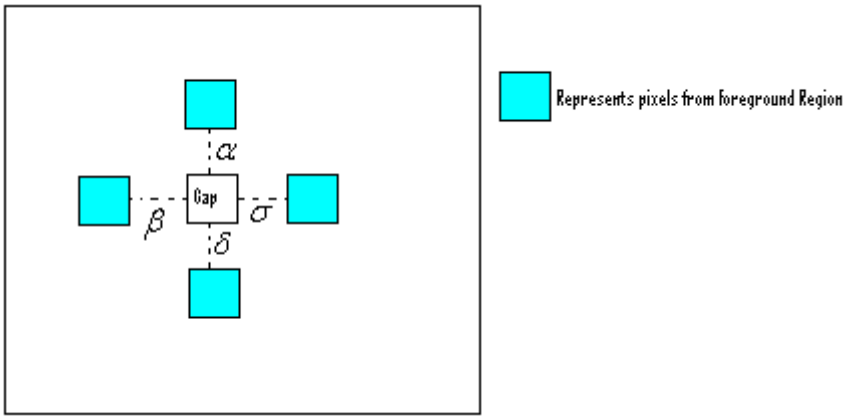


Figure 4.4 : All neighbors of Gap are from Foreground Region

- Gap pixel at coordinates (x', y') is surrounded on all sides by background (vertically when going up or down or horizontally when going left or right in frame ½ all four nearest pixels belong to background region). In this case, we determine $mv_{1/2,0}(x', y')$ by bilinear interpolation of four motion vectors of the nearest background pixels. Then, we use $mv_{1/2,0}(x', y')$ to unidirectionally transfer bilinear interpolated pixel value from frame 0 to frame ½.

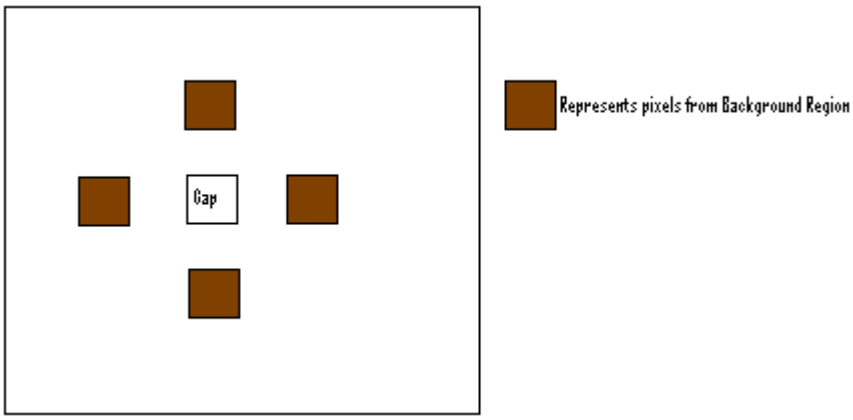


Figure 4.5 : All neighbors of Gap are from Background Region

- Gap pixel at coordinates (x', y') is surrounded on some sides by background and other sides by foreground (vertically when going up or down or

horizontally when going left or right in frame ½ some of the nearest pixels belong to background region and others belong to the foreground region). In this case, we determine $mv_{1/2,0}(x',y')$ by bilinear interpolation of motion vectors of the nearest background or foreground pixels. Bilinear interpolation is used for determining $mv_{1/2,0}(x',y')$, if horizontally or vertically the nearest pixels on both sides are background. If only one nearest pixel on one side is background, we interpolate the motion vector information from remaining three neighbors. For opposite neighboring pixels having different foreground/background identity both vertically and horizontally shown in Figure 4.6, we use the correlation of neighboring vectors to determine which neighbor's motion vectors should be used. We obtain motion vector of gap pixel by interpolating neighbors' that gives largest correlation value.

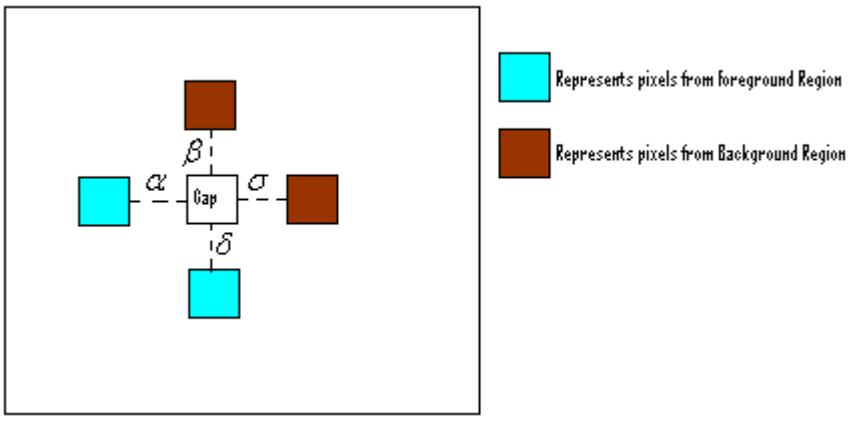


Figure 4.6 : Correlation of vectors are used for this kind of positions

$$mv_{1/2,0,F}(x',y') = \begin{bmatrix} ((\delta/(\delta+\alpha)) \times mv_{1/2,0}(x',y'-\alpha)) + \\ ((\alpha/(\delta+\alpha)) \times mv_{1/2,0}(x'+\delta,y')) \end{bmatrix} \quad \textbf{(4.14)}$$

$$mv_{1/2,0,B}(x',y') = \begin{bmatrix} ((\beta/(\beta+\sigma)) \times mv_{1/2,0}(x',y'+\sigma)) + \\ ((\sigma/(\beta+\sigma)) \times mv_{1/2,0}(x'-\beta,y')) \end{bmatrix} \quad \textbf{(4.15)}$$

$$v_{1,F} = mv_0\left(\left(x'+mv_{1/2,0,F,x}(x',y')\right),\left(y'+mv_{1/2,0,F,y}(x',y')\right)\right) \quad \textbf{(4.16)}$$

$$v_{2,F} = mv_0\left(\left(x'+mv_{1/2,0,x}(x',y'-\alpha)\right),\left((y'-\alpha)+mv_{1/2,0,y}(x',y'-\alpha)\right)\right) \quad \textbf{(4.17)}$$

$$v_{3,F} = mv_0\left(\left((x'+\delta)+mv_{1/2,0,x}(x'+\delta,y')\right),\left(y'+mv_{1/2,0,y}(x'+\delta,y')\right)\right) \quad \textbf{(4.18)}$$

$$v_{1,B} = mv_0\left(\left(x'+mv_{1/2,0,B,x}(x',y')\right),\left(y'+mv_{1/2,0,B,y}(x',y')\right)\right) \quad \textbf{(4.19)}$$

$$v_{2,B} = mv_0\left(\left(x' + mv_{1/2,0,x}(x', y' + \sigma)\right), \left((y' + \sigma) + mv_{1/2,0,y}(x', y' + \sigma)\right)\right) \quad \textbf{(4.20)}$$

$$v_{3,B} = mv_0\left(\left((x' - \beta) + mv_{1/2,0,x}(x' - \beta, y')\right), \left(y' + mv_{1/2,0,F,y}(x' - \beta, y')\right)\right) \quad \textbf{(4.21)}$$

We have used following equation to find correlation of vectors. If two vectors are similar then the correlation value converges to 0, but if they are not similar then correlation value diverges from 0.

$$Cor(v_1, v_2) = \frac{\|v_1 - v_2\|}{\|v_1\| . \|v_2\|} \quad \textbf{(4.22)}$$

Finally, using following equations we decide which motion vector should be assigned to the gap pixel.

$$C1 = \left(Cor\left(v_{1,F}, v_{2,F}\right) + Cor\left(v_{1,F}, v_{3,F}\right)\right) * 0.5 \quad \textbf{(4.23)}$$

$$C2 = \left(Cor\left(v_{1,B}, v_{2,B}\right) + Cor\left(v_{1,B}, v_{3,B}\right)\right) * 0.5 \quad \textbf{(4.24)}$$

$$C2 > C1 \rightarrow mv_{1/2,0}(x', y') = mv_{1/2,0,F}(x', y') \quad \textbf{(4.25)}$$

$$C1 > C2 \rightarrow mv_{1/2,0}(x', y') = mv_{1/2,0,B}(x', y') \quad \textbf{(4.26)}$$

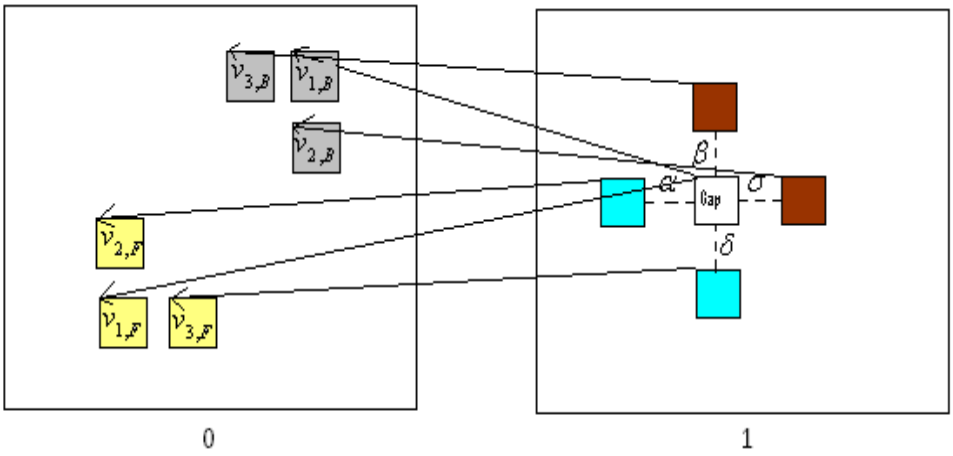


Figure 4.7 : Shows up which motion vectors are used for correlation in the frame 0

Finally, we use $mv_{1/2,0}(x', y')$ to transfer bilinear interpolated pixel value from frame 0 to frame ½.

4.3.3 One Motion Vector Assigned Pixel Compensation

For a pixel in frame ½ at coordinates (x', y') with one motion vector assigned to it, both $mv_{1/2,1}(x', y')$ and $mv_{1/2,0}(x', y')$ are used to transfer the average of bilinear

interpolated values from frames 1 and 0. Hence for such a pixel the motion compensation is bidirectional.

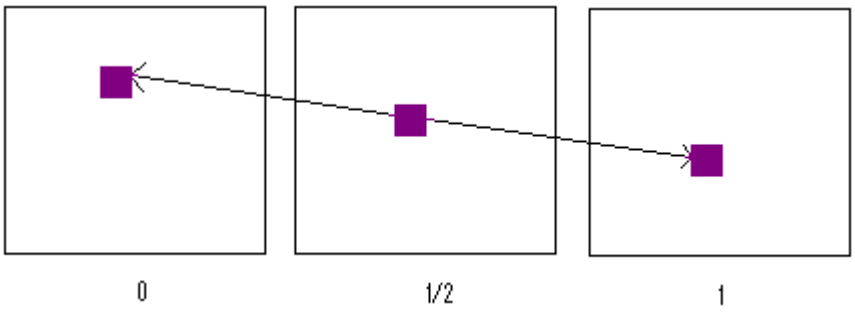


Figure 4.8 : Shows up motion vectors used for Bidirectional Compensation

For reducing blocking artifacts and improving quality of interpolated frame, overlapped block motion compensation (OBMC) can be applied. In this method, each block is reconstructed as weighted average of multiple predictions using MVs of both current block and its neighbouring blocks [4]. Some conventional video coders can cause blocking artifacts. OBMC is a good motion compensation method to remove these problems. Therefore, it provides interpolated pictures with high quality.

The proposed algorithm described above was integrated with a foreground/background segmentation method designed for videos which have stationary camera view. The segmentation method uses Gaussian mixture model (GMM). GMM is a background subtraction method that is improved to cope with pixel values with complex distributions [11]. Complex distributions are estimated with the help of Gaussian mixture probability density.

4.4 Pseudo Code of Proposed Algorithm

$mv_{i,j}(x,y)$: motion vector at (x,y) points from frame i to frame j.

1. Determine foreground region in frame 1 via existent software

2. a. Scale motion vectors from $mv_{1,0}(x,y)$ to $mv_{1,1/2}(x,y)$ for all (x,y). (If $mv_{1,0}(x,y)$ has half pixel accuracy, $mv_{1,1/2}(x,y)$ has quarter pixel accuracy. If $mv_{1,0}(x,y)$ has full pixel accuracy, $mv_{1,1/2}(x,y)$ has half pixel accuracy.)

   b. For each (x,y) determine $\tilde{x} = x - mv_{x,1,1/2}(x,y)$ $\tilde{y} = y - mv_{y,1,1/2}(x,y)$ where $(\tilde{x},\tilde{y})$ is a half or quarter pixel accurate point in frame ½ . Now, for all pixels (x',y') in half pixel neighborhood of $(\tilde{x},\tilde{y})$ (i.e. $(|x'-\tilde{x}| \le 0.5) AND (|y'-\tilde{y}| \le 0.5)$) in frame ½, assign $mv_{1,1/2}(x,y)$ to (x',y').

   In other words $mv_{1/2,1}(x',y') = -mv_{1,1/2}(x,y)$. Also $mv_{1/2,0}(x',y') = mv_{1,1/2}(x,y)$.

3. The foreground/background identity of pixel (x',y') in frame ½ is established as follows: Since $mv_{1/2,1}(x',y') = -mv_{1,1/2}(x,y)$ if (x,y) is foreground (x',y') becomes foreground otherwise (x',y') becomes background.

4. There are three possibilities:

   a. More than one scaled motion vector is assigned to the same location (x',y') in frame ½ from frame 1 by the procedure in step 2 b.:

      i. Select the motion vector $mv_{1,1/2}(x_i,y_i)$ for which $(x'-\tilde{x}_i)^2 + (y'-\tilde{y}_i)^2$ is minimum. If there is more than one such motion vector use the one for which $mv_{1,0}(x_i,y_i)$ has the least block matching error (SAD) between frames 1 and 0.

$$mv_{1,1/2}(x',y') = 0.5 \times (\arg\min_{mv_{1,0}(x,y)}(SAD(mv_{1,0}(x_i,y_i)))) \quad \textbf{(4.27)}$$

      ii. Use $mv_{1/2,1}(x',y') = -mv_{1,1/2}(x_i,y_i)$ to transfer pixel value from frame 1 to frame ½ by bilinear interpolation.

$$x'' = x' + mv_{1/2,1,x}(x',y') \quad \textbf{(4.28)}$$

$$y'' = y' + mv_{1/2,1,y}(x',y') \quad \textbf{(4.29)}$$

$$x_1 = \lfloor x'' \rfloor \ , \ x_2 = \lceil x'' \rceil \quad \textbf{(4.30)}$$

$$y_1 = \lfloor y'' \rfloor \ , \ y_2 = \lceil y'' \rceil \quad \textbf{(4.31)}$$

$$f_{1/2}(x', y') = 0.25 \times \begin{pmatrix} f_1(x_1, y_1) + f_1(x_2, y_2) + \\ f_1(x_1, y_2) + f_1(x_2, y_1) \end{pmatrix} \quad \textbf{(4.32)}$$

b. No motion vector is assigned to location (x', y') in frame ½ (gap) by the procedure in step 2 b.:

i. Apply interpolative filtering to determine $mv_{1/2,1}(x', y')$ depending on type of region (x', y') belongs. The three types are:

1. (x', y') is surrounded on all sides by foreground (vertically when going up or down or horizontally when going left or right in frame ½ all four nearest pixels with known motion vectors belong to foreground region):

Determine $mv_{1/2,0}(x', y')$ by bilinear interpolation of these four motion vectors of the nearest foreground pixels.

$$mv_{1/2,0}(x', y') = \begin{pmatrix} (\delta/(\delta+\alpha)) \times mv_{1/2,0}(x'-\alpha, y') + \\ (\sigma/(\sigma+\beta)) \times mv_{1/2,0}(x', y'-\beta) + \\ (\alpha/(\delta+\alpha)) \times mv_{1/2,0}(x'+\delta, y') + \\ (\beta/(\sigma+\beta)) \times mv_{1/2,0}(x', y'+\sigma) \end{pmatrix} \times 0.5 \quad \textbf{(4.33)}$$

Use $mv_{1/2,0}(x', y')$ to unidirectionally transfer pixel value from frame 0 to frame ½ by bilinear interpolation.

$$x'' = x' - mv_{1/2,0,x}(x', y') \quad \textbf{(4.34)}$$

$$y'' = y' - mv_{1/2,0,y}(x', y') \quad \textbf{(4.35)}$$

$$x_1 = \lfloor x'' \rfloor \ , \ x_2 = \lceil x'' \rceil \quad \textbf{(4.36)}$$

$$y_1 = \lfloor y'' \rfloor \ , \ y_2 = \lceil y'' \rceil \quad \textbf{(4.37)}$$

$$f_{1/2}(x', y') = \begin{pmatrix} f_0(x_1, y_1) + f_0(x_2, y_2) + \\ f_0(x_1, y_2) + f_0(x_2, y_1) \end{pmatrix} \times 0.25 \quad \textbf{(4.38)}$$

2. (x', y') is surrounded on all sides by background (vertically when going up or down or horizontally when going left or

right in frame ½ all four nearest pixels belong to background region)

Determine $mv_{1/2,0}(x',y')$ by bilinear interpolation of four motion vectors of the nearest background pixels.

$$mv_{1/2,0}(x',y') = \begin{pmatrix} (\delta/(\delta+\alpha)) \times mv_{1/2,0}(x'-\alpha, y') + \\ (\sigma/(\sigma+\beta)) \times mv_{1/2,0}(x', y'-\beta) + \\ (\alpha/(\delta+\alpha)) \times mv_{1/2,0}(x'+\delta, y') + \\ (\beta/(\sigma+\beta)) \times mv_{1/2,0}(x', y'+\sigma) \end{pmatrix} \times 0.5 \quad \textbf{(4.39)}$$

Use $mv_{1/2,0}(x',y')$ to unidirectionally transfer pixel value from frame 0 to frame ½ by bilinear interpolation.

$$x'' = x' - mv_{1/2,0,x}(x',y') \quad \textbf{(4.40)}$$

$$y'' = y' - mv_{1/2,0,y}(x',y') \quad \textbf{(4.41)}$$

$$x_1 = \lfloor x'' \rfloor,\ x_2 = \lceil x'' \rceil \quad \textbf{(4.42)}$$

$$y_1 = \lfloor y'' \rfloor,\ y_2 = \lceil y'' \rceil \quad \textbf{(4.43)}$$

$$f_{1/2}(x',y') = \begin{pmatrix} f_0(x_1,y_1) + f_0(x_2,y_2) + \\ f_0(x_1,y_2) + f_0(x_2,y_1) \end{pmatrix} \times 0.25 \quad \textbf{(4.44)}$$

3. (x',y') is surrounded on some sides by background and other sides by foreground (vertically when going up or down or horizontally when going left or right in frame ½ some of the nearest pixels belong to background region and others belong to the foreground region)

   Determine $mv_{1/2,0}(x',y')$ by bilinear interpolation of motion vectors of the nearest background pixels.

Use interpolation if horizontally or vertically the nearest pixels on both sides are background. If only one nearest pixel on one side is background, interpolate the motion vector information from remaining three neighbors. If opposite neighboring pixels have different foreground/background, use the correlation of neighboring vectors to determine which neighbor's motion vectors should be used.

Obtain motion vector of gap pixels by interpolating neighbors' that is giving large correlation value.

Use $mv_{1/2,0}(x',y')$ to transfer pixel value from frame 0 to frame ½ by bilinear interpolation.

$$x'' = x' - mv_{1/2,0,x}(x',y') \quad \textbf{(4.45)}$$

$$y'' = y' - mv_{1/2,0,y}(x',y') \quad \textbf{(4.46)}$$

$$x_1 = \lfloor x'' \rfloor,\ x_2 = \lceil x'' \rceil \quad \textbf{(4.47)}$$

$$y_1 = \lfloor y'' \rfloor,\ y_2 = \lceil y'' \rceil \quad \textbf{(4.48)}$$

$$f_{1/2}(x',y') = \begin{pmatrix} f_0(x_1,y_1) + f_0(x_2,y_2) + \\ f_0(x_1,y_2) + f_0(x_2,y_1) \end{pmatrix} \times 0.25 \quad \textbf{(4.49)}$$

c. There is one motion vector assigned to location (x',y') in frame ½:

Use $mv_{1/2,1}(x',y')$ and $mv_{1/2,0}(x',y')$ (determined in step 2 b.) to bidirectionally transfer values from frames 0 and 1 by bilinear interpolation and average the two values to obtain the pixel value at location (x',y').

$$x'' = x' + mv_{1/2,0,x}(x',y'),\ x''' = x' + mv_{1/2,1,x}(x',y') \quad \textbf{(4.50)}$$

$$y'' = y' + mv_{1/2,0,y}(x',y'),\ y''' = y' + mv_{1/2,1,y}(x',y') \quad \textbf{(4.51)}$$

$$x_1 = \lfloor x'' \rfloor,\ x_2 = \lceil x'' \rceil \quad \textbf{(4.52)}$$

$$y_1 = \lfloor y'' \rfloor,\ y_2 = \lceil y'' \rceil \quad \textbf{(4.53)}$$

$$x_3 = \lfloor x''' \rfloor,\ x_4 = \lceil x''' \rceil \quad \textbf{(4.54)}$$

$$y_3 = \lfloor y''' \rfloor,\ y_4 = \lceil y''' \rceil \quad \textbf{(4.55)}$$

$$f_{1/2}(x',y') = \begin{pmatrix} \begin{pmatrix} f_0(x_1,y_1) + f_0(x_2,y_2) + \\ f_0(x_1,y_2) + f_0(x_2,y_1) \end{pmatrix} \times 0.25 + \\ \begin{pmatrix} f_1(x_3,y_3) + f_1(x_4,y_4) + \\ f_1(x_3,y_4) + f_1(x_4,y_3) \end{pmatrix} \times 0.25 \end{pmatrix} \times 0.5 \quad \textbf{(4.56)}$$

5. EXPERIMENTS

In this part, we are showing test results of our work using some test sequences. In video processing area, showing quality of the video has a significant place. Therefore, it should be measured correctly, but none of the current measurement methods can show quality of the video with perfect results. We have used some objective methods to represent accuracy of processed video.

One of the methods we have used is peak signal to noise ratio (PSNR) which is the most popular measurement method in video processing area. There is brief information below about PSNR. For human eye, quality of the video can be understood from its structural form. Therefore, only PSNR measurement is not enough to show quality and structural similarity test (SSIM) [17] has been used to represent sutructural quality of the video. This method compares local patterns of pixel intensities that have been normalized for luminance and contrast.

5.1 Peak Signal To Noise Ratio (PSNR)

As stated above, PSNR is the most widely used method for mesurement of quality of reconstructed frames. As can also be unterstood from its name, it is a ratio between the peak possible power of the signal and power of the noise that has degrading effect on quality of the image.

PSNR can be thought as a method approximation to human perception, therefore it may not show quality of the video perfectly. For some frames, PSNR can be low but structural similarity between reference frame and reconstructed frame can be high. It can be nearly same. PSNR can be calculated based on mean squared error (MSE). In following equation, m and n represents dimensions of the frame. G and H are reference and compensated frames.

$$MSE = \frac{1}{m \times n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left[G(i,j) - H(i,j) \right]^2 \quad \textbf{(5.1)}$$

$$PSNR = 10 \times \log_{10} \left(\frac{MAX_I^{\,2}}{MSE} \right) \quad \textbf{(5.2)}$$

where MAX_I is the maximum possible pixel value of the image. It is equal to 255.

5.2 Structural Similarity Test (SSIM)

SSIM is applied on luma components and it can have value between -1 and 1. If two frames are the same, then SSIM value is 1. SSIM can be calculated with a set of equations. x and y represents two nonnegative image signals. First we should find average of x and y.

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \textbf{(5.3)}$$

μ_y can be found in a similar manner. Next, variance of x and y signals are found.

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - {\mu_x}^2)\right)^{1/2} \quad \textbf{(5.4)}$$

Finally, covariance of two signals are found by the following equation.

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y) \quad \textbf{(5.5)}$$

Thus, SSIM is found via following method.

$$SSIM(x,y) = \frac{2(\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{({\mu_x}^2 + {\mu_y}^2 + C_1)({\sigma_x}^2 + {\sigma_y}^2 + C_2)} \quad \textbf{(5.6)}$$

In equation 5.6, C1 and C2 is used to stabilize the division. They can be determined by the following equations where $k_1 = 0.01$ and $k_2 = 0.03$ and L is dynamic range of the pixel values.

$$C_1 = (k_1 L)^2 \qquad C_2 = (k_2 L)^2 \quad \textbf{(5.7)}$$

5.3 Experimental Results

We made some comparison tests to show fairness and effectiveness of our algorithm. We have used four un-compressed video sequences to compare the proposed method with the method described in [9]. They are Foreman, Carphone, Garden and Mobile. The resolutions of the sequences are 176 x 144 (QCIF), 176 x 144, 352 x 240, and 352 x 288 respectively. 50 frames are interpolated from two consecutive odd frames

for measurement (2,4,6,8,…..,100). Temporal frequency of videos were increased to 30fps from 15fps. Peak signal to noise ratio (PSNR) results of proposed algorithm are tabulated in Table 5.1.

For structural comparison, we have also used SSIM method to show fairness of our method. As can be seen from Table 5.1 that the proposed method constructs structurally better frames than [9]. Both SSIM and PSNR measurements of our algorithm gives better results than method described in [9].

Table 5.1: SSIM and PSNR Performances

Video Sequences	**Method in [9]**	**Method in [9]**	**Proposed Method**	**Proposed Method**
	PSNR(dB)	SSIM	PSNR(dB)	SSIM
Carphone	33.2780	0.9542	**34.066**	**0.970**
Foreman	30.6988	**0.9543**	**32.343**	0.927
Mobile	23.3280	0.9106	**27.670**	**0.942**
Garden	23.7904	0.8938	**28.117**	**0.945**

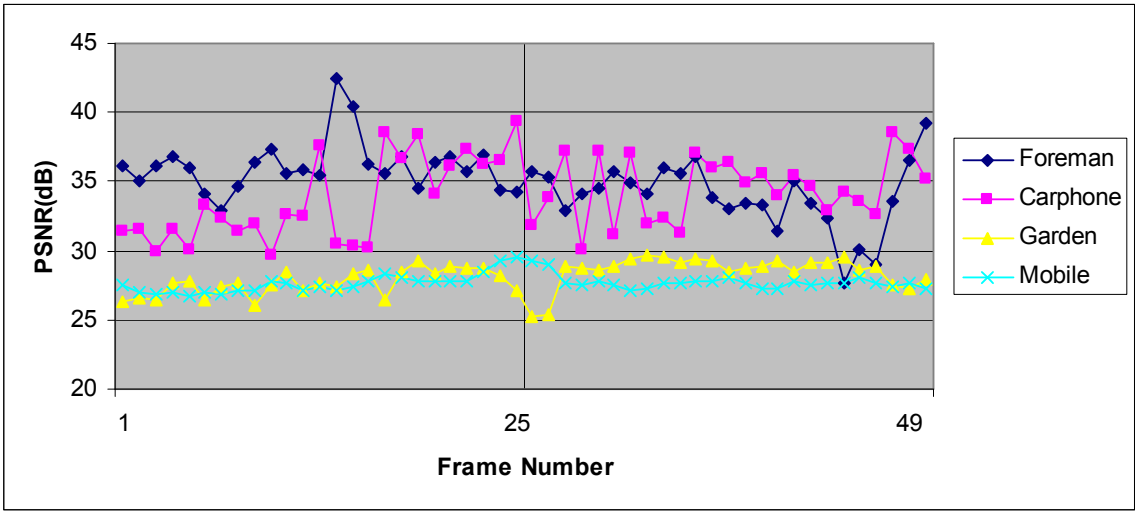
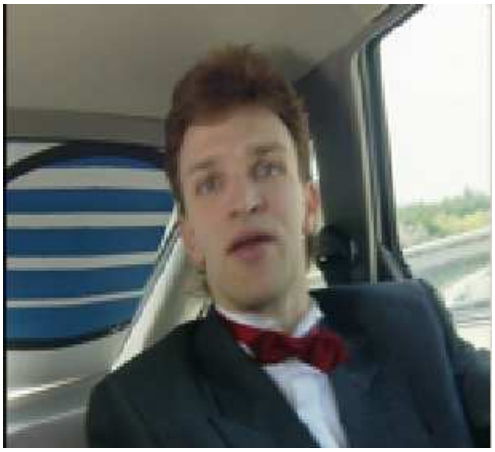


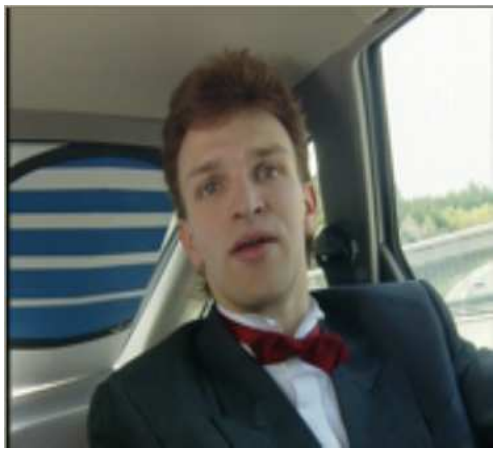
Figure 5.1 : Quality representation of video sequences (Foreman, Carphone, Garden, and Mobile)

Figure 5.2 and Figure 5.3 shows up interpolated frames from Carphone and Foreman

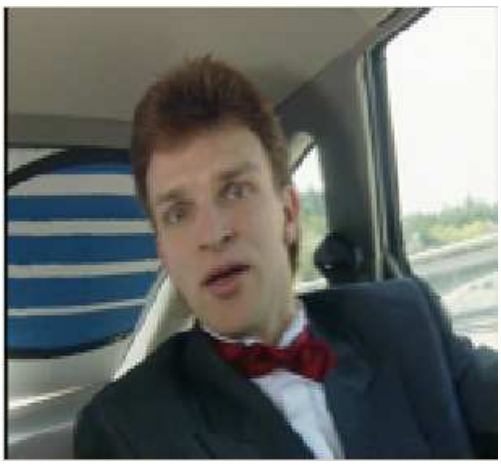
sequences. Since Carphone sequence has complex motions and Foreman sequence has unstable camera scene, illustrated frames are not better than interpolated frames of News and Akiyo that are shown in Figure 5.5 and Figure 5.6.



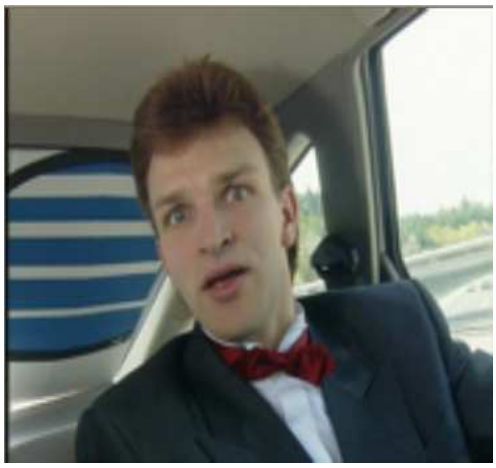
a) Interpolated frame 18



b) Original frame 18



c) Interpolated frame 38



d) Original frame 38

Figure 5.2 : Interpolated frames of 17 and 38 with proposed algorithm from Carphone video sequence

a) Interpolated frame 24 b) Original frame 24

c) Interpolated frame 46 d) Original frame 46

Figure 5.3 : Interpolated frames of 24 and 45 with proposed algorithm from Foreman video sequence

We also compared our simulation results with the results obtained in [14]. We have used four video sequences that are in CIF (352x288) format. They are Akiyo, News, Mobile and Stefan. Since first 150 interpolated frames were used for Akiyo and Mobile sequences, and first 50 interpolated frames were used for Stefan and News in [14] for measurement, we also followed the same simulation path. You can find PSNR results of proposed algorithm tabulated in Table 5.2. There is also gain of the algorithm over the [14] in following table.

Table 5.2: Gain of proposed method over [14]

Video Sequences	**Method in [14] (dB)**	**Proposed Method (dB)**	**Gain from proposed method**
Akiyo	39.458	**46.752**	7.294
News	32.386	**37.065**	4.679
Mobile	20.757	**28.252**	7.495
Stefan	22.347	**26.813**	4.466

You can find SSIM results of our algorithm tabulated in table 5.3, but there is no comparison with [14], because [14] does not contain SSIM results. Therefore, we have only made PSNR comparison.

Table 5.3: PSNR and SSIM Performances

Video Sequences	**Method in [14]**	**Proposed Method**	**Proposed Method**
	PSNR(dB)	PSNR(dB)	SSIM
Akiyo	39.458	**46.752**	0.996
News	32.386	**37.065**	0.981
Mobile	20.757	**28.252**	0.951
Stefan	22.347	**26.813**	0.916

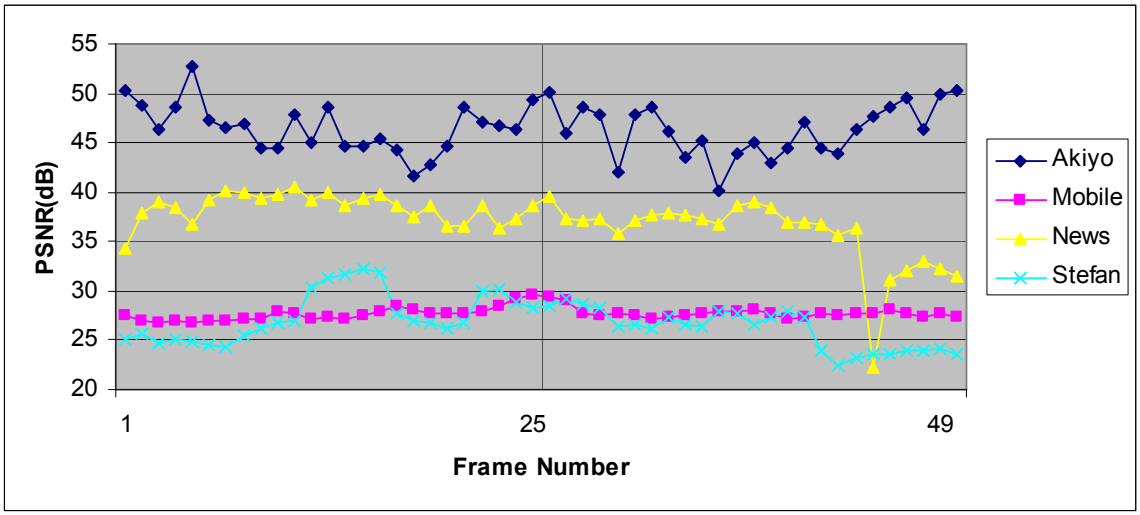


Figure 5.4 : Quality representation of video sequences (Akiyo, Mobile, News, and Stefan)

There are some interpolated frame samples from Akiyo and News sequences in following pages. You will notice that these frames are reconstructed better than Carphone and Foreman frames as represented in figures 5.2 and 5.3. The reason of this difference is that the segmentation method used in this thesis gives better results when the video has stationary camera view and there is no complex object motions in the video. Since News and Akiyo sequences have stationary camera, interpolated frames have better quality then Carphone's and Foreman's. In the future, if more powerful segmentation algorithms are found, they can be integrated to this work and thus better frames can be obtained for Foreman and Carphone.

a) Interpolated frame 48 b) Original frame 48

c) Interpolated frame 76 d) Original frame 76

Figure 5.5 : Interpolated frames of 48 and 75 with proposed algorithm from Akiyo sequence

a) Interpolated frame 40

b) Original frame 40

c) Interpolated frame 76

d) Original frame 76

Figure 5.6 : Interpolated frames of 39 and 75 with proposed algorithm from News sequence

Table 5.4 demonstrates PSNR and structural similarity (SSIM) test results of all tested videos.

Table 5.4: PSNR and SSIM results of all tested sequences

Video Sequences	**PSNR (dB)**	**SSIM**
Akiyo	46.752	0.996
News	37.065	0.976
Foreman	32.343	0.927
Carphone	34.066	0.967
Mobile	28.252	0.951
Garden	28.117	0.945
Stefan	26.813	0.916

In the pictures of Figure 5.7 and Figure 5.8 one can subjectively make comparison with [14] and our algorithm. In figure 5.7, one sees that our algorithm reconstructs a frame without artifacts and thus its objective quality is higher than [14]. Furthermore, some artifacts (seen on numbers 17, 28 and 29 in Mobile sequence) are removed by our algorithm for a better reconstruction in Figure 5.8. In this figure it can be seen that the funnel of the train is more properly reconstructed by our proposed algorithm. The artifacts in the reconstruction by [14] make this shape appear to have an inclined form.

(a)

(b)

(c)

Figure 5.7 : Comparison of Akiyo Sequence with a) [14] and b) Proposed Algorithm c) Original frame 70

Table 5.5 shows us comparison of the algorithm which is using correlation based motion vector processing proposed by Nguyen. He is using video sequences that are compressed by H.264, but we used raw video sequences in our tests. While some of the PSNR and SSIM results are better than [20], some of them are worse.

Table 5.5: Comparison with proposed algorithm via [20]

Video Sequences	**Method in [20]**	**Method in [20]**	**Proposed Method**	**Proposed Method**
	PSNR(dB)	SSIM	PSNR(dB)	SSIM
Football	**25.110**	**0.780**	22.237	0.670
Foreman	31.700	**0.960**	**32.343**	0.927
Stefan	26.490	0.900	**26.568**	**0.905**

(a)

(b)

(c)

Figure 5.8 : Comparison of Mobile Sequence with a) [14] and b) Proposed Algorithm c) Original frame 40

Table 5.6 shows us reliability of the proposed algorithm. For comparison, we have reconstructed each video sequence with three different methods described below. The proposed algorithm gives better PSNR and SSIM results.

LinearMC: Bi-directional linear motion compensation is used.

NoSegmentation: Proposed Algorithm with gap pixel motion vectors reconstructed using linear interpolation.

Segmentation: Proposed Algorithm with motion based segmentation described in Chapter 3.

Table 5.6: LinearMC, NoSegmentation and Segmentation Comparison

Video Sequences	**LinearMC**		**NoSegmentation**		**Segmentation**	
	PSNR(db)	SSIM	PSNR(db)	SSIM	PSNR(db)	SSIM
Akiyo	43.075	0.993	46.743	0.996	**46.752**	**0.996**
News	34.325	0.973	**37.080**	**0.981**	37.065	0.976
Foreman	30.495	0.906	30.832	0.917	**32.343**	**0.927**
Carphone	30.830	0.916	33.092	0.963	**34.066**	**0.967**
Mobile	24.646	0.900	28.077	0.946	**28.252**	**0.951**
Garden	26.057	0.926	27.430	0.939	**28.117**	**0.945**
Stefan	24.230	0.843	26.284	0.913	**26.813**	**0.916**

6. CONCLUSION

In this thesis, we introduced a foreground/background segmentation based frame rate up-conversion method. For MC-FRUC, the difficulty of identifying covered-uncovered regions introduces artifacts into the interpolated frame. In this work, the foreground/background segmentation data was incorporated into the MC-FRUC algorithm as a solution to this problem. Experimental results with several test video sequences have shown that this new method has better reconstruction performance in terms of PSNR and SSIM than that of previous works such as [9], [13] and [14] on interpolated frames. Our method removes blocking artifacts and thus it provides higher subjective quality of view.

The performance of the proposed MC-FRUC method depends on the employed segmentation algorithm. Currently, only segmentation algorithms capable of handling stationary background have been integrated with MC-FRUC method. In the future, we plan to test the proposed method with more advanced segmentation algorithms that can handle panning or more complex motions of background.

REFERENCES

[1] S. Ince, J.Konrad, "Geometry-Based Estimation of Occlusions From Video Frame Pairs", *IEEE Acoustics, Speech and Signal Processing*, Pages ii/933 - ii/936 Vol. 2, 2005.

[2] P.M Jodoin, C. Rosenberger, M. Mignotte, "Detecting Half-Occlusion with a Fast Region Based Fusion Procedure", *British Machine Vision Conference*, Edinburgh, Vol. 16, Pages 2535-2550, September 2007.

[3] Wei Hong, "Low-Complexity Occlusion Handling for Motion-Compensated Frame Rate Up-Conversion", *Consumer Electronics 2009, ICCE'09*, *Digest of Technical Papers International Confrence*, Pages 1-2, 2009.

[4] H.F. Ates, B.Cizmeci, "Occlusion artifact removal in video frame rate upconversion", *In 17th Signal Processing and Communications Applications Conference, (SIU 2009)*, pages 277–280, Antalya, Turkey, 2009. IEEE.

[5] G. De Haan, Paul W.A.C. Biezen, H. Huijgen, O. A. Ojo, "True Motion Estimation With 3-D Recursive Search Block Matching", *In IEEE Transactions Circuits and Systems Video Technologies, IEEE Press*, Page 368, August 2002.

[6] J. K. Su, R. M. Mersereau, "Motion Estimation Methods for Overlapped Block Motion Compensation", *Image Processing, IEEE Transactions*, Volume 9, Pages 1509-1521, 2000.

[7] A. M. Huang, T.Q. Nguyen, "Correlation-based motion vector processing for motion compensated frame interpolation", *IEEE Transactions on Image Processing,* Volume 17, Pages 694–708, 2008

[8] M. Cetin, and I. Hamzaoglu, "An Adaptive True Motion Estimation Algorithm for Frame Rate Conversion of High Definition Video", *International Conference on Pattern Recognition 2010,* Pages 4109-4112.

[9] X. Gao, Y. Yang, B. Xiao, "Adaptive frame rate up-conversion based on motion classification", *Elsevier, International Journal of Signal Processing,* Volume 88, Pages 2979-2988, 2008.

[10] A. N. Netravali, J. D. Robbins, **"**Motion-adaptive interpolation of television frames", *IEICE Transactions on Information and Systems,* Pages 1117-1126, 2008.

[11] Zoran Zivkovic, "Improved Adaptive Gaussian Mixture Model for Background Subtraction", *Pattern Recognition 2004, ICPR 2004, Proceedings of 17th International Conference*, Vol-2, Pages 28-31, 2004.

[12] R. Cucchiara, M. Piccardi, A.Prati, "Detecting Moving Objects, Ghosts, and Shadows in Video Streams", *IEEE Pattern Analysis and Machine Intelligence*, Pages 1337-1342 Vol-25, 2003.

[13] Y.M Chen, IvanV. Bajic, C.Qian, "Frame Rate Up-Conversion of Compressed Video Using Region Segmentation and Depth Ordering", *Communications, Computers and Signal Processing*, Pages 431-436, 2009.

[14] S.J Kung, D.G Yoo, S.K. Lee, Y.H Kim, "Design and Implementation of Median Filter based Adaptive Motion Vector Smoothing for Motion Compensated Frame Rate Up-Conversion", *IEEE Consumer Electronics ISCE'09 13th International Symposium*, Pages 745-748, 2009.

[15] J. C. Silveira, Jacques C. R. Jung, S.R. Musse, "A Background Subtraction Model Adapted to Illumination Changes", *Proc. ICIP'06*, Page1817-1820, 2006.

[16] Y. Wang, T. Tan, K.F. Loe, J.K. Wu, "A Probabilistic Approach for Foreground and Shadow Segmentation in Monocular Image Sequences", *Pattern Recognition,* Vol. 38, Iss. 11, Pages1937-1946, Nov. 2005.

[17] H. R. Sheikh, E.P Simoncelli, Z. Wang, A.C Bovik, "Image quality assessment: from error visibility to structural similarity", *IEEE Transactions on Image Processing,* Pages 600–612, April 2004.

[18] C. Konstantopoulos, A. Svolos, C. Kaklamanis, "Hierarchical Block Matching Motion Estimation on a Hypercube Multiprocessor", *4th International ACPC Conference Including Special Tracks on Parallel Numerics*

and Parallel Computing in Image Processing, Volume 1557/1999, Pages 265-275, 1999.

[19] C. Zhu, X. Lin, L.P. Chau, "Hexagon Based Search Pattern for Fast Block Motion Estimation", *IEEE Transactions and Circuits and Systems for Video Technology*, Vol. 12, No. 5, Pages 349-355, May 2002

[20] A.M. Huang, T. Nguyen, "Correlation-Based Motion Vector Processing with Adaptive Interpolation Scheme for Motion-Compensated Frame Interpolation", *IEEE Transactions on Image Processing*, Vol. 18, No. 4, Pages 740-752, April 2009.

[22] A.M. Huang, T. Nguyen, "Motion Vector Processing Based On Trajectory Curve Analysis For Motion-Compensated Frame Interpolation", *IEEE International Conference on Image Processing (ICIP2009)*, Pages 377-380, Nov. 2009.

[23] E. Turetken, A. Alatan, "Region Based Motion-Compansated Frame Rate Up-Conversion by Homography Parameter Interpolation", *IEEE International Conference on Image Processing (ICIP2009)*, Pages 381-384, Nov. 2009.

[24] L. Zhang, C. Wang, W. Zhang, Y. Tan, "Frame Rate Up-Conversion with Edge Weighted Motion Estimation and Trilateral Interpolation", *IEEE International Conference on Circuits and Systems (ISCAS 2009)*, Pages 1637-1640, May. 2009.

[25] B. Choi, J. Han., C. Kim., S. Ko, "Motion Compensated Frame Interpolation Using Bilateral Motion Estimation and Adaptive Overlapped Block Motion Compensation", *IEEE Transaction on Circuits and Systems For Video Technology*, Vol.17, No.4, Pages 886-893, April 2007.

[26] A. McIvor, Q. Zang, Reinhard Klette, "The Background Subtraction Problem for Video Surveillance Systems", *Lecture Notes in Computer Science*, 2001, Volume 1998/2001, Pages 176-183.

CURRICULUM VITAE

Candidate's full name:	Mehmet Mutlu ÇEKİÇ
Place and date of birth:	ANKARA / 04.02.1986
Permanent Address:	Mecidiyeköy Mah. Eski Osmanlı Sok. No:33/7 Şişli/İstanbul
Universities and Colleges attended:	M.Sc. İstanbul Technical University
	B.Sc. Işık University
	Yozgat Science High School