ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

AN ENHANCED COMPRESSION TECHNIQUE USING IMPROVED MOTION COMPENSATION

M.Sc. THESIS

Kevser BOZOĞLU

Department of Electronics and Communication Engineering

Telecommunication Engineering Programme

June 2012

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

AN ENHANCED COMPRESSION TECHNIQUE USING IMPROVED MOTION COMPENSATION

M.Sc. THESIS

Kevser BOZOĞLU
504091370

Department of Electronics and Communication Engineering

Telecommunication Engineering Program

Thesis Advisor: Prof. Dr. Melih PAZARCI

June 2012

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GELİŞMİŞ HAREKET KESTİRİMİ KULLANARAK VIDEO SIKIŞTIRMA İYİLEŞTİRMESİ

YÜKSEK LİSANS TEZİ

Kevser BOZOĞLU
504091370

Elektronik ve Haberleşme Mühendisliği Anabilimdalı

Telekomünikasyon Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Melih PAZARCI

Haziran 2012

Kevser Bozoğlu, a **M.Sc.** student of ITU **Graduate School of Science, Engineering and Technology** student ID 504091370, successfully defended the **thesis** entitled "**An Enhanced Compression Technıque Using Improved Motion Compensation**", which she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Melih PAZARCI** İstanbul Technical University	.....
Jury Members :	**Prof. Dr. Melih PAZARCI** İstanbul Technical University	.....
	Prof. Dr. Bilge GÜNSEL İstanbul Technical University	.....
	Asst. Prof. Hülya YALÇIN İstanbul Technical University	.....

Date of Submission : 04 May 2012
Date of Defense : 08 June 2012

FOREWORD

I wish to thank, first, Prof. Dr. Melih Pazarcı for his guidance, attention, insight, and patience during this work. Without his continuous support, encouragement and constructive criticisms, it would not be possible for me to complete this thesis.

I want to express my respects to Prof. Dr. Ahmet Dervişoğlu and Inst. Deniz Pazarcı for teaching me at Yeditepe Univesity and also thank them for their encouragement to start my master education.

Special thanks are due to my friends Ayşe Akdoğan and Özge Irmak with whom courses were smooth and enjoyable.

Financial support in the form of MSc scholarship from TUBITAK-BIDEB is sincerely acknowledged.

Above all, I am especially indepted to my parents Ali-Hanife Bozoğlu for their love, generous support during my whole life and also for their continuous effort to my education.

June, 2012 Kevser BOZOĞLU

TABLE OF CONTENTS

Page

ABBREVIATIONS

AVC	**:** Advanced Video Coding
BMA	**:** Block Matching Algorithm
FPS	**:** Frames per Second
FSBMA	**:** Full Search BMA
GOP	**:** Group of Pictures
IMC	**:** Improved Motion Compensation
ISO	**:** International Organization for Standardization
ITU	**:** International Telecommunication Union
MPEG	**:** Motion Picture Experts Group
MSE	**:** Mean Squared Error
PSNR	**:** Peak Signal to Noise Ratio
SAD	**:** Sum of Absolute Differences
SIFT	**:** Scale Invariant Feature Transform

LIST OF TABLES

Page

LIST OF FIGURES

Page

AN ENHANCED COMPRESSION TECHNIQUE USING IMPROVED MOTION COMPENSATION

SUMMARY

In the digital age, almost all video storage and delivery systems depend heavily on compression technology. Video compression standards have been used and improved since 90s. There are many standards that use motion compensation to reduce unnecessary data and compress video. This thesis proposes an improved compression technique based on both block matching compensation and SIFT. This new technique reveals an improved motion compensation (IMC) which is achieved by using invariant features.

SIFT finds local invariant features in a given image. It has been used in many implementations in the literature. However, none of them used invariant features to obtain a better motion compensation. In this thesis, SIFT is used to detect rotated regions and obtain better reconstruction with higher PSNR than classical block matching.

To achieve this, invariant features are found in adjacent frames and then feature keypoints are matched. In conventional BMA, each non-overlapping 16x16-pixels block in current frame is predicted from previous frame by finding the block that gives minimum SAD value. However, if an object rotates from previous frame to the current, BMA cannot achieve an accurate prediction. In this case, the algorithm implemented in this thesis comes up with more accurate results because SIFT keypoints are utilized to detect and correct rotation between frames in the implemented algorithm.

The proposed algorithm is tested on both test frames and real video frames. PSNR results are given for comparison of the proposed algorithm with classical BMA. It is seen that the proposed algorithm, which is called IMC in this thesis, caused an improvement of almost ~1.4 dB for the reconstructed test frame. This improvement is approximately 0.5 dB for real video frames. PSNR values are calculated for whole images. To see the block improvements, SAD values of blocks for compared algorithms are exhibited. These improvements may also be visually observed in the given reconstructed frame pictures.

GELİŞMİŞ HAREKET KESTİRİMİ KULLANARAK VİDEO SIKIŞTIRMA İYİLEŞTİRMESİ

ÖZET

Bulunduğumuz dijital çağda, hemen hemen tüm video depolama ve iletim sistemlerinin temeli veriyi sıkıştırma teknolojisine dayanır. Dijital videoyu doğrudan saklamak ya da iletmek çok yüksek veri kapasitesi gerektirir. Bu nedenle, dijital video saklanacaksa veya iletilecekse öncesinde sıkıştırılır. Video sıkışıtırma standartları 90lı yıllardan beri kullanılmakta ve sürekli geliştirilmektedir.

İlk video sıkıştırma standartı 1990 yılında uygulanan ITU H.261 standartı olarak kabul edilebilir. ISO ve ITU organizasyonlarının ortaklaşa onayladığı MPEG-2/H-262 standartı 1993 yılında kullanılmaya başlanmıştır ve günümüzde de yaygın olarak kullanılmaktadır. 1995 yılında, ITU H.263 standartını geliştirmiştir. Günümüzün video konferans sistemleri ve cep telefonu kodeklerinde de çoğunlukla H.263 standartı mevcuttur [14]. Son sıkıştırma algoritması, ITU ile ISO'nun birlikte geliştirdikleri H.264/MPEG-4 Part10/AVC standartıdır. Bu son standart, geliştirirlmiş sıkıştırma kalitesi sayesinde MPEG-2'ye gore zaman zaman %50 ye kadar daha az bitle iletim sağlayabilir. Başka bir deyişle, H.264 standartı bir videoyu önceki standartlarla aynı kalitede ancak onlardan iki kat daha fazla sıkıştırabilir. Sonuç olarak H.264, 21. Yüzyılın en gelişmiş video kodlama standartı olarak görülmektedir [10].

Video sıkıştırma işlemi genel olarak, video çerçevelerini (frame) depolamak veya iletmek için ihtiyaç duyulan bit sayısını azaltmayı hedefler. Video sıkıştırma standartları bilgiyi/veriyi azaltmak için farklı metodlar kullanırlar. Bu sebeple, her bir standartın bit oranı ve veri kalitesi de farklıdır. Yukarıda bahsedilen tüm standartlar veriyi kodlarken hareket tahmini (*motion estimation*) tekniklerini kullanırlar. En yaygın kullanılan hareket tahmini tekniği, blok eşleme algotritması (*block matching algorithm*) tekniğidir. Bu teknikte video çerçeveleri belirli sayıda pixellerden oluşmuş bloklara ayrılır. Şimdiki video çerçevesi içindeki her bir blok, önceki çerçeveden aynı büyüklükte ve kendisine en çok benzeyen bir blokla eşleştirilir. Bu şekilde eşleşen bloklar tahmin edilen (*predicted*) çerçeveyi oluşturur. Ancak, klasik blok eşleme algoritması, video süresince dönen bir cisim olması durumunda çok iyi sonuçlar verememektedir.

Blok eşleme algoritmasında çeşitli blok arama stratejileri kullanılabilir. Bunlardan bazıları kapsamlı arama (*exhaustive or full search*), hızlı arama (*fast search*), çapraz arama (*cross search*) olarak adlandırılabilir. Bu tez kapsamlı blok eşleme algoritmasını temel alacak şekilde geliştirilmiş ve bu algoritmanın iyileştirilmesi amaçlanmıştır. Kapsamlı blok eşleme algoritmasında blok ve pencere boyutu kullanıcı tarafından belirlenebilmekle birlikte, bu boyutlar algoritmayı şu şekilde etkilerler: Daha büyük blok boyları, genelde gürültüye karşı daha dayanıklıyken, küçük bloklar daha belirgin çizgiler yakalayabilirler. Doğru eşlemeyi bulmak içinse

arama penceresi boyutunu doğru belirlemek önemlidir. Pencre boyutu büyüdükçe doğru eşleme olasılığı artarken, algoritmanın hesap yükü ve hesaplama için geçen zaman da artmaktadır. Genelde kullanılan blok boyutları 4x4, 8x8 veya 16x16 piksel olmakla beraber en yaygın kullanılan blok boyutu 16x16 pikseldir. Arama penceresi de genelde bloğun etrafında sağ, sol, üst ve altta 7şer pixel genişlik sağlayacak şekilde seçilir.

Literatür tarama sürecinde, SIFT algoritması ve algoritmanın hangi alanlarda kullanıldığı incelenmiştir. SIFT algoritması, input olarak kullanılan bir resmin kilit-nokta tanımlayıcılarını (key-point descriptors) bulur. Her bir kilit-nokta, resmin rotasyondan, ölçeklemeden ve gürültüden bağımsız bir özelliğini oluşturur. Bir kilit nokta ise 128 elemandan oluşan bir vektörle tanımlanır. Her bir vektörün eleman sayısının bu kadar fazla olmasının sebebi, her bir kilit-noktayı diğerlerinden ayıracak özelliklerin kendini tanımlayan vektörde saklanıyor olmasıdır.

SIFT algoritması ilk olarak [2]'de daha sonra ise [3]'te anlatılmıştır. Hareket tahmini ve takibi, robot lokalizasyon ve haritalama, resimlerin panaromik birleştirilmesi, üç boyutlu (3D) sahne modellemesi gibi bir çok alanda başarılı uygulamalarının olduğu [5]'te belirtilmiştir. Ayrıca, [11]'de bir resimde kopyalanmış bölgenin belirlenmesi, [12]'de bahsedilen kamera affine model parametrelerinin hesaplanması, [13]'te sunulan çok görüşlü videolarda daha doğru yan bilgilerin bulunması gibi başarımlar SIFT kilit-nolkaları ile elde edilmiştir. Tüm bu araştırmalar sonucunda SIFT'in daha önce blok eşleme algoritmasında geliştirmek için kullanıldığına rastlanmamıştır. Bu tezin amacı, Ölçekten Bağımsız Nitelikler Transformu'ndan (SIFT) yararlanarak daha iyi bir hareket kestirim algoritması elde etmektir.

[4]'te patenti Lowe tarafından alınan SIFT algoritmasının internette farklı parametreler içeren bir çok gerçeklemesi bulunmaktadır. Bu tezde [6]'da verilmiş olan aynı zamanda [3] ve [4]'te açıklanan gerçekleme kullanılmıştır.

Önçalışma amacıyla, [6]'da verilen gerçekleme kullanılarak bir videoya ait ardarda gelen iki çerçevenin kilit-noktları bulunmuştur. SIFT algoriması çıktı olarak kilit-noktaların x ve y koordinatlarını, büyüklüğünü ve oryantasyon değerlerini vermektedir. Yine [6]'da bulunan eşleme algoritması ile bu çerçevelerdeki kilit-noktalar eşlenmiştir. Böylece kilit-noktalara ait hareket vektörleri de hesaplanıp çizdirilmiştir. Sonuçta her bir blok (16x16 pixel) içinde en az 0, en fazla 5 eşleşen kilit-nokta olabildiği görülmüştür.

Tezde önerilen algoritma, blok eşleme algoritmasının iyi sonuç vermediği yerlerde devreye girer. Eğer video içinde rotasyon hareketi yapan bir obje varsa blok eşleme algoritması iyi sonuç vermezken, tezde önerilen algoritma rotasyondan etkilenen blokları belirleyerek bu bloklara ters rotasyon uygulayıp tahmin edilen resme bu şekilde yerleştirmeyği hedefler.

Bunu başarmak için kullanılan yöntem şu şekildedir: Önceki resimden alınan bloğun şimdiki resme dönüşüm matrisi, eşleşen ve kolineer olmayan kilit-noktalar kullanılarak bulunabilir. Bir blokta eşleşen nokta sayısı arttıkça daha karmaşık dönüşüm matrisleri hesaplanabilir. Genelde bir blokta en fazla dört ya da beş eşleşen nokta bulunsa da bu noktalar bazen kolineer olabiliyor ya da bazı noktların büyüklük ve oryantasyonları farklı olsa da aynı lokasyonda bulunabiliyorlar. Bu durumda algoritmada iki veya üç kolineer olmayan kilit-nokta kullanarak belirlenebilen iki boyutlu (*2D*) dönüşümlere odaklanılmıştır.

Bu aşamada, bir video içerisinde ardarda gelen çerçevelerdeki kilit-noktlar bulunduktan sonra [6]'da verilen eşleme algoritması amaca yönelik değiştirilmiştir. Eşleme algoritmasının girdileri şimdiki çerçeve ile referans çerçevedir. Çıktıları ise şimdiki çerçeve içindeki her bir 16x16 blok için hesaplanmış makro-blok tablolardır. Bu tablolarda, eşleşmiş ve kolineer olmayan kilit-nokta sayısı, her bir blok içindeki kilit noktaların bulunduğu satır ve sütun bilgileri, bu kilit-noktlarla eşleşen referans çerçevedeki kilit-noktaların satır ve sütun bilgileri ve her bir eşleşen noktanın rotasyon bilgisi tutulmaktadır.

Tüm bu veriler saklandıktan sonra, klit-noktaların yer (koordinat) bilgileri *cp2tform* fonksiyonunun girdileri olarak kullanılırlar. *cp2tform* fonksiyonu MATLAB'in hazır fonksiyonudur ve çift kontrol noktalarından özel dönüşümü hesaplar. Bu fonksiyonun girdileri şimdiki çerçevenin kilit-nokta koordinatları ile bu noktalarla eşleşmiş olan referans çerçevenin koordinat bilgileridir. Nokta koordinat bilgilerinin yanında girdi olarak *transformtype* parametresine de gerek duyulmaktadır. *Transformtype*, fonksiyonun kontrol noktalarından bulacağı transformun türüdür ve her transform için gereken kontrol nokta sayısı farklıdır. Bizim algoritmamızda *transformtype*'ı belirleyen klit-nokta sayısıdır ve kilit-nokta sayısı (*point_num*) bize üç duruma; üç farklı transforma ulaşma olanağı sunar:

İlk durumda, bir blok içinde en az üç ve daha fazla kilit-nokta vardır ve bu sayı *affine* transform için gerek ve yeter koşul olduğundan transformtype'ı affine olarak belirleyerek affine dönüşüm matrisini bulabiliriz. İkinci durum blok çinde eşleşen iki kilit-nokta olmasıdır. İki nokta affine dönüşümü belirlemek için yeterli olmadığından burada transformtype parametresi "*nonreflective similarity*" olarak verilir. Bu dönüşüm türünde rotasyon, kayma ve ölçekleme gibi parametreler belirlenebilirken şekil değişimleri belirlenemez. Bu sebeple tezin amacı göz önünde bulundurulursa affine dönüşüme gore rotasyon belirleme açısından bir kaybı yoktur. Son olarak, blok içinde eşleşen tek bir nokta olması durumundaysa, bir nokta herhangi bir dönüşüm matrisini çözmek için kullanılamayacağından dolayı, bu nokta çeversine açı farkı kadar rotasyon uygulanır.

Uygulanan gerekli dönüşümden sonra, yeni yöntem kullanarak kestirilen blok ile klasik blok eşleme yöntemi sonucunda kestirilen blok karşılaştırılır ve optimum sonucu hangi algoritma verdiyse o algoritmanın öngördüğü blok kestirilen resim çerçevesine atanır.

Sonuç olarak önerilen algoritma hem test resimlerinde hem de gerçek video resimler üzerinde test edilmiştir. Testler sonucunda, BMA ile karşılaştırma yapmak için elde edilen resimlerin PSNR değerleri hesaplanmıştır. Elde edilen verilere gore, önerilen algoritmanın sonuçları BMA sonuçları ile karşılaştırıldığında test frameler üzerinde ~1.4 dB'ye varan, gerçek video resimleri üzerinde ise 0.5 dB'ye varan iyileştirmeler sağladığı gözlemlenmiştir. Bu iyileştirmeler, sonuçlar kısmındaki şekillerden de görsel olarak gözlemlenebilir.

1. INTRODUCTION

Directly storing or transmitting digital video generally requires very high data capacity. Compression is needed for economical storage and/or transmission. Video compression has the goal of reducing the number of bits required to store or convey video frames. It can be said that digital video compression started with the ITU H.261 standard in 1990. The MPEG-2/H-262 standard was approved by ISO and ITU standardization organizations in 1993 and widely used today. In 1995, ITU developed the H.263 standard, which is dominantly used in today's video conferencing and cell phone codecs [14]. H.264/MPEG-4 Part10/AVC is the latest compression standard developed by the ITU and ISO organizations together. Because of its improved compression quality, H.264 can reach %50 additional bit rate savings over MPEG-2. In other words, it can compress video at twice the rate of previous video standards while achieving the same quality. H.264 standard has become the most successful video coding standard of the twenty-first century [10].

All these video compression standards use different methods to reduce data and this is why the bit rate and quality is different for each standard. Standards mentioned here use motion compensation techniques while encoding data. Most widely used motion compensation technique is based on block motion compensation in which the frames are divided into blocks of pixels and then predicted by matching each block in current frame with a block of equal size in the previous frame.

1.1 Purpose of the Thesis

The aim of this thesis is to express a new motion compensation technique by using Scale Invariant Feature Transform (SIFT) which is a local feature technique. Before starting the details on this subject, the areas that use the SIFT algorithm are investigated. This technique is shown to be successful in many fields such as object recognition and tracking, robot localization and mapping, panorama stitching, 3D scene modeling etc. [5]. In [11], a new region duplication detection method based on the image SIFT features is presented. The SIFT is applied to global motion

estimation in [12] and parameters of the camera affine model are computed. In [13], feature keypoints are used to yield more accurate side information for multi-view videos. At this point, there is no study that aims to improve full search BMA by using SIFT. Conventional full search BMA cannot handle cases when a region has rotated in a video. This thesis aims to detect this kind of transformed regions and obtain a better reconstruction with higher PSNR by using the invariance feature of matched keypoints.

1.2 Thesis Organization

This thesis is organized as follows. In Chapter 2, interframe video compression techniques are introduced and motion compensation is explained. BMA, which is a motion estimation search technique, is described in detail because it is used as a base technique in this thesis. Unsuccessful conditions for BMA are discussed and finally two BMA examples are exhibited with their results. SIFT which is the local feature technique is introduced in Chapter 3. After a brief introduction to SIFT, the steps that are executed until keypoint descriptors are obtained have been described. Then, use of SIFT in image matching and motion estimation are explained. Chapter 4 includes information about the development environment and the implementation of the two new algorithms. In other words, improved BMA using SIFT features is explained in detail. Results are given and discussed in Chapter 5. Finally, in Chapter 6, some conclusions are drawn and some recommendations for future work are given.

2. INTERFRAME VIDEO COMPRESSION TECHNIQUES

Video compression aims to reduce redundancy in a video signal by using coding techniques. The redundancy in a video signal is based on two principles. The first one is the spatial redundancy that exists in each frame. The second one is temporal redundancy, which is the fact that, a video frame is very similar to its adjacent frames. Eliminating spatial redundancy is called intra-frame compression. Reducing temporal redundancy is called inter-frame compression. Video compression reduces the number of bits required to store and transmit digital video. Therefore, compressed video can be transmitted more economically over a smaller bandwidth.

Inter-frame compression works by comparing current frame with the previous one. If current frame contains exactly the same areas with the previous one, the system copies this part from the previous frame. If an objects move, the system sends the direction and quantity of movement information, and then these areas are copied with movement compensation.

In video compression, motion estimation and motion compensation techniques are widely used for describing a frame in terms of copied and translated parts of a reference frame. These copied and translated parts are generally 4x4, 8x8 or 16x16 pixels.

2.1 Motion Compensation

In video sequences, there exists a high level of redundancy between consecutive frames. Changes are generally very little from one frame to the next. Therefore, temporal redundancy reduction is achieved by encoding first a reference frame then encoding only the difference between the reference frame and the target frame for consecutive frames.

There are some different redundancy reduction techniques such as differential coding and block-based motion compensation. Differential coding is used by most video compression standards including H.264. In differential coding, a frame is compared

with a reference frame and only the pixels that have changed with respect to the reference frame are coded. A better technique is block-based motion estimation, which gives better results especially if the motion rate is high.

This thesis is based on block-based motion compensation and a new implemented approach is compared with this technique. Figure 2.1 shows the flow chart of block-based motion compensation process. The block matching box in this chart is explained in detail in Chapter 2.2.

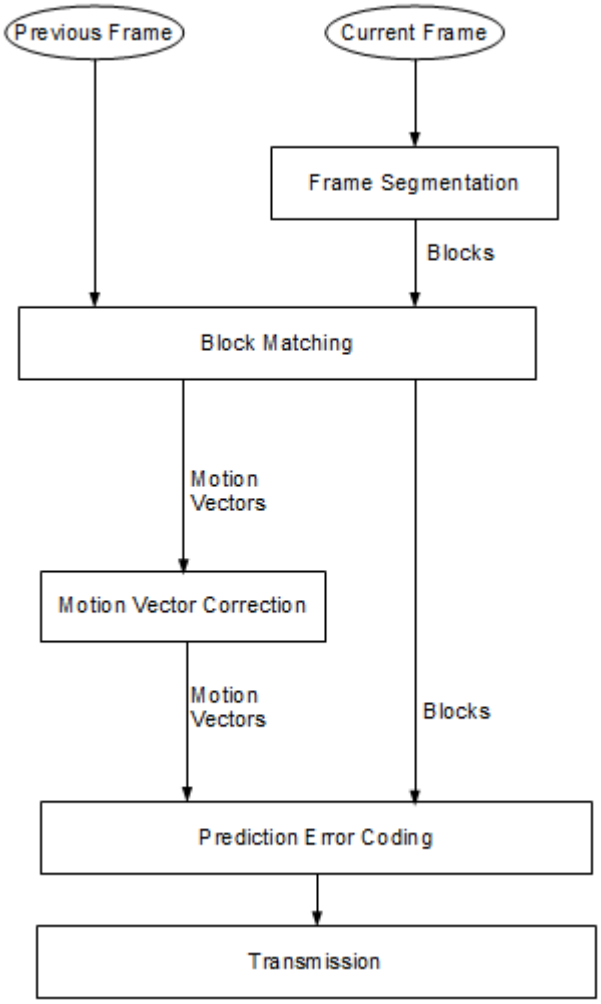

Figure 2.1 : Flow of information through motion compensation process.

2.2 Block Matching Algorithm (BMA)

Block matching algorithm aims to detect the motion of non-overlapping blocks. Each block from the current frame is matched with a block in the previous frame. This matching is achieved by shifting the current block over a predefined region of pixels in the reference frame. At each shift position, the sum of absolute differences (SAD) between the gray values of the two blocks are computed as shown in **(2.1)**. The shift

that gives the lowest SAD is accepted as the best match. SAD calculation is given in **(2.1).**

$$SAD(i,j) = \sum \left| I^{t+1}(i,j) - I^{t}(i + v_x,\ j + v_y) \right| \qquad \textbf{(2.1)}$$

Here, I^{t+1} is the current frame and I^{t} is the previous frame. (i, j) is the beginning location of the block for which SAD is computed. v_x, v_y are x and y components of the motion vector for that block respectively. Utilized v_x and v_y values give the location where minimum SAD is calculated. Therefore, they generate the motion vector for that block.

For block matching algorithm, there are various search strategies such as full (exhaustive) search, fast search, three step search, cross search etc. Full search algorithm is selected as the reference algorithm for this thesis. Block size and search window can be determined as inputs to the full search algorithm. In general, bigger blocks are less sensitive to noise while smaller blocks produce better contours [1]. For this thesis, 16x16 blocks are used for both the reference and proposed algorithms. The size of the search window is important to find the true match. The computation load and time increase with the growth of the search window.

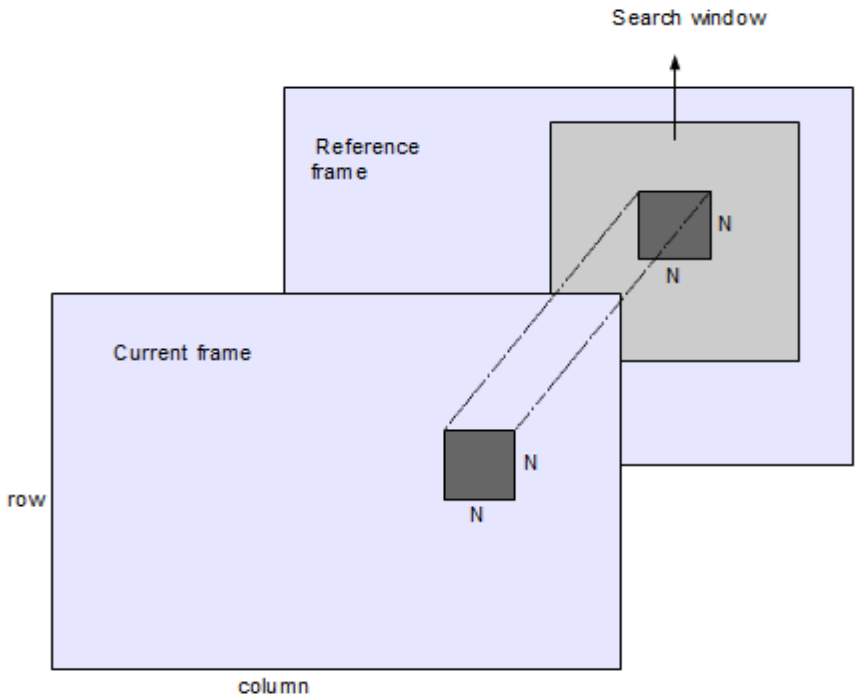

Figure 2.2 : Basic elements of Full Search BMA.

Flow chart of the the FSBMA is shown in the Figure 2.3 below. However, there are some cases that change these implementation steps. One of them is the fact that, in

some cases the true motion vector may address pixels outside the frame. If this situation is detected, this motion vector is neglected and algorithm continues with the next motion vector. The other case is when the frame size is not a multiple of block size. This situation occurs generally at the edges of the image but it is not available in this thesis because all video frames are a multiple of 16x16. Therefore, it is not a drawback for this thesis.

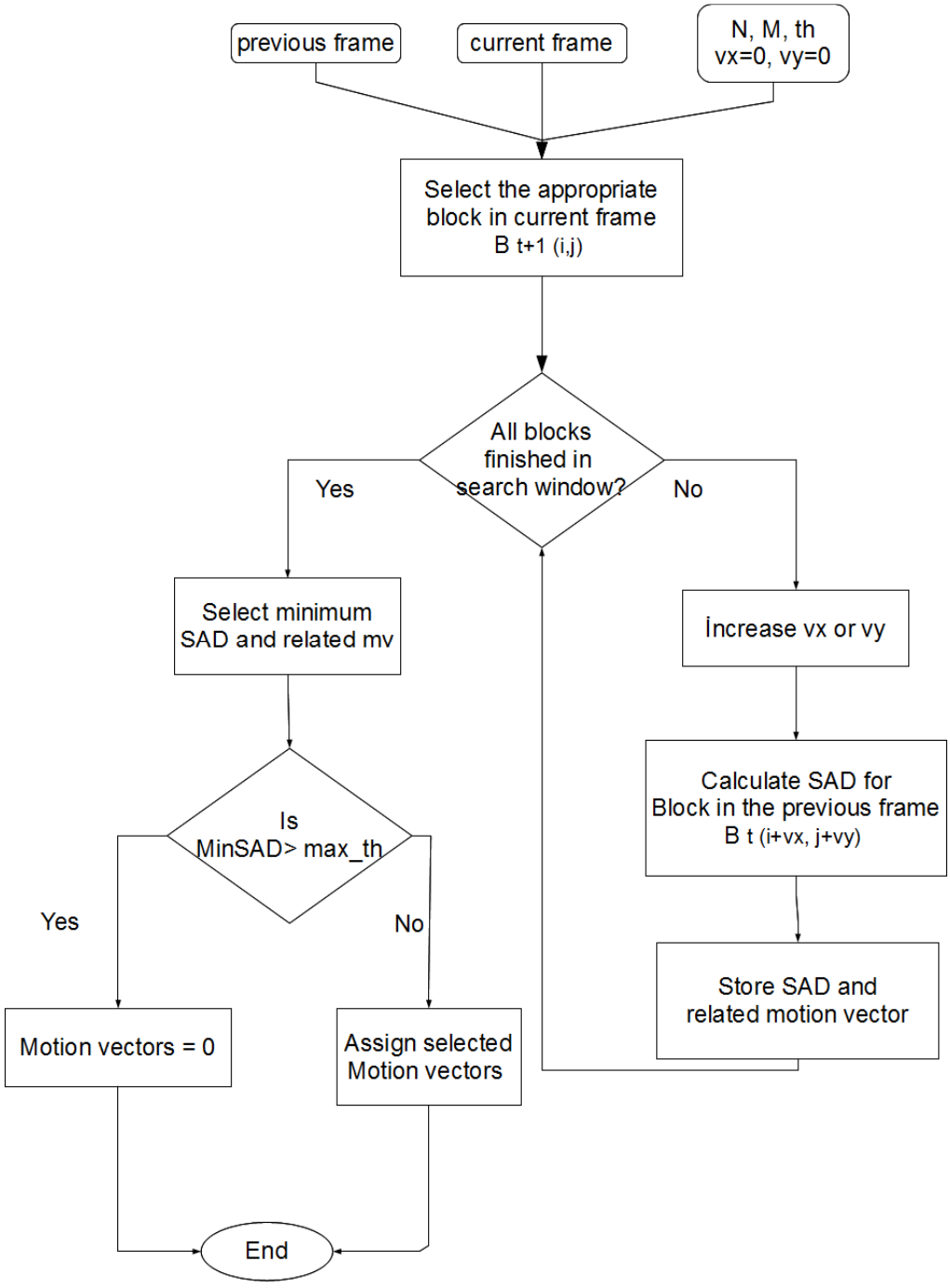

Figure 2.3 : Flow chart of BMA.

2.2.1 Video quality measure

The peak signal to noise ratio (PSNR) in decibel (dB) is widely used as a quality measure in video coding [1]. The PSNR is defined as

$$PSNR = 10\ log_{10} \frac{\psi_{max}^2}{MSE} \qquad (\mathbf{2.2})$$

Where ψ_{max} is the peak (maximum) intensity value of the video signal and mean square error (MSE) between two images (im1 and im2) is calculated as

$$MSE = \frac{1}{row \cdot col} \sum (im1 - im2)^2 \qquad (\mathbf{2.3})$$

Here, row and column represent the row length and column length of the image, respectively.

As it is noted in [1]; a PSNR over 40 dB typically indicates an excellent image (being very close to the original), between 30 to 40 dB usually means a good image (distortion is visible but acceptable), between 20 and 30 dB is quite poor and finally, a PSNR lower than 20 dB is unacceptable.

2.2.2 Unsuccessful conditions of BMA

Block Matching Algorithm cannot find the true match when an object is occluded between frames, when there is an aperture problem or if there is a rotation/affine transformation in the image.

The last problem can be solved by checking the blocks if there is a rotation or transformation. When it is detected, applying a special algorithm for this block will improve the motion estimation result. In this thesis, this idea is implemented by using the Scale Invariant Feature Transform (SIFT) which is explained in next chapter.

2.2.3 BMA results

To observe BMA results, *blockmatching* function is written in MATLAB. Inputs of *blockmatching* function are previous frame (*im_pre*), current frame *(im_cur),* block size (*N*), search window size (*M*) and threshold (*th*). The outputs of the function are motion vector parameters (*Vx*, *Vy*) and predicted image (*predicted*). The use of blockmatching function is as follows:

[predicted, Vx, Vy] = blockmatching(im_pre, im_cur, N, M,th);

Inputs and outputs of *blockmatching* function are observed in two examples below.

In these examples, M=16, N=16, th=1.

Example 1: foreman video

a)  b) 

Figure 2.4 : a)Reference image, b) Current image (right).

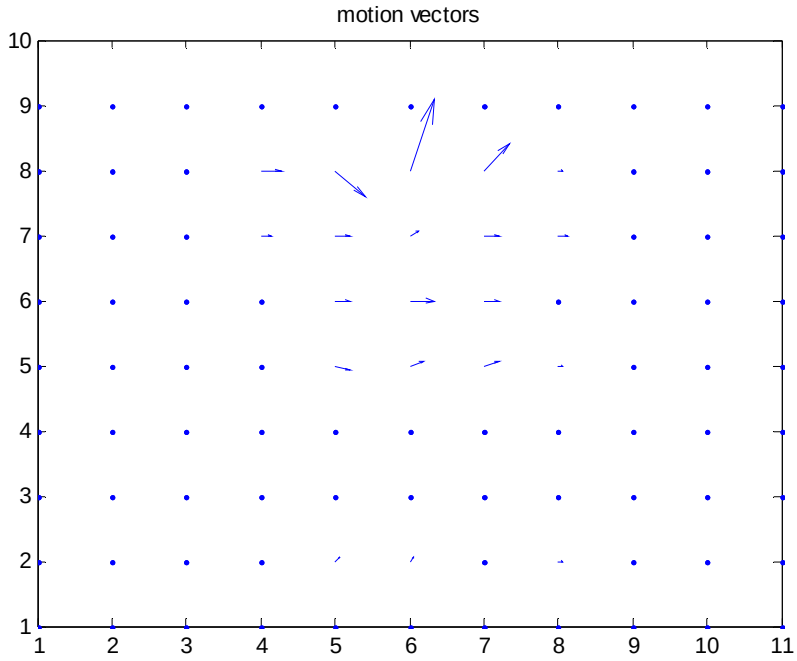

Figure 2.5 : Motion vectors for Example 1

Figure 2.6 : Predicted image for the target frame (PSNR = 38.7828)

Example 2:

a)

b)

Figure 2.7 : a) Reference image, b) Current image.

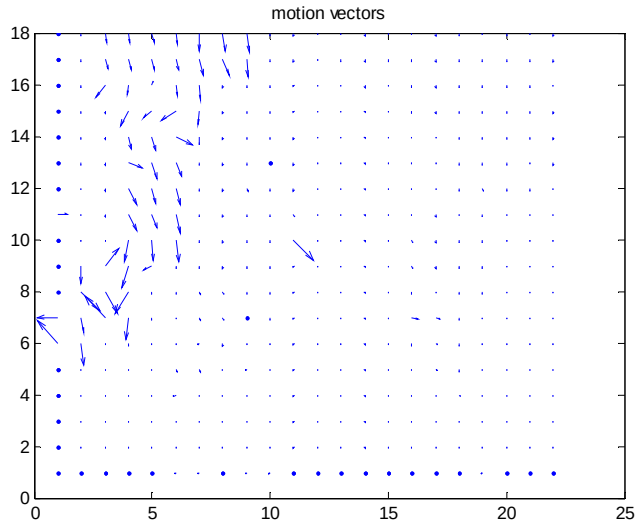

Figure 2.8 : Motion vectors for Example 2

Figure 2.9 : Predicted image for the current frame (PSNR = 33.7365)

3. SIFT AND MOTION ESTIMATION

The Scale Invariant Feature Transform (SIFT) algorithm assigns features of an image which are distinct, and these features can be used to identify similar objects even if they are rotated, scaled or brightened in other images.

The SIFT was first published in [2], later in [3] and is also patented in [4]. However there is not just one implementation, there are many implementations with different parameters on the internet due to the fact that distinctive features for an image are not unique. Implementation in [6] which is also explained in [3] and [4] is used in this thesis.

3.1 Sift Algorithm

SIFT takes an image as an input and generates a set of key-point descriptors. A keypoint is an image feature which is so stable that rotation, scaling or noise cannot distort the keypoint. A descriptor is a 128 dimensional vector array that describes a keypoint. The reason for this high dimension is that, each descriptor should include a lot of information to distinguish this keypoint from others.

The SIFT algorithm can be divided into five computational steps:

3.1.1 Step 1: Scale Space and DoG Image

In the first step of the SIFT algorithm, the input image is blurred by applying a one dimentional Gaussian function $G(x,\sigma)$, given in **(3.1)** to the vertical and horizontal directions.

$$G(x,\sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-x^2/2\sigma^2} \quad \textbf{(3.1)}$$

In [4], σ is selected as $\sqrt{2}$ and it is implemented by using 7 samples of $G(x,\sigma)$.

After calculating the blurred image, the Difference-of-Gaussian (DoG) image is obtained by subtracting the blurred image from the input image. Then the resolution of the DoG image is checked. If it is less than a predefined resolution, which is 30x30 pixels in [4], this step is ended. If the DoG image resolution is higher than the predefined resolution, the DoG image is down-sampled (1.5 times in [4]). This lower resolution image is then used as the input image so it is blurred again. This flow is repeated until DoG image which has less than 30x30 pixels is obtained. The flow chart of the algorithm is given in Figure 3.1.

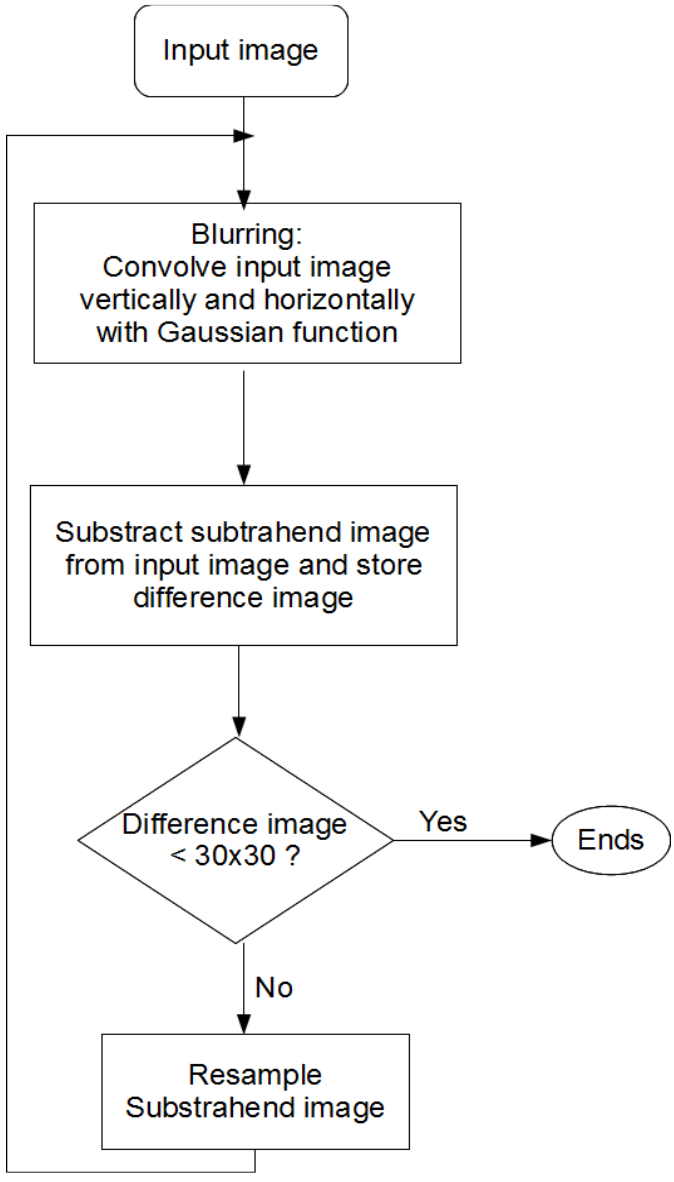

Figure 3.1 : Finding Scale Space and DoG[4]

Here, input image and blurred image form an octave set. In [4], an octave set consists of two images (one input and one blurred). Generally, to obtain an octave set, input image is blurred progressively, and more than one blurred image is produced as in

[3]. In addition, the number of octaves are determined by two parameters: resolution of the first image and the predefined DoG resolution. Figure 3.2 shows the DoG pyramid as implemented in [3].

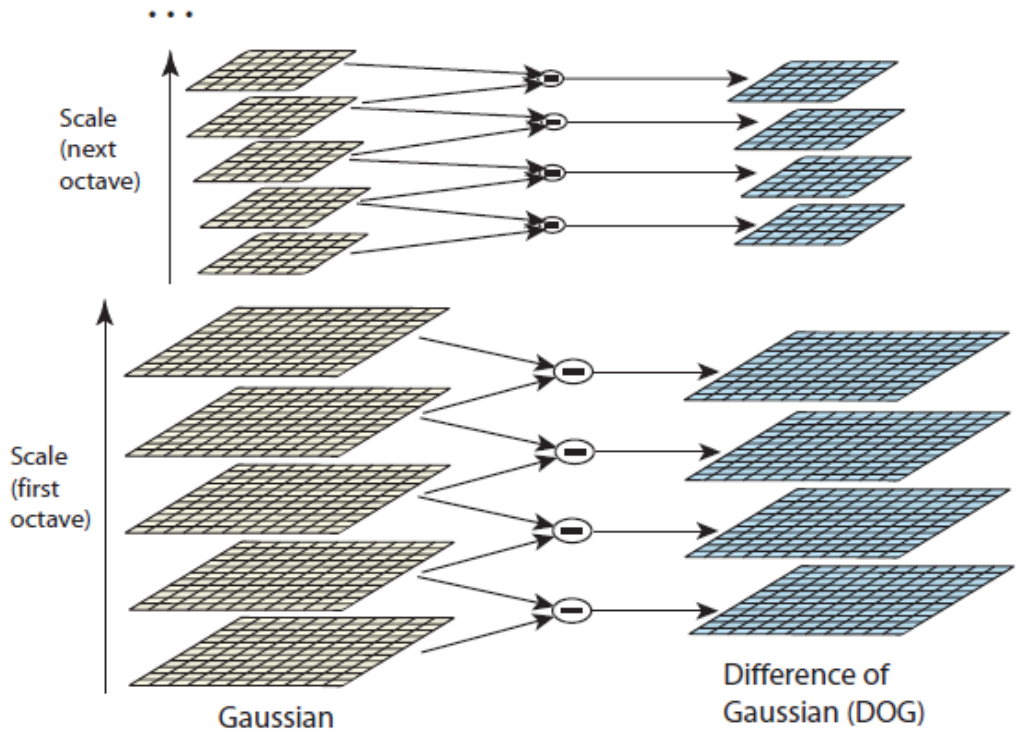

Figure 3.2 : Scale Space and DoG

3.1.2 Step 2: Maxima and Minima (Extrema) Detection

In Step 1, DoG images which have different resolutions are obtained. In this step, pixel amplitude extrema is going to be located in each DoG image with the algorithm that is shown in Figure 3.3. An extrema can be defined as any value in a diference image greater than all its neighbours in scale space.

In this algorithm, each pixel of each difference image is iterated and checked with all its neigbours at the same level, and it is also checked with previous and next levels as depicted in Figure 3.4. Cross signed pixel is the compared pixel and green colored pixels are the neighbours of it. As a result, this check is done with totally 26 points (8 same level, 9 previous level, 9 next level) for one iteration.

These found extremas are not exact extremas because they do not have to be on a pixel, they may be on somewhere between pixels.

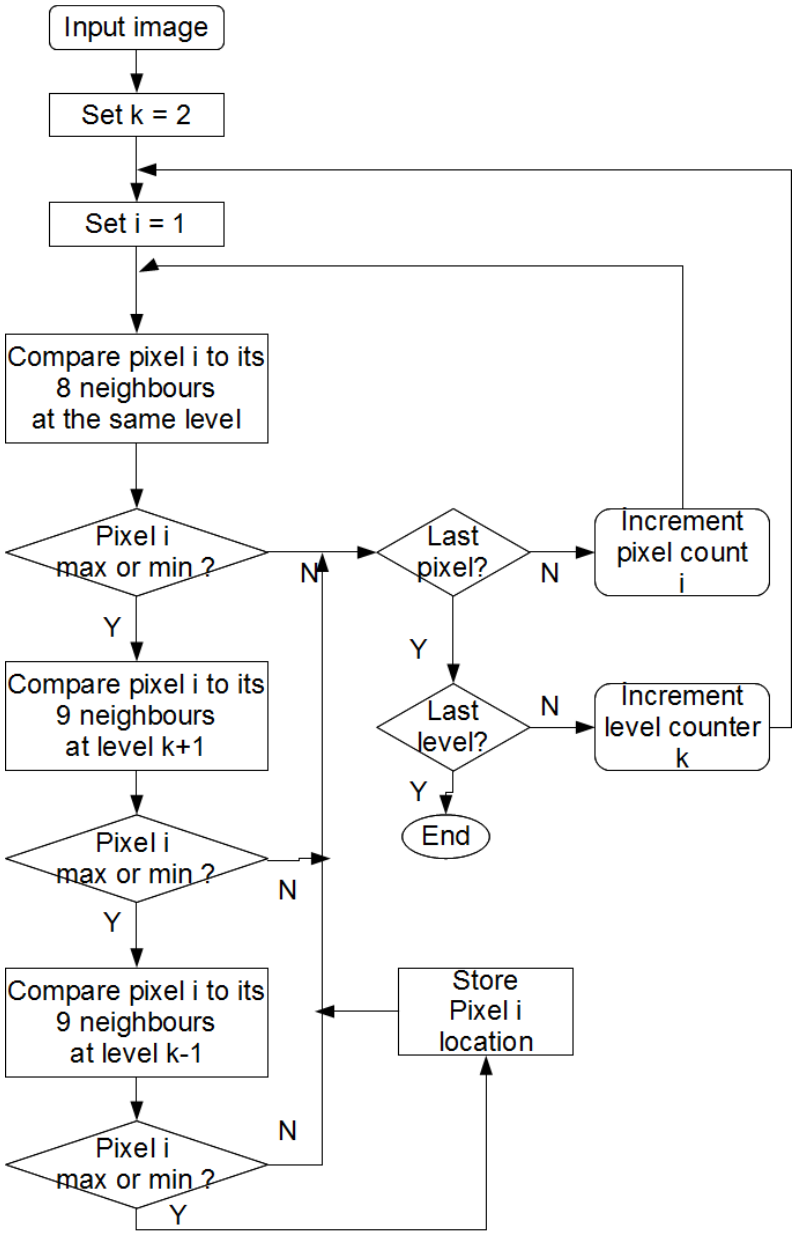

Figure 3.3 : Extrema Detection Algorithm [4]

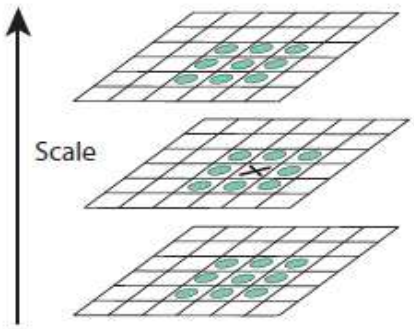

Figure 3.4 : Extrema detection in DoG Images [3]

To access data between pixels, in [3], localization of the keypoint is improved to subpixel acuracy by using a second order Taylor series expansion of scale space function $D(x, y, \sigma)$:

$$D(x) = D + \frac{\partial D^T}{\partial x^2} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \qquad (\mathbf{3.2})$$

The derivative of this function with respect to x is taken and set to zero. This gives the true extrema location [3].

3.1.3 Step 3: Keypoints Elimination

In this step, some weak points are eliminated from the candidate list of keypoints. These weak keypoints have one of two specific features which are low contrast and being located on an edge.

Candidate keypoints with a value below a threshold are discarded in order to reject unstable extrema with low contrast. For example, it is assumed that pixel values are in the range [0,1] and an extrema with a value of less than 0.03 are eliminated in [3].

To eliminage keypoints on edges, the situation used in [3] that, the exixstance of a large principle curvature across the edge but a small curvature in the perpendicular direction in the DoG function. Principle curvatures can be computed from a 2x2 Hessian matrix which is computed at the location and scale of the keypoint where the derivatives are estimated by taking differences of neighboring sample points:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \qquad \mathbf{(3.5)}$$

The eigenvalues of H are proportional to the principle curvatures of D. Since only the ratio of eigenvalues is considered, it is not needed to compute eigenvalues. If the eigenvalue with the largest magnitude is named as α and the smaller one is β, the sum of eigenvalues can be computed from the trace of H and their product can be computed from the determinant.

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \qquad \mathbf{(3.6)}$$

$$Det(H) = D_{xx} D_{yy} - (D_{xy})^2 = \alpha\beta \qquad \mathbf{(3.7)}$$

If the ratio between α and β is r, then it can be expresed as $\alpha = r\beta$. Then,

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \qquad (\mathbf{3.8})$$

It can be seen that **(3.9)** depends on just the ratio of the eigenvalues, r, not their individual values. The equation **(3.9)** is minimum when the two eigenvalues are equal and it increases with r. Then in order to eliminate these keypoints, it is enough to check whether the ratio of principle curvature is below some threshold, r (=10 in [3]).

$$\frac{D_{xx} + D_{yy}}{D_{xx}D_{yy} - (D_{xy})^2} < \frac{(r+1)^2}{r} \qquad (\mathbf{3.9})$$

3.1.4 Step 4: Orientation Assignment

This step aims to assign a consistent orientation to the keypoints based on local image properties. The gradient magnitude $\mathrm{m(x,y)}$ and orientation $\theta(\mathrm{x,y})$ is pre-computed using pixel differences:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad \mathbf{(3.10)}$$

$$\theta(x,y) = \tan^{-1}\frac{\mathrm{L(x,y+1) - L(x,y-1)}}{\mathrm{L(x+1,y) - L(x-1,y)}} \qquad (\mathbf{3.11})$$

An orentation histogram is formed from the gradient orientations of sample points within a 3σ window around the keypoint [4]. In this histogram, 360 degrees is divided into 36 bins in [3] so each bin consists of 10 degrees. For example, if the gradient direction of a sample point is 21, it will be in the bin for 20-29 degrees. After all pixels are added to a bin, there will be one peak or multiple peaks. All peaks within 80% in the orientation histogram will be assigned to the keypoint. Therefore, there will be keypoints with the same location and scale but different orientations if there are multiple peaks within 80%.

3.1.5 Step 5: Descriptor Vector Calculation

This stage aims to assign a specific vector to each keypoint so each keypoint will be different than others. To achieve this, a 16x16 pixel region around the keypoint is divided into 4x4 sub-regions. For each 4x4 region, gradient magnitutes and orientations are calculated. Their coordinates and gradient orientation are rotated

relative to key-point orientation and are located in a particular bin among 8 bins which are -22.5 to 22.5 , 22.5 to 67.5, 67.5 to 112.5 etc. Their gradient magnitudes are weighted by an appropriate gaussian weighting function to give less significance to gradients that are far from the center of a descriptor. Finally, a keypoint will have 4x4 regions with 8 directions; a total of 128 different numbers.

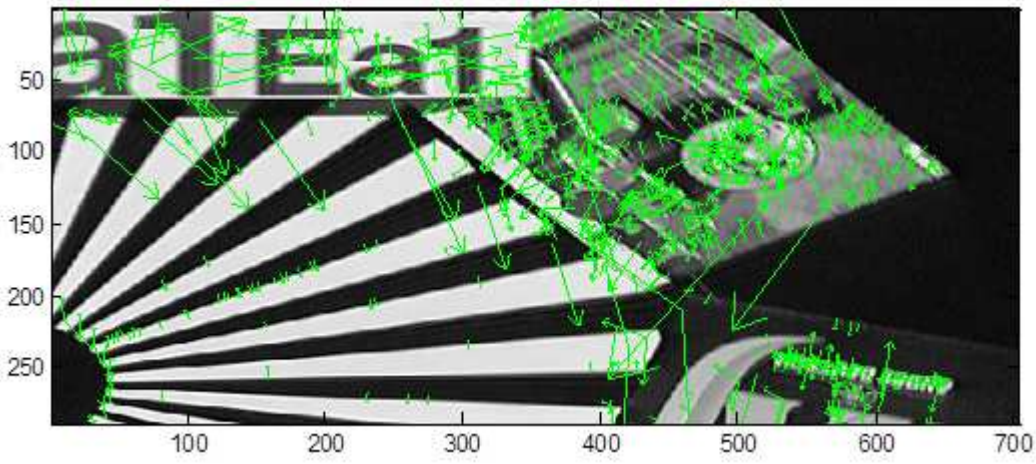

Figure 3.5 : Using [6], 987 SIFT keypoints are found.

Found keypoints in Figure 3.5 are shown on the 288x704 image.

3.2 Motion Vectors using SIFT

This part aims to find motion vectors between two consequtive frames by using SIFT keypoints. It is also a preliminary study of Chapter 4 of this thesis because the number of keypoints per block is important for IMC and is observed for an image and the suitabilitiy of motion vectors that are found through SIFT keypoints are investigated in this part. These analyses will be useful while utilizing keypoints to improve block mathing results.

To investigate the use of SIFT in the subject of motion estimation, SIFT keypoints are found and stored for two sequential frames (let's say *im1* and *im2*) of a video. Then keypoints of *im1* are matched with the keypoints of *im2*.

In order to extract SIFT keypoints, *sift* function which is written in MATLAB and included in the demo software in [6] that is provided by David Lowe is used. In the *sift* function, there is a call to an exacutable file named as *siftWin32* which is the program that finds the invariant keypoints. Input to the sift function is an image file, and it gives *descriptors* and *locs* matrices of the input image as output.

Descriptors is K-by-128 matrix where K is the number of keypoints. Each row gives an invariant descriptor for a keypoint, so descriptor is a vector of 128 values. The parameter *locs* is a K-by-4 matrix which includes the keypoints' location information as *row*, *column*, *scale* and *orientation*. In other words, the first column of *locs* matrix gives the y-position of the keypoints, second column gives the x-positon. Third and fourth columns give the scale and orientation of the keypoint, respectively. Orientation is in the range $[-\pi, \pi]$ radians. When *siftWin32* is called, it writes invariant keypoints to tmp.key.

To find motion vectors via invariant keypoints, first the *sift* function is called to find *descriptors* and *locs* matrices of *im1*, then the respective matrices of *im2* are found by another "*sift*" call. After these matrices are stored, matching operation starts and each keypoint of *im1* is compared with each keypoint of *im2*.

To match im1 keypoints with im2 keypoints, instead of Euclidean distance, [6] suggested the use of a modified distance function to improve the matching speed. In this algorithm, there are two equal length vectors which are 128 long descriptors of im1 and im2. Then the dot product of these vectors are calculated as $\text{im1} \bullet \text{im2}$ which gives the same result as with $|\text{im1}||\text{im1}|\cos\alpha$. Here, angle α between the vectors can be found easily with inverse cosine function. For one keypoint, there is a K-long angle (α) vector; then the values in this vector are sorted in ascending order. The first match in that order is checked. If the distance of first match is less than *distRatio* times the distance to the second match, where distRatio is a constant, the first match is accepted as the best match. In [6], *distratio* is used as 0.6 radians. Figure 3.7 shows matched keypoints of two sequential frames of a video.

Now, the location of a keypoint for both im1 and im2 are known. Then the motion vectors can be calculated and drawn for matched invariant keypoints. In Figure 3.8, motion vectors are drawn by using the MATLAB quiver function and they are shown with red arrows on im1.

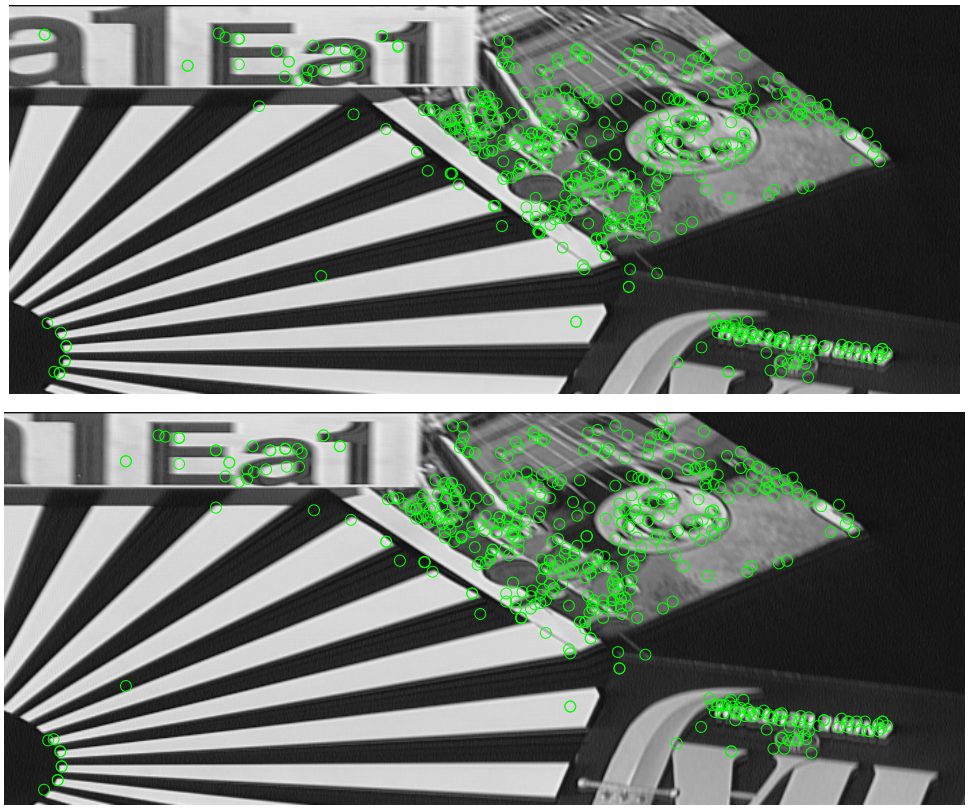

Figure 3.6 : 80th (top) and 83th (bottom) frames of video.

Locations of 508 matched keypoints are shown by circles on corresponding frames.

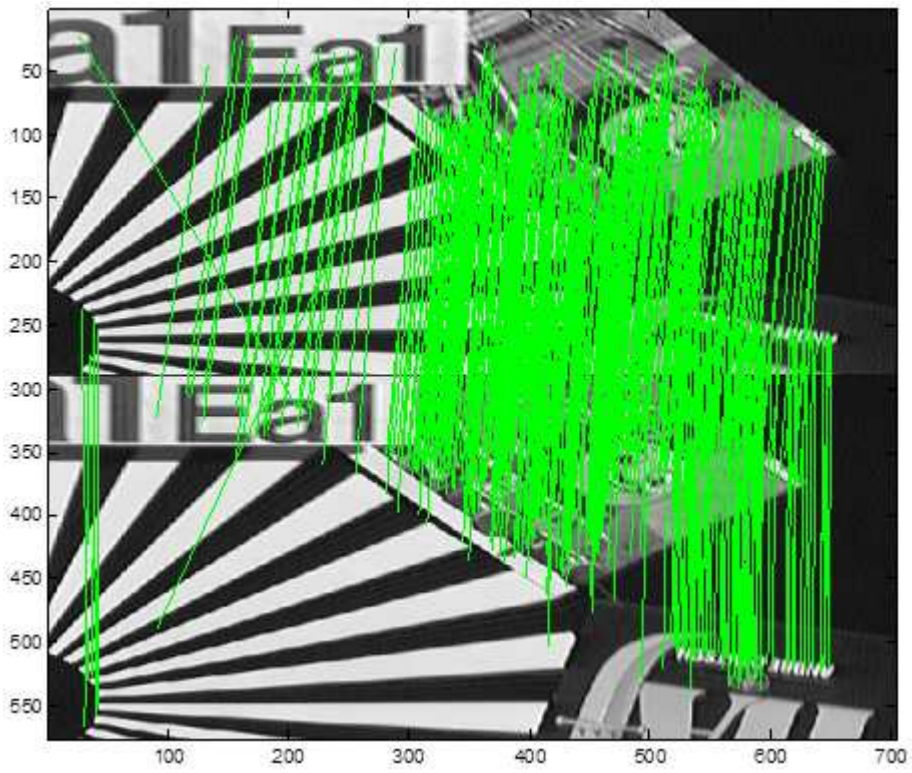

Figure 3.7 : 80th and 83th odd fields of an interlaced video.

Each matched keypoint is connected with a line in Figure 3.7.

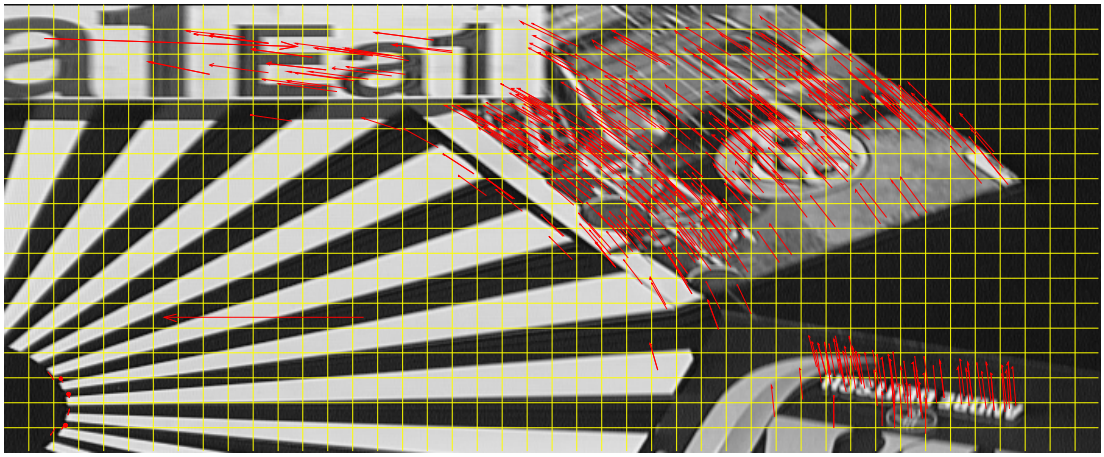

Figure 3.8 : Motion vectors for odd fields 80 to 83 are shown with red arrows.

The frame in Figure 3.8 is divided into 16x16 pixel blocks shown with yellow lines.

4. THE PROPOSED IMPROVED MOTION COMPENSATION TECHNIQUE

In this chapter a new block based motion estimation based on SIFT is proposed to yield a better predicted reconstructed image at the end of the motion compensation process. When a better prediction is achieved for the current frame, the interframe differences will be reduced thus yielding a better compression for block based interframe compression algorithms. The new algorithm is explained clearly in sub-sections of this chapter and the results are exhibited in the next chapter. Simulation results verifiy that the proposed algorithm yields a higher PSNR of reconstructed images, as compared to full search BMA which is the classical motion estimation algorithm as explained in the 2nd chapter. In this algorithm, 16x16 sized blocks are utilized for both BMA and the new algorithm, however any other size may be used.

This new algorithm checks the blocks for which the full search BMA does not give a good result. If there is a rotation, the new algorithm detects and derotates the block in the opposite direction and then uses it in the predicted image.

4.1 Development Environment

The new algorithm is implemented using the MATLAB R2009b software development environment. A PC with Intel Core i5 2.53 GHz CPU and 4 GB of RAM was used.

4.2 Using SIFT in BMA

To improve the BMA algorithm, the SIFT algorithm is used to find rotated objects (blocks) and then applies inverse rotation before reconstruction. Basic understanding of the new algorithm is depicted in Figure 4.1. Here, a block size is NxN (where N is 16 for this thesis) and the reference region is sized as MxM which depends on the rotation angle (α) as seen in **(4.1)**.

$$M = N(\sin\alpha + \cos\alpha) \qquad \textbf{(4.1)}$$

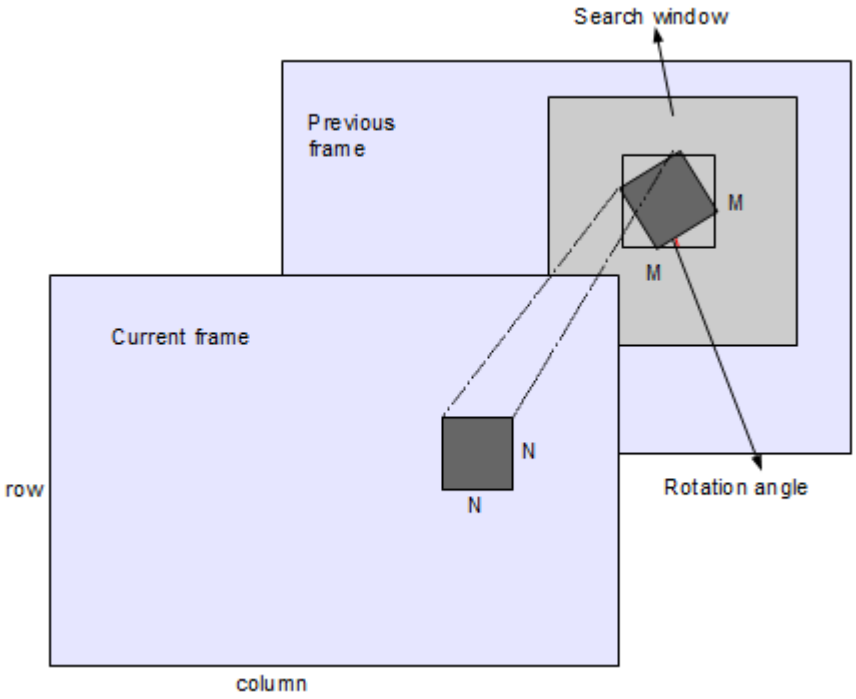

Figure 4.1 : Basic understanding of new algorithm.

4.2.1 Stage 1

In this stage, SIFT keypoints are found for two adjacent frames and matched by using *match* function, which is also provided in [6]. Hovewer, this fuction is modified for Stage 1 algorithm as:

[loc_x, loc_y, delteta, M] = match_alg1 (im_pre, im_cur);

Here, *loc_x*, *loc_y*, *delteta*, *M* are macroblock tables and all have $(\frac{\text{row}}{\text{N}})(\frac{\text{col}}{\text{N}})$ cells where N=16 for this thesis. *im_pre* and *im_cur* are the previous and current frames respectively.

loc_x: For every non-overlapping (16x16) block of current frame, matched SIFT points' x-coordinates in previous frame are averaged through *mean* function and stored in loc_x matrix.

loc_y: For every non-overlapping (16x16) block of current frame, matched SIFT points' y-coordinates in previous frame are averaged through *mean* function and stored in loc_y matrix.

delteta: For every non-overlapping (16x16) block of current frame, matched SIFT points' orientation differences are calculated and stored in *delteta* matrix. This data also gives information about the amount of rotation.

M: is the edge length of reference region in previous frame. After delteta is calculated, *M* can be calculated easily as in **(4.1).** M is calculated for each non-overlapping (16x16) block of current frame and stored in *M* matrix.

After all these data are stored, the BMA algorithm is applied to adjacent frames. For blocks whose minimum SAD is grater than 750 and less than a maximum threshold, SIFT keypoints are checked if there is a rotation or not through *delteta* matrix. If the rotation *delteta(a,b)* is higher than 1 degree, rotation in the reverse direction is applied to the region around the (*loc_x(a,b), loc_y(a,b)*) point. This region's edge length is *M(a,b)* taken from *M* matrix. To apply rotation, *imrotate* [8] function of MATLAB is used. This function takes the image and the rotation angle in degrees as inputs, rotates the image by specified angle in opposite direction around its center point.

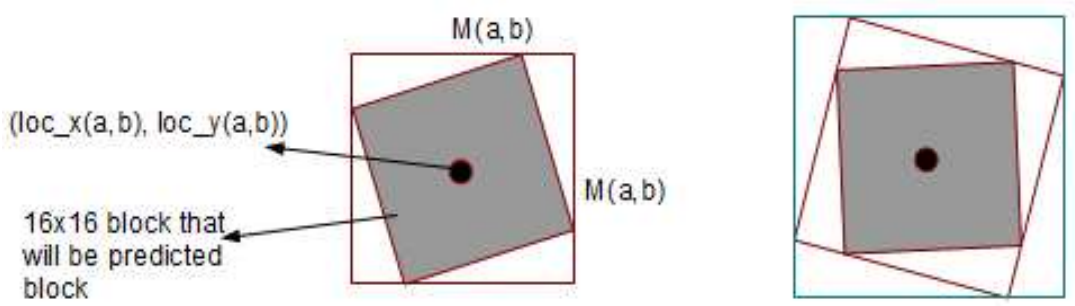

Figure 4.2 : The region (left) is taken from previous image and rotated to the opposite direction as seen on (right). The predicted block is obtained.

Finally, the block that is obtained by BMA is compared with the block that is obtained using Stage 1. Then, the block which gives a better result is accepted as predicted block. In experiments, it is realized that if locations (loc_x and loc_y macroblock tables) are calculated by the "median" instead of "mean" of keypoint coordinates, better results are obtained. However, this stage does not yield a remarkable enhancement because the center of rotation cannot be calculated accurately by taking the median or mean of keypoint coordinates. Stage 1 image results and also numerical results are exhibited in 5th Chapter. Due to the fact that, the center of region is determined by averaging the locations of SIFT keypoints coordinates, results are usually not better than BMA results. Even if Stage 1 algorithm did not give very well results, it was an inspiration for IMC. The flowchart of the Stage 1 algorithm is shown in Figure 4.3.

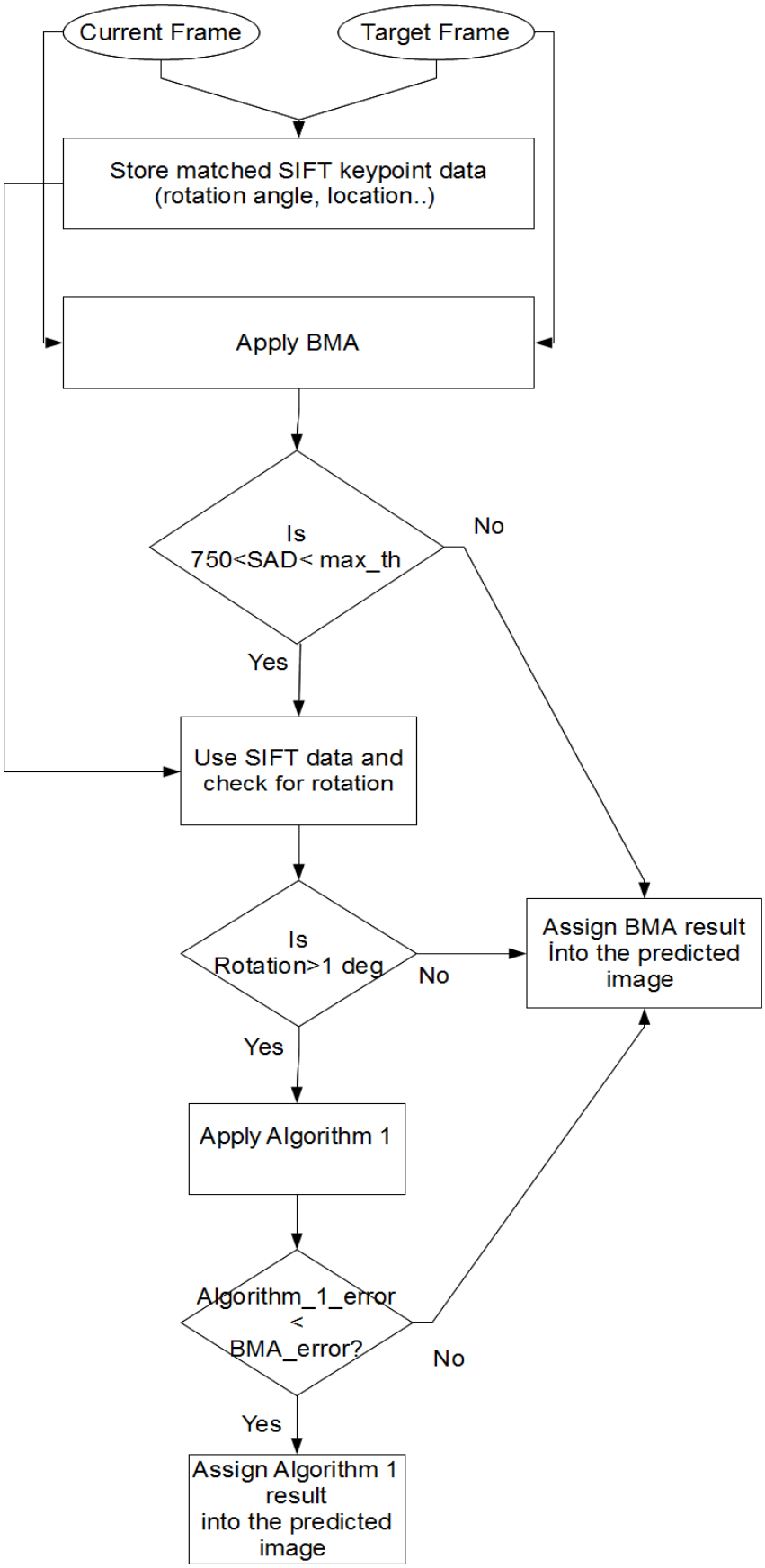

Figure 4.3 : Flow chart of Stage1.

4.2.2 Improved Motion Compensation (IMC)

In stage 2, SIFT points are still used but this time they are modified through *match* function for algorithm 2 as:

[point_num, loc1x, loc1y, loc2x, loc2y, rotation] = match_alg2 (im_pre, im_cur) ;
Here, all outputs are macro-block tables for blocks.

point_num: matrix includes the number of matched noncollinear keypoints for each block.

loc1x: matrix includes x-coordinates of keypoints for each block in previous image.

loc1y: matrix includes y-coordinates of keypoints for each block in previous image.

loc2x: matrix includes x-coordinates of keypoints for each block in current image.

loc2y: matrix includes y-coordinates of keypoints for each block in current image.

rotation: matrix includes orientation differences of keypoints for each block in current image.

The idea is to find a transformation matrix of the 16x16 block from previous image to current image by using matched noncollinear keypoints. When there are higher number of matched keypoints per block, it is possible to detect transformations that are more complex. Generally, there are at most 4 or 5 keypoint in a 16x16 block as mentioned in Chapter 3.2. However, these matched keypoints may be collinear by chance or because some keypoints may have the same location and scale with different orientations as also mentioned in chapter 3.1.4. Therefore, this algorithm focuses on 2D transformations that can be detected by 2 or 3 noncollinear keypoints.

4.2.2.1 Eliminating collinear and same located points

First, collinearity is checked and collinear points are eliminated. To achieve that, *noncollinear* function is created and used as:

[lxn, lyn, num] = noncollinear (lx, ly);

This function takes the x-axis and y-axis locations of matched keypoints in lx and ly arrays respectively, and then gives the outputs as new noncollinear vectors; lxn and lyn. In noncollinear function, MATLAB's *unique* function which finds unique values in an array is used. For example, locations of six matched keypoints in a block are:

lx = [635.5000 635.5000 630.6900 630.6900 636.0000 626.4800 626.4800]

ly = [539.5000 539.5000 537.5300 537.5300 537.1500 535.6400 535.6400]

It is seen that, two pairs in lx and ly are at the same location. This situation causes the collinearity problem. In these arrays, there are four keypoints that can be used to find the transformation matrix. After lx and ly arrays are given to the “noncollinear” function, the outputs will be as:

lxn = [635.5000 630.6900 636.0000 626.4800]

lyn = [539.5000 537.5300 537.1500 535.6400]

Moreover, three points are said to be collinear if they lie on the same line. It means that these three points can not generate a plane or simply a triangle. In this case; the collinearity of points (x_1, y_1), (x_2, y_2), (x_3, y_3) can be checked like that:

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = 0 \qquad \textbf{(4.4)}$$

or it can be expressed as:

$$x_1(y_2 - y_3) - x_2(y_1 - y_3) + x_1(y_1 - y_2) = 0 \qquad \textbf{(4.3)}$$

4.2.2.2 Obtaining the transform matrix

After noncollinear points are obtained, they are used to find transformation of blocks from current frame to the previous one. The most general transformation model is affine model which contains all transformations (translation, expantion, rotation and similarity translations). It means that, if affine transformation in blocks of consequtive frames can be found and corrected, it will cause a better reconstruction than correcting just rotation. However, to find affine transformation matrix at least three control points are needed as it is explained in Case 1. Behind that, non-reflective similarity transformation that supports translation, rotation and isotropic scaling can be calculated with two control points as it is also explained in Case 2.

In this thesis, after matched keypoints are obtained, possible transformations are found with the *cp2tform* function of MATLAB. Keypoint locations are given as inputs to the *cp2tform* function. tform = cp2tform(input_points, base_points, transformtype) takes pairs of control points and uses them to infer a spatial

transformation [7]. *input_points* is an m-by-2 double matrix containing the x- and y-coordinates of control points in the image that will be transformed. *base_points* is an m-by-2 double matrix containing the x- and y-coordinates of control points specified in the base image. *transformtype* specifies the type of spatial transformation to infer [7]. Here, "image that will be transformed" is the reference (previous) block and "base image" is the current block.

transformtype depends on the number of keypoints in the related block. In this algorithm three different cases are investigated;

Case 1: Three or more matched keypoints in a block:

If there are three or more matches for a 16x16 block in target image, it means that "affine" transformation parameters can be obtained. Affine transformation of a point $[x\ y]^T$ to the point $[u\ v]^T$ can be shown as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad \textbf{(4.4)}$$

Here, (x, y) represents the location of keypoint in current block where (u, v) represents its match in target block. Affine rotation, scale and stretch parameters are represented by m_i parameters and translation is represented by t_i parameters in **(4.4)**. It can be seen that, m_i and t_i parameters should be known to solve an affine transformation. Then six equations are needed. **(4.4)** can be rewritten with two equations:

$$u = m_1 x + m_2 y + t_x \quad \textbf{(4.5)}$$

$$v = m_3 x + m_4 y + t_y \quad \textbf{(4.6)}$$

There are two equations but six unknowns in equations **(4.5)** and **(4.6)**. It is clear that a single match is not enough and at least three matches are needed to provide a solution for affine parameters.

As a result, *transformtype* in cp2tform function will be "affine" if there are three or more matches in a block.

Case 2: Two matched keypoint in a block:

Two matches are not enough to solve affine transformation, so the *transformtype* is set to "nonreflective similarity" here. Because this transformation comes up with translation, rotation and scaling but shapes are unchanged. Straight lines remain

straight and parallel lines are still parallel [7]. Non-reflective similarity transformation of a point $[x\ y]^T$ to the point $[u\ v]^T$ can be shown as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ -m_2 & m_1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad \textbf{(4.7)}$$

It is clear that there are four parameters (m_1, m_2, t_x, t_y) that are needed to be calculated for this transformation. It is known that two independent equations can be written with one control point pair, than the transformation can be calculated with two control points. Since non-reflective similarity is supports rotation and translation, it is already applicable to the aim of this thesis.

Case 3: One keypoint in a block:

One match is not enough to solve any transformation type. Therefore, algorithm behaves here such as the stage 1 operations of Chapter 4.2. It accepts x- and y-coordinates of keypoint as center of MxM region in current image where M can be calculated from **(4.1).** Then rotates the region and cuts the 16x16-pixel center block predicted block if its SAD is lower than BMA prediction.

4.2.2.3 Determination of region that will be transformed

The decision for the type of the transformation is explained in 4.2.2.2. In this part, the region in reference image will be determined to apply transformation. This region should be selected so that the region will include 16x16-pixel predicted block after translation. To achieve this, the region center is set as the location which is the minimum of x-and y-coordinates of the key-points in block. For example, there are three keypoints in a block and their coordinates are given in Table 4.1. Here, the center of the region should be chosen as (308.66, 326.4). Then the 30x30-pixel square region is selected around this point to apply translation.

Table 4.1: Coordinates of keypoints in a block.

	x-coordinate	**y-coordinate**
keypoint 1	315.3600	332.1500
keypoint 2	311.2000	326.4000
keypoint 3	308.6600	337.7100

a b c d

Figure 4.4 : a: original block from current frame, b: selected region from reference image , c: the region after transformation of (b), d: candidate predicted block which is selected from (c).

4.2.2.4 Finding predicted block in transformed region

The original block, which is depicted as (a) in Figure 4.4 is searched in transformed image (c) with an algorithm that checks each 16x16 overlapped blocks of (c) and the block that gives minimum SAD is accepted as candidate predicted block. Under this condition, candidate block is compared with the predicted block through BMA and the best one is used as the predicted block. This is how (d) is obtained from (c) in Figure 4.4.

Improved Motion Compensation algorithm that is explained in 4.2.2 is denoted in Figure 4.5.

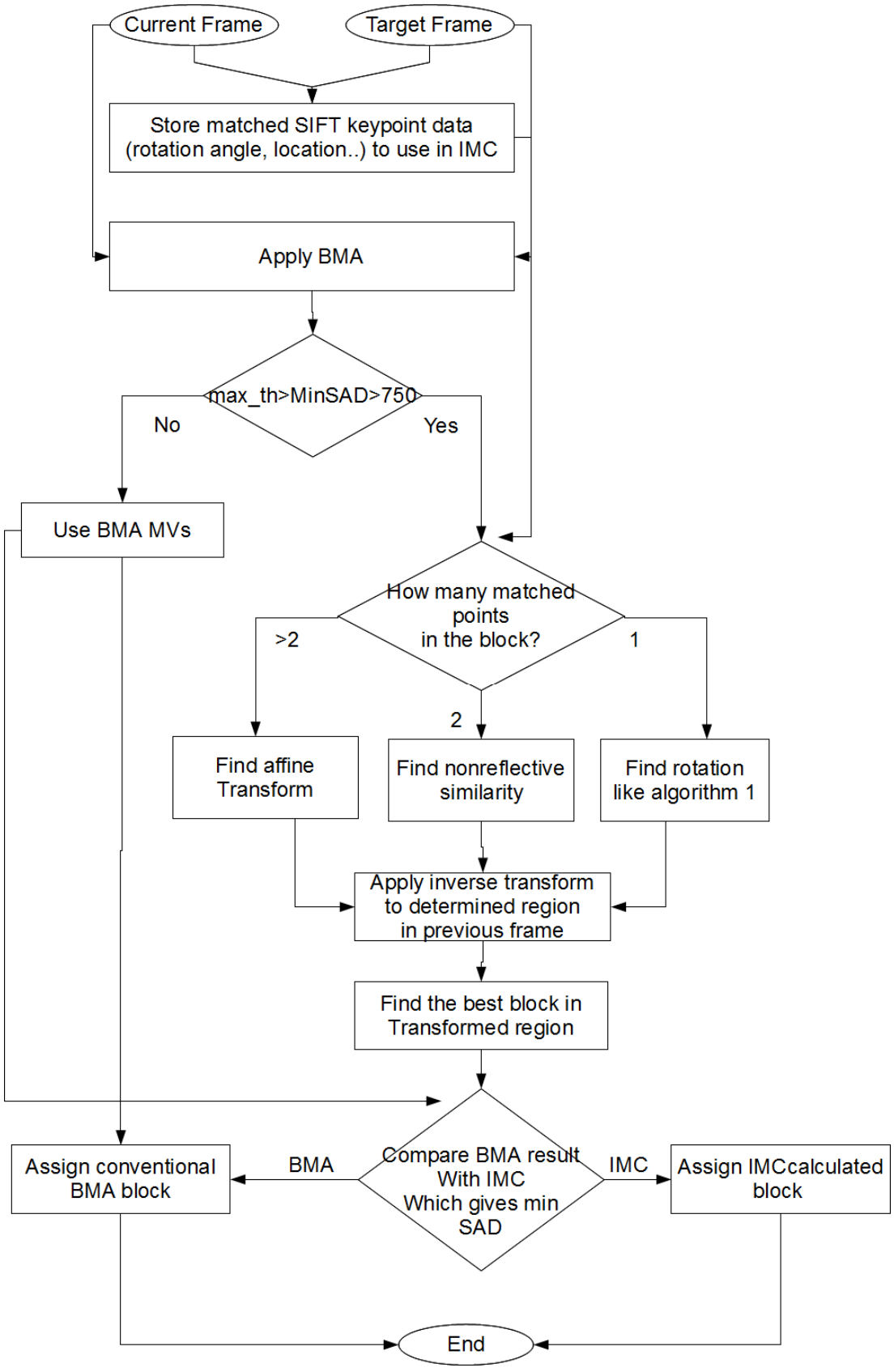

Figure 4.5 : Flow chart of IMC.

5. RESULTS

The algorithm used in Stage 1 (Chapter 4.2.1) and improved motion compensation (IMC in Chapter 4.2.2) algoritms are applied on both test frames and real video frames. In Chapter 5.1, test frame results are exhibited for these new algorithms and in Chapter 5.2, only real video results are exhibited for IMC. All tests are achived on grayscale images in this thesis. If an RGB image is used as an input, it will be automatically converted to grayscale image by all algorithms; BMA, Stage 1 and IMC. For real applications, IMC can be applied to Y components of video frames and the obtained reconstruction information can be applied for all components Y, Cb and Cr.

5.1 Test Frame Results

In this part, a reference and current frame pair are created to test algorithms.They can be seen in Figure 5.1. Size of test frames are both 576x768x3. Current frame is created by rotating the distinctive object in reference frame by 30 degrees towards right. Reconstructed frames that are obtained by the classical BMA algorithm, Stage 1 algorithm and the proposed IMC algorithm are shown in Figure 5.2, Figure 5.3, and Figure 5.4 respectively. Moreover, PSNR values are calculated to measure the performance of algorithms and they are denoted on the Table 5.1. The calculated PSNR values are for the entire frame, not for the rotated section only.

Table 5.1: Test frame results.

	PSNR	**# of blocks checked for rotation**	**# of blocks applied enhancement**
BMA	42.7554	No check	No enhancement
Stage 1	42.8159	61	10
IMC	44.1504	61	50

It is obvious that PSNR improvement of IMC algorithm is ~1.4 dB as compared to BMA for test frames. “# of blocks checked for rotation” column on the table represents number of unsucessful blocks of BMA. It can be seen that 61 blocks are needed to be improved to eliminate distortions of rotation for created test frames. “# of blocks applied enhancement” column on the table represents the blocks on which Stage 1 and IMC algorithms are applied and gives lower SAD than BMA algorithm. While 10 over 61 unsuccessful blocks could be improved through Stage 1 algorithm, this rate is 50 over 61 blocks through IMC algorithm. To see this improvement visually, Figure 5.4 should be compared with Figure 5.

Figure 5.1 : Reference test frame

Figure 5.2 : Current test frame

Figure 5.3 : Reconstructed frame with BMA motion vectors.

Figure 5.4 : Reconstructed frame with Stage 1.

Figure 5.5 : Reconstructed frame with IMC.

5.2 Real Video Frame Results

Test video frames are obtained from the sample video which can be downloaded from; ftp://ftp.tek.com/tv/test/streams/Element/MPEG-Video/625/. Several test videos are available at this site, however the selected one contains more rotational motion than the others. Properties for this video is given in Table 5.2.

Table 5.2: Video properties

Video	**Properties**
Name	bbc3_400.m2v
Size	71.5MB
Total # of frames	375
Duration	15s
FPS	25
Width	704
Height	576
Compression	MPEG-2
Encoding type	interlaced

By decoding the video file, yuv files (.y, .u, .v) are obtained. Both yuv frames and rgb frames are generated from obtained seperate files. However, algorithms are aplied only on odd fields to eliminate interlace artifacts. This is why reference, current and predicted frame sizes are 288x704.

In this part, consequtive frames (81, 82, 83, 84, 85) are reconstructed from a reference frame (80) to see the effect of the algorithm on both B-(bi-directional) and P-(predicted) frames.

At this point, it may be useful to give some information on GOP. The GOP is a group of pictures (frames) within an MPEG coded video stream. There are three types of compressed frames that are organized in a GOP to generate interframe compression. These compressed frames are:

- I-frames: Intra frames, also known as reference frames, contain all data to re-create a complete image. It does not require data from other frames in the GOP.

- P-frames: Predicted frames contain motion-compensated difference information from the closest preceeding I- or P-frame.
- B-frames: Bidirectional interpolated (bi-directional) frames are encoded based on an interpolation from I and P frames that come before and after them.

GOP begins with an I-frame. Afterwards several P-frames follow, in each case with some frames distance. The remaining gaps contain B-frames. An example for this explanation is given in Figure 5.6. As the interframe separation increases between consecutively coded frames, such as when there are several B-frames in between, the prediction suffers.

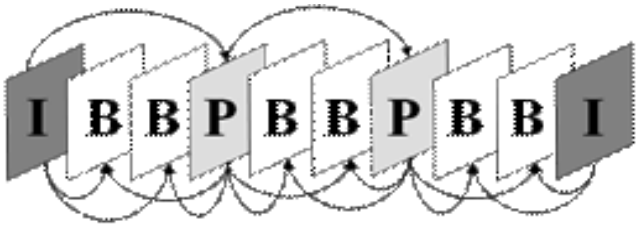

Figure 5.6 : An example of GOP structure

For reconstructed frames (81, 82, 83, 84, 85), PSNR values are given on Table 5.3, computation times are given on Table 5.4 for comparison. Moreover, reconstructed frames of both BMA and IMC are exhibited from Figure 5.7 to 5.23.

Table 5.3: Real video results

Reconstructed Frame	BMA PSNR	IMC PSNR	# of blocks checked for rotation	# of blocks applied enhancement
81	35.1961	35.2023	206	7
82	32.8725	33.1946	195	59
83	31.4469	31.9038	166	81
84	30.7037	30.9523	164	75
85	30.7105	31.0117	155	74

PSNR results in Table 5.3 are calculated for whole images. Therefore, improvement of IMC is repressed in these results. To see the difference of algorithms on blocks, enhancement amount of blocks are investigated and given as SAD values in Appendixes A1.

Table 5.4: Computation Time of algorithms (in seconds)

Reconstructed Frame	BMA time	IMC time
81	15.424923	17.983291
82	14.188610	18.051247
83	14.175006	17.942229
84	14.336778	17.849704
85	14.271704	17.813706

Figure 5.7 : 80th odd field

Figure 5.8 : 81th odd field

Figure 5.9 : Reconstracted 81th odd field through BMA

Figure 5.10 : Reconstructed 81th odd field through IMC

Figure 5.11 : 82th odd field

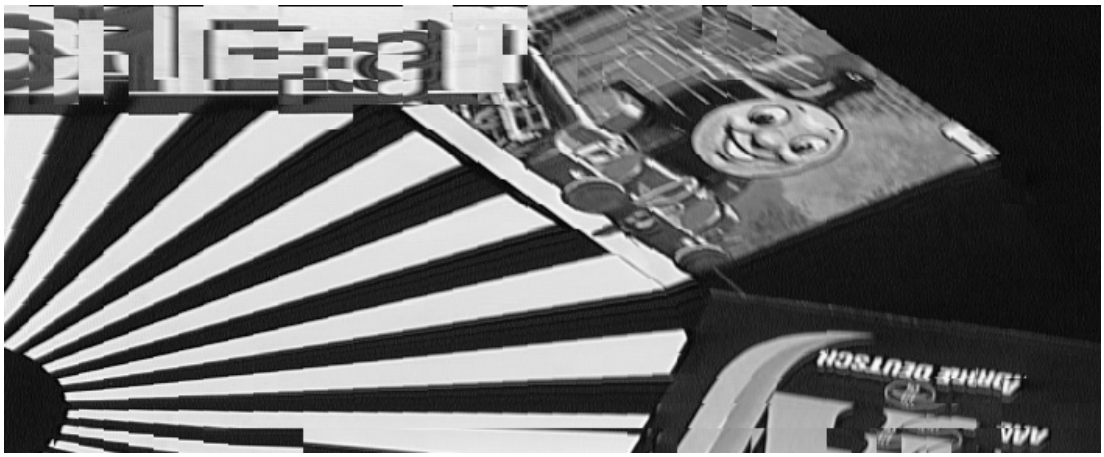

Figure 5.12 : Reconstructed 82th odd field through BMA

Figure 5.13 : Reconstructed 82th odd field through IMC

Figure 5.14 : 83th odd field

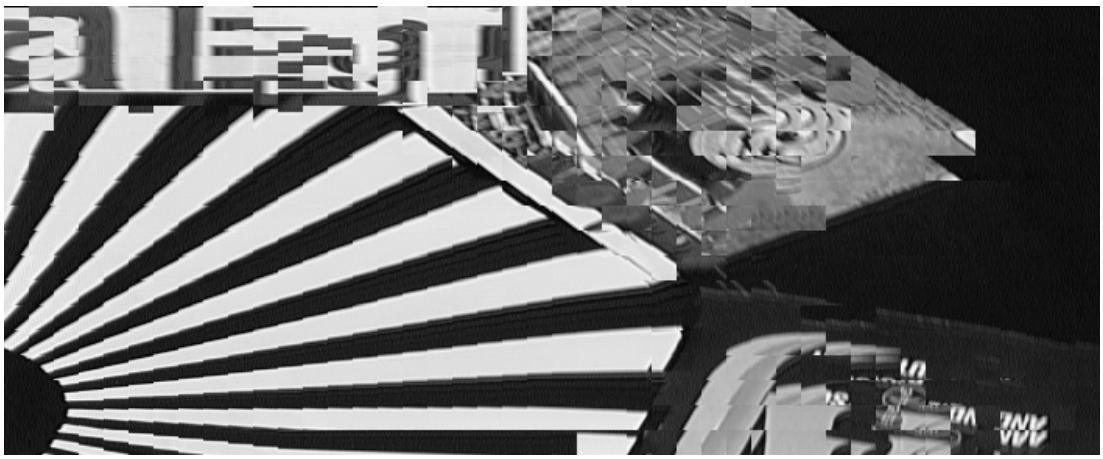

Figure 5.15 : Reconstructed 83th odd field through BMA

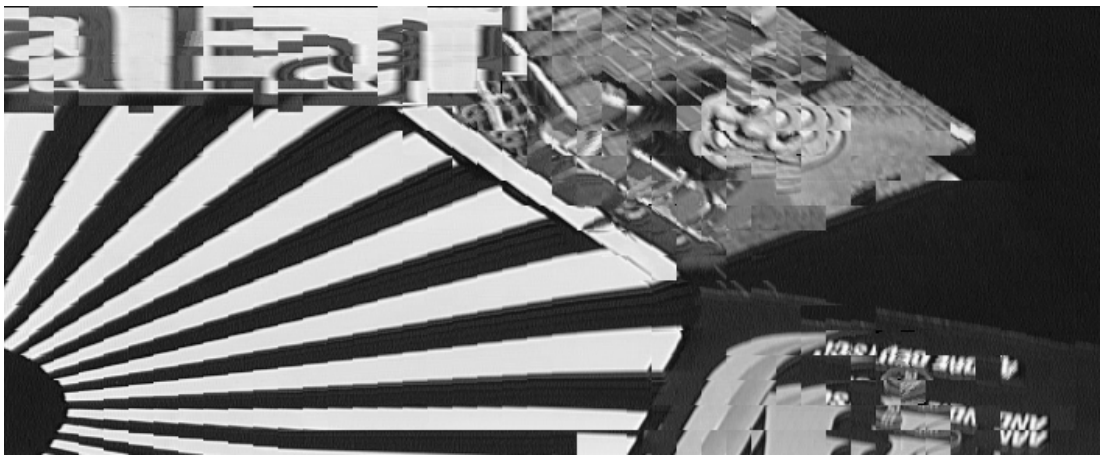

Figure 5.16 : Reconstructed 83th odd field through IMC

As it can be seen from the figures above and Table 5.3, the improvement is better for B-frame and/or P-frame.

Figure 5.17 : 84 th odd field

Figure 5.18 : Reconstructed 84th odd field through BMA

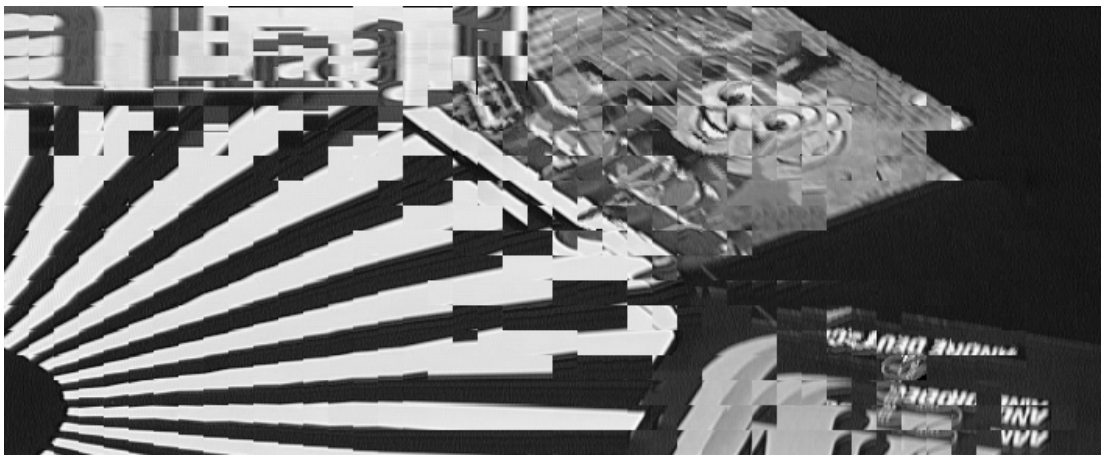

Figure 5.19 : Reconstructed 84th odd field through IMC

Figure 5.20 : 85 th odd field

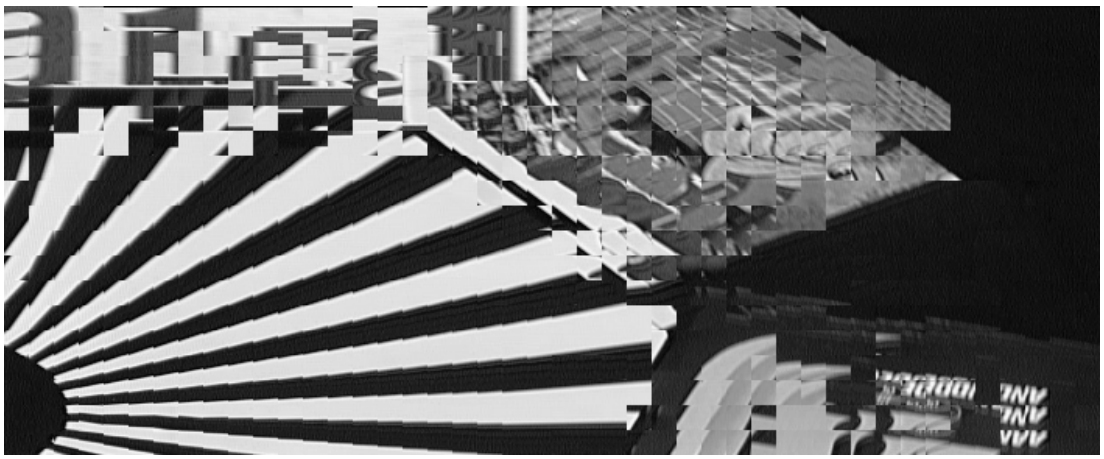

Figure 5.21 : Reconstructed 85th odd field through BMA

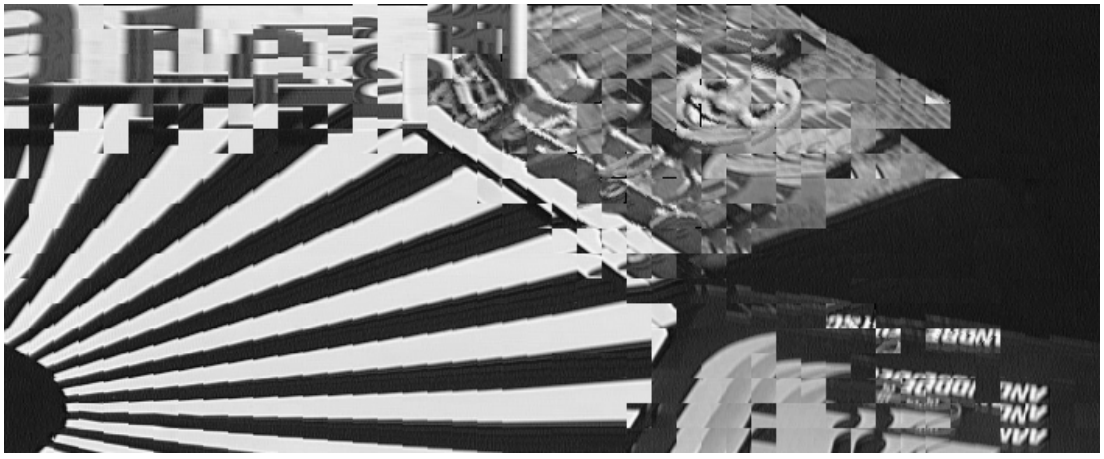

Figure 5.22 : Reconstructed 85th odd field through IMC

6. CONCLUSION

This thesis focused on the utilization of Scale Invariant Feature Transform (SIFT) to achieve a better motion compensation than BMA. Proposed algorithm is called Improved Motion Compensation (IMC) that aims to yield higher PSNR values for reconstructed frames. It tries to detect especially rotated objects/regions between reference and current frames and then corrects it. MATLAB experiments showed that IMC yields ~1.4 dB higher in PSNRs for test frames and ~0.5 dB higher for real video frames as compared to the BMA. Number of SIFT key-points in the rotated region directly effects the performance of IMC. Therefore, it can be said that IMC will be more successful if video frames consists of complex patterns.

REFERENCES

[1] **Wang, Y., Osterman, J., Zhang, Y.,** 2001: Video Processing and Communications, *Prentice Hall*, ISBN:0-13017547-1

[2] **Lowe, D. G.,** 1999: Object Recognition from Local Scale Invarinat Features, *International Conference on Computer Vision Corfu Greece*, pp. 1150-1157

[3] **Lowe, D. G.,** 2004: Distinctive Image Features from Scale-Invariant Keypoints, *International Journal of Computer Vision*, Vol. 60, no. 2, pp. 91-110.

[4] **Lowe, D. G.,** 2004: Method and Apparatus for Identifying Scale Invariant Features in an Image and Use of Same for Locating an Object in an Image, *US Patent*, No: 6,711,293 dated 23.3.2004.

[5] **SIFT**. (n.d.). In *Wikipedia*. Date Retrieved: 03.03.2012, address: http://en.wikipedia.org/wiki/Scale-invariant_feature_transform

[6] **Lowe, D. G.,** 2005: Demo Software: SIFT Keypoint Detector V4, address: http://www.cs.ubc.ca/~lowe/keypoints/

[7] **cp2tform.** MATLAB - Spatial Transformation and Image Registration functions www.mathworks.com/help/toolbox/images/ref/cp2tform.html

[8] **imrotate.** MATLAB - Spatial Transformation and Image Registration functions www.mathworks.com/help/toolbox/images/ref/imrotate.html

[9] **imtransform** MATLAB -Spatial Transformation and Image Registration functions

www.mathworks.com/help/toolbox/images/ref/imtransform.html

[10] **Marques, O.,** 2011: Practical Image and Video Processing Using MATLAB, *Wiley*, ISBN: 978-1-1180-9347-4

[11] **Pan, X., Swei, L.,** 2010: Region Duplication Detection Using Image Feature Matching, *IEEE Transactions on Information Forensics and Security*, Vol. 5, no. 4, pp. 857-867.

[12] **Guo, S., Qui, C., Ye X.,** 2009: A Kind of Global Motion Estimation Algorithm Based on Feature Matching, *International Conference on Mechatronics and Automation*, pp. 107-111.

[13] **Chen, C., Lee, S., Chen, J.,** 2011: An Improved Block Matching Algorithm for Multi-view Video with Distributed Video Codec, *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1-6.

APPENDICES

APPENDIX A.1 : Amount of (SAD) enhancements for blocks
APPENDIX A.2 : Affine Transformations

APPENDIX A.1

Table A.1: SAD enhancements of example blocks.

	80^{th} to 81^{th} fields		80^{th} to 82^{th} fields	
Block	BMA SAD	IMC SAD	BMA SAD	IMC SAD
1	1931	1856	5479	4403
2	1734	1500	6018	3854
3	2281	2183	4028	2540
4	1068	1057	8182	1680
5	1138	1090	10223	2334
6	2758	2529	4200	960
7	3200	3030	10486	2458
8			2485	1443
9			9644	1046
10			5087	1391

Table A.2: SAD enhancements of example blocks (cont.)

	80^{th} to 83^{th} fields		80^{th} to 84^{th} fields	
Block	BMA SAD	IMC SAD	BMA SAD	IMC SAD
1	18501	4100	14292	9859
2	1734	1726	9129	2665
3	9353	2183	7673	2393
4	4675	2040	13951	3268
5	6698	2969	17943	4401
6	7974	1915	16330	3799
7	4548	1237	22961	5495
8	7259	1404	15573	5774
9	10723	2377	4179	1013
10	11083	3725	13074	5530

Table A.3: SAD enhancements of example blocks (cont.)

	80^{th} to 85^{th} fields	
Block	BMA SAD	IMC SAD
1	22037	5834
2	21983	4873
3	18339	5542
4	14561	5136
5	11435	5780
6	16523	4979
7	6021	1559
8	6779	5106
9	7253	2775
10	4893	1435

APPENDIX A.2

To represent affine transformations with matrices, homogeneous coordinates can be used. This means representing a 2D (x, y) as a 3D (x, y, 1), and similarly for higher dimensions. Using this system, translation can be expressed with matrix multiplication. The functional form $u = x + t_x;\ v = y + t_y$ becomes:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \textbf{(A 2.1)}$$

All ordinary linear transformations are included in the set of affine transformations, and can be described as a simplified form of affine transformations. Therefore, any linear transformation can be also represented by a general transformation matrix. The latter is obtained by expanding the corresponding linear transformation matrix by one row and column, filling the extra space with zeros except for the lower-right corner, which must be set to 1. For example, the anti-clockwise rotation matrix from above becomes:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \cos\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(A 2.2)}$$

CURRICULUM VITAE

Name Surname:	Kevser BOZOĞLU
Place and Date of Birth:	Gönen/Balıkesir 22.04.1986
E-Mail:	kevser.bozoglu@gmail.com
B.Sc.:	T.C. Yeditepe University-Electrical and Electronics Engineering, 2009