EFFICIENT DETECTION AND TRACKING OF SALIENT REGIONS FOR VISUAL PROCESSING ON MOBILE PLATFORMS

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

BY

GÜLHAN SERHAT

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

SEPTEMBER 2009

Approval of the thesis:

EFFICIENT DETECTION AND TRACKING OF SALIENT REGIONS FOR VISUAL PROCESSING ON MOBILE PLATFORMS

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

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

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

Assist. Prof. Dr. Afşar Saranlı
Supervisor, **Electrical and Electronics Engineering Dept., METU** _____

Examining Committee Members:

Prof. Dr. Kemal Leblebicioğlu
Electrical and Electronics Engineering Dept., METU _____

Assist. Prof. Dr. Afşar Saranlı
Electrical and Electronics Engineering Dept., METU _____

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

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

Cevahir Çığla, M.S.
Research Engineer, VESTEL _____

Date: 11.09.2009

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

Name, Last Name : Gülhan SERHAT

Signature :

ABSTRACT

EFFICIENT DETECTION AND TRACKING OF SALIENT REGIONS FOR VISUAL PROCESSING ON MOBILE PLATFORMS

Serhat, Gülhan
M.S., Department of Electrical and Electronics Engineering
Supervisor: Assist. Prof. Dr. Afşar Saranlı

September 2009, 111 pages

Visual Attention is an interesting concept that constantly widens its application areas in the field of image processing and computer vision. The main idea of visual attention is to find the locations on the image that are visually attractive. In this thesis, the visually attractive regions are extracted and tracked in video sequences coming from the vision systems of mobile platforms. First, the salient regions are extracted in each frame and a feature vector is constructed for each one. Then Scale Invariant Feature Transform (SIFT) is applied only to the salient regions to extract more stable features. The tracking is achieved by matching the salient regions of consecutive frames by comparing their feature vectors. Then the SIFT points of salient regions are matched to calculate the shift values for the matched pairs. Limiting the SIFT application to only the salient regions results in significantly reduced computational cost. Moreover, the salient region detection procedure is also limited to the predetermined regions throughout the video sequence in order to increase the efficiency. In addition, the visual attention

channels are limited to the most dominant features of the regions. Experimental results that compare the algorithm outputs with ground-truth data reveal that, the proposed algorithm has fine tracking performance together with acceptable computational cost. Promising results are obtained even with blurred video sequences typical of ground vehicles and robots and in an uncontrolled environment.

Keywords: Visual Attention, Saliency, Video Tracking, SIFT, Feature Extraction

ÖZ

HAREKETLİ PLATFORMLARDA GÖRSEL İŞLEMEDE KULLANILMAK ÜZERE DİKKAT ÇEKİCİ BÖLGELERİN VERİMLİ BİR ŞEKİLDE ÇIKARILMASI VE TAKİP EDİLMESİ

Serhat, Gülhan
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Yrd. Doç. Dr. Afşar Saranlı

Eylül 2009, 111 sayfa

Görsel Dikkat, görüntü işleme ve bilgisayarla görme alanlarında günden güne yeni uygulama alanları bulan ilginç bir konudur. Temel fikir, resim üzerindeki görsel olarak dikkat çekici bölgeleri bulmaktır. Bu tezde, hareketli platformların görsel sistemlerinden gelen videolar üzerinde görsel olarak dikkat çekici bölgeler tespit ve takip edilmiştir. İlk olarak, her karede dikkat çekici bölgeler bulunmuş ve her biri için bir özellik vektörü çıkarılmıştır. Daha sonra, daha kararlı özellikler çıkarmak için bu bölgelere SIFT algoritması uygulanmıştır. Ardışık karelerin dikkat çekici bölgeleri, özellik vektörlerini kullanarak eşleştirilmiş ve böylece takip etme işlemi gerçekleştirilmiştir. Daha sonra, SIFT noktaları eşleştirilmiş ve eşlenen çiftlerden yer değişimi miktarları hesaplanmıştır. SIFT algoritmasının sadece görsel olarak dikkat çekici bölgelere sınırlandırılması işlem yükünü ciddi anlamda azaltmıştır. Buna ek olarak verimliliği arttırmak için, video boyunca dikkat çekici bölge bulma işlemi de daha önceden belirlenmiş bölgelere sınırlandırılmıştır. Son olarak, görsel dikkat kanalları bölgenin en baskın görsel dikkat kanalına

sınırlandırılmıştır. Algoritma çıktılarını gerçek verilerle kıyaslayan deney sonuçlarına göre, önerilen algoritma kabul edilebilir bir işlem yüküne ve iyi bir takip etme performansına sahiptir. Yerde giden araçların ve robotların tipik özelliği olan bulanık video görüntülerinde ve kontrolsüz ortam videolarında dahi umut vadeden sonuçlar elde edilmiştir.

Anahtar Kelimeler: Görsel Dikkat, Dikkat Çekicilik, Video Takibi, SIFT, Özellik Çıkarımı

To my mom Azime Serhat

ACKNOWLEDGEMENTS

First, I would like to thank to my thesis supervisor Afşar Saranlı for his guidance, consideration and support. His broad vision, advices and comments assisted this research immensely.

Thanks to SensoRHEX project group, for their contributions and comments throughout the last two years.

Special thanks to my colleague Timuçin Noyan for our technical discussions, his ingenious ideas and friendship.

Special thanks to my friend M. Emre Çavdaroğlu for his comments, corrections and calming support.

Finally, I would like to thank to my family for their care, encouragement and support.

I would like to express gratitude to TÜBİTAK Science Fellowships and Grant Programs Department (BİDEB) for their financial support.

TABLE OF CONTENTS

TABLE OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

VA	Visual Attention
SIFT	Scale Invariant Feature Transform
VAonly	Visual Attention only
SIFTonly	SIFT only
VA+SIFT	Visual Attention + SIFT
VASR	Visual Attention + SIFT + RANSAC
R-based VASR	Region based Visual Attention + SIFT + RANSAC
RC-based VASR	Region based and Channel limited Visual Attention + SIFT + RANSAC
MoReC	Motion based, region based and channel limited Visual Attention + SIFT + RANSAC
Shaky	The test video with shaky motion
Planar	The test video in which the salient region positions do not overlap but form planar trajectories
Blurred	The test video with motion blur
CarPark	The test video recorded in a car park with no intentionally placed salient objects

CHAPTER 1

INTRODUCTION

The aim of image processing and computer vision is to automatically interpret the visual inputs and to obtain information from them. Nevertheless, the amount of data coming from the visual sensors, such as an image, a video stream or a view from multiple cameras, is usually enormous. In most of the cases it is impossible to process the data in total in a bottom up manner. The same problem also holds for human beings. Our eyes supply a great amount of visual data every instant. However, we can successfully interpret the data and easily deduce the relevant high level information to carry out our tasks. What makes this possible within the computational limitations of the human brain may be that, some portions of the visual input data is selected for further processing while the rest is simply ignored. This is called *Visual Attention*.

Visual attention is an important topic in image processing and computer vision which aims to find the visually attractive parts of the input image that can later be used for various purposes. These attractive regions are generally called *salient regions* and saliency is defined as "the state or quality of standing out relative to neighboring items" [26].

There are two motivations behind visual attention. The first one is reducing the dimension of the visual input data, thus reducing the processing power and the memory storage required. This is achieved by eliminating the irrelevant data by means of selecting the regions that presumably involve more information. The

second motivation is to further advance autonomy. Since visual attention automatically selects the visually attractive parts of the visual input, it promises to eliminate the need of human intervention and assistance to visual data processing tasks.

Visual attention has various application areas including high-level scene analysis, image compression and robot vision applications. To give some examples:

- Targets can be automatically detected in natural scenes using visual attention [12], [13].
- Object recognition can be guided by visual attention to achieve high-level scene analysis [21], [31].
- Intelligent image and video compression can be achieved, where the attractive parts are compressed at a higher resolution [17], [11].
- Advertisement designs can be validated by comparing the outputs of visual attention with the desired scheme [25].
- Robot self localization can be accomplished by detecting and tracking the landmarks in the environment [27], [28].

Visual attention has a number of different implementations, which can be categorized into two groups as *saliency-based* methods and *object-based* methods. Saliency-based methods are in fact pixel-based. That is, they compute a saliency value for each pixel, or for a group of pixels, by comparing it with its surroundings. These methods may or may not employ global operations that work on the whole image like global normalization. On the other hand, object-based methods extract the objects in the scene and treat the objects as a whole while computing their saliencies.

In this thesis, the problem of efficient and reliable salient region detection and tracking in a video sequence is studied with a motivation to apply it for visual preprocessing on autonomous mobile platforms. Thus, the proposed method is considered as a preprocessing step, regarding that its outputs may be used in various ways. In this context, many interesting application areas are possible. As

an example, for an autonomous robot with the mission of exploring its environment, say a battlefield or a remote planet; our method would be a preprocessing step that selects and tracks the parts of the scene that are worth analyzing further by subsequent algorithms. In a similar scenario, the relative motion of fixed landmarks in the environment, which are automatically selected by visual attention, may be used to estimate the vehicle ego-motion.

Since mobile devices are selected as the target platform, their characteristic features should be taken into consideration when proposing a solution. Some of the requirements can be listed as below:

- The proposed method should work on video streams,
- In mobile platforms, the camera is in motion which results in blurred videos. So the proposed method must be compatible with this issue.
- In mobile platforms, processing power is usually limited, thus the proposed method should be efficient.

Taking the requirements into account, an efficient salient region detection and tracking method is proposed. The method takes the video input and detects and tracks a pre-determined number of salient regions. Since the method is assumed as a preprocessing step, a saliency-based visual attention algorithm is employed [14]. However, a plain saliency approach has both reliability as well as computational complexity issues for the considered application domain. To increase reliability, the feasibility of using Scale Invariant Feature Transform (SIFT) [18] on salient regions is investigated. And to achieve high computational efficiency, information gathered from the past frames is utilized and smart processing decisions are made such as limiting the regions to be processed.

1.1 Literature Review

Visual Attention has become an attractive topic over past years. The saliency based model of Visual Attention is first proposed in [15] in 1985 by Koch and Ulman and has led to many methods and implementations since that time. Itti *et*

al, in 1998, have proposed a complete mathematical model for saliency using intensity, color and orientation differences as saliency sources [14]. This method has constituted a base for computational saliency-oriented visual attention methods. In [24], Ouerhani and Hügli have added the Harris *cornerness* measure [9] to the method as a saliency source. There are also some studies that concentrate on object-based visual attention such as [5] and [33]. In 2003, Ma and Zhang [19] proposed a new framework by dividing the image into blocks and focusing on the difference between the center of one block and the neighboring blocks and employed a fuzzy growing method. In [7], using integral images in Itti's method is proposed instead of computing features at several scales. This technique introduced a significant gain in computational efficiency, resulting in a real time visual attention system. In [1], a contrast determination filter is used in various scales to obtain a saliency map at the same resolution of the original image and the obtained saliency map is used to segment the whole objects.

There also exist several published studies that utilize visual attention as a feature extraction method. To give some examples, Ouerhani and Hügli present an image segmentation method based on visual attention [22], and object tracking method for dynamic scenes [23]. [31] is an example of object recognition based on visual attention. A context-based scene recognition method for mobile robotics applications is introduced in [29].

The topic of salient region tracking can be found in [24] and [16]. In the work of Ouerhani and Hügli in [24], corner conspicuity channel is added to the intensity and color channels. Each "spot of attention" is characterized by its spatial location and a feature vector. The feature vector displays the contributions of intensity, red-green, blue-yellow and corner channels to the detection of that spot. A trajectory is built for each tracked spot of attention. A newly found spot of attention is added to a trajectory if its spatial location and feature vector are close to the head of the trajectory. Otherwise, a new trajectory is created for it. In the end the trajectories that are long enough are selected as landmarks.

In [16], Li also has an implementation of salient region detection and tracking in video. Salient regions are extracted using orientation and color maps. In the following video frames, detection is not performed but these regions are tracked using a color-based tracking scheme. When tracking a salient region, only the rectangular region around its previous position is searched. Periodically, salient region detection procedure is repeated.

The approach studied in this thesis suggests first determining Visual Attention regions and then applying SIFT only on these regions to extract more specific features. This topic is not entirely new in the literature, but has applications only on still images.

In [32], Walther, Rutishauser and Koch suggested applying the feature extraction algorithms on the regions extracted by VA algorithm and conducted some experiments using the SIFT features. The experiments presented that "saliency-based region selection improves object recognition in highly cluttered scenes considerably". Moreover, salient region selection has added new capabilities to the object recognition algorithm such as "learning multiple objects from single image".

In [8], Gao has studied eliminating SIFT points that has low saliency values to achieve the task of image retrieval. The results showed that keypoint elimination enhances both the speed and the accuracy of image retrieval. Speed is improved due to the reduced number of keypoints while accuracy is improved since keypoints with high saliency values are usually more distinctive.

In [3], visual attention is used to speed up object recognition using SIFT features. First salient regions are detected by Itti & Koch's visual attention algorithm. Contrast, orientation, color, and intensity low-level features are used as saliency measures. Then Lowe's SIFT algorithm is employed to extract "a signature of the attended object". This signature is used to compare the object with the object database, but first the object database is put in the decreasing order of similarity with the objects with already computed low-level features. Experiments show that

ordering the search database with respect to saliency features increases the speed of object recognition in complex natural scenes.

1.2 Motivation

This thesis proposes an efficient salient region detection and tracking method for the visual processing sub-systems of mobile platforms. Such a technique may constitute a basis for various tasks. Exploring unknown environments using unmanned vehicles, autonomous target detection, autonomous search and rescue and self-localization are only some of the examples. We are highly motivated by the extend of possible application areas.

1.3 Scope of the Thesis

With the proposed method, multiple salient regions can be detected and tracked in video sequences. First, the salient regions on the frame are found by applying the Visual Attention algorithm of Itti et al [14]. Then, a saliency feature vector is extracted for each salient region. Each region is tracked by a trajectory and the salient regions found in a new frame are either added to the existing trajectories or initiate a new trajectory depending on their saliency feature vectors. Then, SIFT algorithm [18], which extracts more specific and robust image features, is applied only on the salient regions. In consecutive frames, the extracted SIFT points are matched to compute the inter-frame displacement values, thus to achieve more precise tracking. Eliminating the false SIFT point matches by RANSAC [6] enhanced the performance. The increasing computational cost of combining visual attention with SIFT features is tackled by limiting the salient region search areas and by limiting the visual attention channels. SIFT feature extraction is also limited to the determined salient regions. As a last step, the search regions are further narrowed by estimating the salient region locations using the extracted camera motion information.

At each step the methods are tested and the experimental results are compared both with ground truth data and with each other. In addition, the proposed algorithms are also tested with realistic video sequence data to evaluate their performances.

1.4 Contributions

The following points, to the best of our knowledge, represent the novel contributions of this thesis:

- Development of a complete and efficient salient region detection and tracking method.
- Testing of a salient region detection and tracking method with videos representing legged motion.
- Applying SIFT on only the visually salient regions on a video sequence to enhance the tracking performance.
- Limiting the visual attention channels on successive frames by using only the most dominant visual channel of previous frame.

1.5 Outline of the Thesis

The organization of this thesis is as follows: In chapter 2, three major algorithmic components of the proposed approach, namely Visual Attention, SIFT and RANSAC, are explained in detail to form the theoretical background of the thesis. In chapter 3, the test data and performance criteria are introduced first. Then the proposed approach to achieve a reliable and efficient salient region detector and tracker is presented and explained in an incremental manner. In addition, the results of the comprehensive experiments that are conducted to verify the claimed ideas are presented. In chapter 4, the proposed methods are extensively tested with realistic data. Last, chapter 5 presents the conclusions and possible directions for future work.

CHAPTER 2

THEORETICAL BACKGROUND

In this thesis, the Visual Attention [14], Scale Invariant Feature Detection (SIFT) [18] and Random Sample Consensus (RANSAC) [6] algorithms are commonly utilized. This chapter gives the algorithmic details of these algorithms as the theoretical background. Section 2.1 examines the Visual Attention, section 2.2 examines the SIFT algorithm and section 2.3 examines the RANSAC algorithm.

2.1 Visual Attention

Despite its very limited processing speed, human brain has the ability to process highly complicated scenes in real time. One of the reasons that make this possible is a preprocessing step called "visual attention". In human brain, there exists a mechanism that selects some attention-drawing parts of the scene, which are called "salient locations". Then these salient parts are segmented and only these regions are passed to further processing units. This decrease in the dimension of the data results in a very significant increase in the processing speed.

The features that are required for a region to be salient have been investigated a lot through many experiments done on primates. That features include, but not limited to, intensity contrast, color contrast, orientation contrast, texture contrast and shape contrast. In humans, the attention system has both a "bottom-up,

saliency-driven, task-independent" mechanism, as well as a "top-down, volition-controlled, task-dependent" one [14].

The bottom-up, saliency-based visual attention scheme is introduced in [15] by Koch and Ulman. In [14], Itti et al, have formulated and implemented the theory, which considerably increased the popularity of the visual attention concept.

The algorithm presented in [14] uses RGB color images as input and has 4 main steps:

1. Extracting some visual features from the input image and creating feature maps for them.
2. Constructing conspicuity maps for each feature, which show the parts of the image that strongly differ from their surroundings.
3. Constructing a saliency map by combining all conspicuity maps.
4. Selecting the most salient locations by using a winner-take-all neural network on the saliency map.

The flow of the algorithm can be visualized in Figure 2.1.

2.1.1 Feature Maps

In order to detect the locations which are noticeably different from their surroundings, a "center-surround" mechanism is implemented. The method is based on computing the differences between fine and coarse scales. A pixel at scale $c \in \{2,3,4\}$ is taken as the center and the corresponding pixel at scale $s = c + \delta$, where $\delta \in \{3,4\}$, is taken as the surround. Then the coarser scale is interpolated to the finer scale and point-by-point subtraction gives the across-scale difference, denoted by Θ. Using each c and δ values, 6 maps are constructed per feature. This allows multi-scale feature extraction.

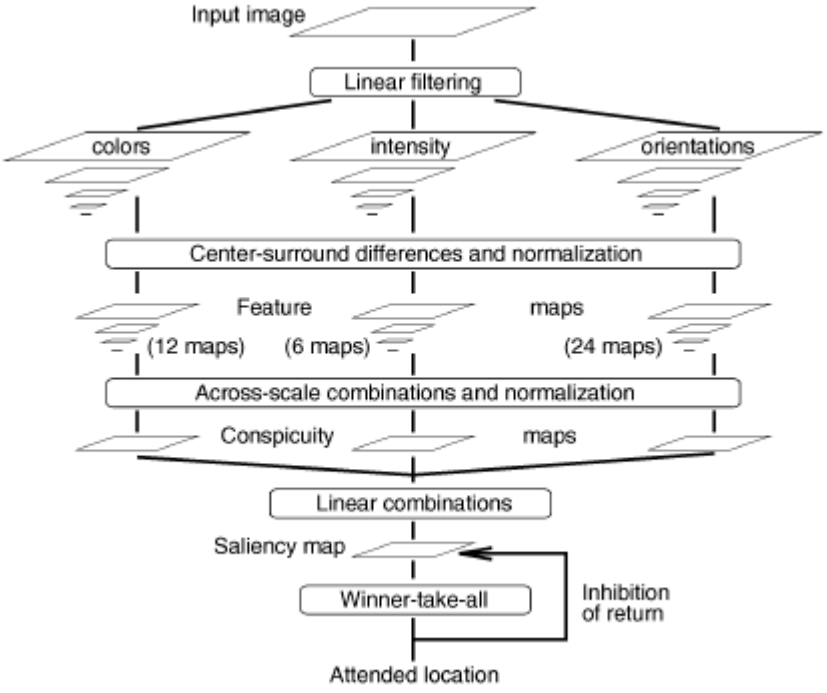


Figure 2.1 – Flowchart of Visual Attention algorithm

First, color, intensity and orientation features are extracted from the image. The center-surround difference computation and normalization of the features give the feature maps. Then, feature maps are combined into conspicuity maps, which are also combined into the saliency map. Finally, a winner-take-all neural network selects the most salient locations [14].

The visual cues used in [14] are intensity, color and orientation. Based on these cues, 7 features are extracted for each image.

- Intensity cue:

  First, intensity image is obtained by

$$I = (r + g + b)/3 \tag{2.1}$$

  where *r, g, b* are *red, green, blue* channels of the image respectively.

Then, a set of 6 center-surround difference maps are computed by

$$I(c,s) = \left| I(c)\ \Theta\, I(s) \right| \tag{2.2}$$

where $c \in \{2,3,4\}$ and $\delta \in \{3,4\}$.

- Color cue:
  First, four color channels are created, which are defined as follows:

  For red:

$$R = r - (g+b)/2 \tag{2.3}$$

  For green:

$$G = g - (r+b)/2 \tag{2.4}$$

  For blue:

$$B = b - (r+g)/2 \tag{2.5}$$

  For yellow:

$$Y = (r+g)/2 - \left| r-g \right|/2 - b \tag{2.6}$$

  Then, red-green and blue-yellow "double opponency system", which also exists in human visual cortex, is employed to extract two chromatic features.

  For red-green:

$$RG(c,s) = \left| \left(R(c) - G(c)\right) \Theta \left(G(s) - R(s)\right) \right| \tag{2.7}$$

And for blue-yellow:

$$BY(c,s) = \left|\left(B(c) - Y(c)\right)\Theta\left(Y(s) - B(s)\right)\right| \qquad (2.8)$$

A set of 12 center-surround difference maps are computed.

- Orientation cue:
  Oriented Gabor pyramids are used in $\Theta = \{0°, 45°, 90°, 135°\}$ degrees to extract local orientation features. 24 center-surround difference maps are computed by,

$$O(c,s,\theta) = \left|O(c,\theta)\,\Theta\,O(s,\theta)\right| \qquad (2.9)$$

Thus, a total of 42 feature maps are computed: 6 for intensity, 12 for color and 24 for orientation.

2.1.2 Conspicuity Maps

The 42 feature maps are combined into 3 conspicuity maps $\overline{I}$, $\overline{C}$ and $\overline{O}$ corresponding to intensity, color and orientation respectively. To be able to combine different feature maps, a map normalization operator is introduced. The normalization operator, $\mathcal{N}(.)$, applies three stages:

- The values in the map are normalized to a fixed range [0..*M*].
- The location of the map's global maximum *M is found* and the average *m* of all other local maxima is computed.
- The map is multiplied by $(M-m)^2$.

The normalization operator is illustrated in Figure 2.2. It can be seen that this normalization operation amplifies "small number of strong peaks", while smoothing "numerous comparable peaks".

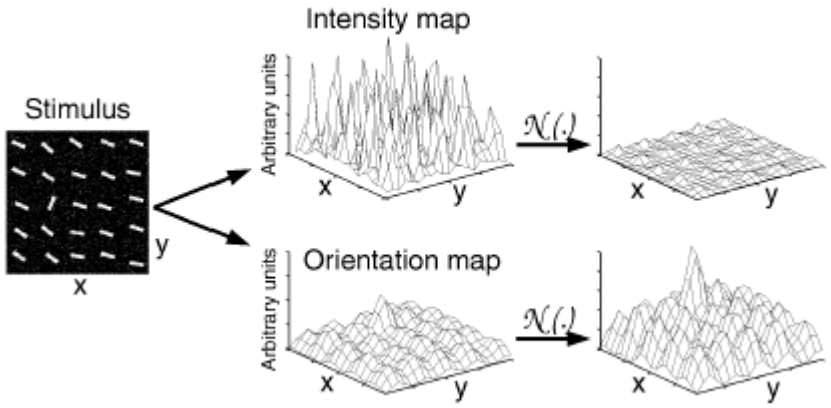


Figure 2.2 – Feature map normalization operator

The normalization operation suppresses frequent peaks while amplifying rare peaks. In the sample image, the numerous intensity differences are suppressed while the unique orientation difference is amplified [14].

The intensity, color and orientation based feature maps are normalized and combined into related conspicuity maps according to the following formulas:

$$\overline{I} = \bigoplus_{c=2}^{4} \bigoplus_{s=c+3}^{c+4} \mathrm{N}\big(I(c,s)\big) \tag{2.10}$$

$$\overline{C} = \bigoplus_{c=2}^{4} \bigoplus_{s=c+3}^{c+4} \big[N\big(RG(c,s) + N\big(BY(c,s)\big)\big)\big] \tag{2.11}$$

$$\overline{O} = \sum_{\theta \in \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}} N\left(\bigoplus_{c=2}^{4} \bigoplus_{s=c+3}^{c+4} N\big(O(c,s,\theta)\big) \right) \tag{2.12}$$

2.1.3 Saliency Map

At this step, the three conspicuity maps are normalized and combined into a final map called Saliency Map by,

$$S = \frac{1}{3}\left(N(\bar{I}) + N(\bar{C}) + N(\bar{O})\right) \qquad (2.13)$$

The saliency map directly illustrates the salient locations in the image, where the maximum value on the map corresponds to the most salient region on the image.

2.1.4 Selecting the most salient location

A winner-take-all (WTA) neural network scheme is employed to control the attention shifts. First, the maximum value of the map is searched. This point, corresponding to the most salient region, is regarded as the winner and the "focus of attention" is shifted to this point. Then, a local inhibition mechanism is activated around this point to avoid further attention shifts to this already focused region. Accordingly, the next most salient location becomes the winner and attention is shifted to that region. The number of regions to attend can be adjusted.

In this thesis, iLab Neuromorphic Vision Toolkit - Windows porting is utilized as the visual attention code [10].

2.2 Scale Invariant Feature Transform

Scale Invariant Feature Transform (SIFT) is an image feature extraction method developed by David Lowe [18]. SIFT features are known for being invariant and distinctive, so they can reliably be used for image matching purposes. They are independent from scale and rotation, and robust to affine distortions, viewpoint changes, noise and illumination changes. Additionally, SIFT features are highly distinctive, thus can be correctly matched against a large set of features. By the

help of these properties, SIFT features can be used in many areas including object recognition, stereo matching, 3D structure estimation and motion tracking.

SIFT algorithm takes gray-scale input images. The method consists of four main steps:

1. Stable keypoint candidates are searched over all scales using difference-of-Gaussian (DOG) function.
2. The intensity values around each keypoint are modeled to determine the keypoint's location and scale in sub-pixel accuracy. In addition, unstable keypoints are eliminated.
3. Local gradient directions are computed and based on them, orientations are assigned to each keypoint.
4. The local gradients around each keypoint are used to construct a keypoint descriptor.

2.2.1 Scale-space extrema detection

2.2.1.1 Obtaining difference-of-Gaussian images

The algorithm starts with detecting candidate keypoints which are stable against scale change. To achieve scale independence, stable features are searched over all possible scales.

As scale space kernel, variable scale Gaussian function is used.

Let $L(x, y, \sigma)$ be the scale space of an image, $G(x, y, \sigma)$ be the variable scale Gaussian defined as $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$ and $I(x, y)$ be the input image;

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.14}$$

To be able to detect stable keypoint locations, difference-of-Gaussian function $D(x, y, \sigma)$ is used, which is the difference between two adjacent scales.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \quad (2.15)$$

$$= L(x, y, k\sigma) - L(x, y, \sigma) \quad (2.16)$$

where *k* is a constant multiplicative factor between the scales.

Difference-of-Gaussian function is a relatively efficient scale-space function. Furthermore, it provides a close approximation to scale normalized Laplacian of Gaussian function, which is known to produce the most stable scale invariant image features [20].

The method of obtaining the difference-of-Gaussian function can be visualized in Figure 2.3.

- First, the original input image $I(x, y)$ is serially convolved with Gaussian functions $G(x, y, \sigma)$. This results in a series of images, called an octave, shown in left side of the figure. In an octave, the scale of each image differs from the scale of previous image by a constant factor *k*, and the last image in an octave has twice the scale of the first image. Each octave is chosen to be divided into *s* intervals, where s is an integer and related to *k* by $k = 2^{1/s}$. To be able to cover a complete octave for local-extrema detection, *s+3* images are created for each octave.
- To produce the difference-of-Gaussian images, shown in right side of the figure, images with adjacent scales in an octave are subtracted.
- After processing the complete octave, the image whose scale is twice the initial scale is subsampled by 2 to obtain the first image of the second octave.

Some typical values for the algorithm are $k = \sqrt{2}$, s=2, and s+3=5 images are created for each octave. The first image in the octave will be the input image with scale σ, the second will have scale $\sqrt{2}\sigma$, and the third will have scale 2σ, so the initial scale will be doubled. But to cover the complete octave for local extrema

detection, 2 more images will be created to complete the number to s+3 which equals 5. When passing to the next octave, the third image, whose scale is 2σ, will be downsampled.

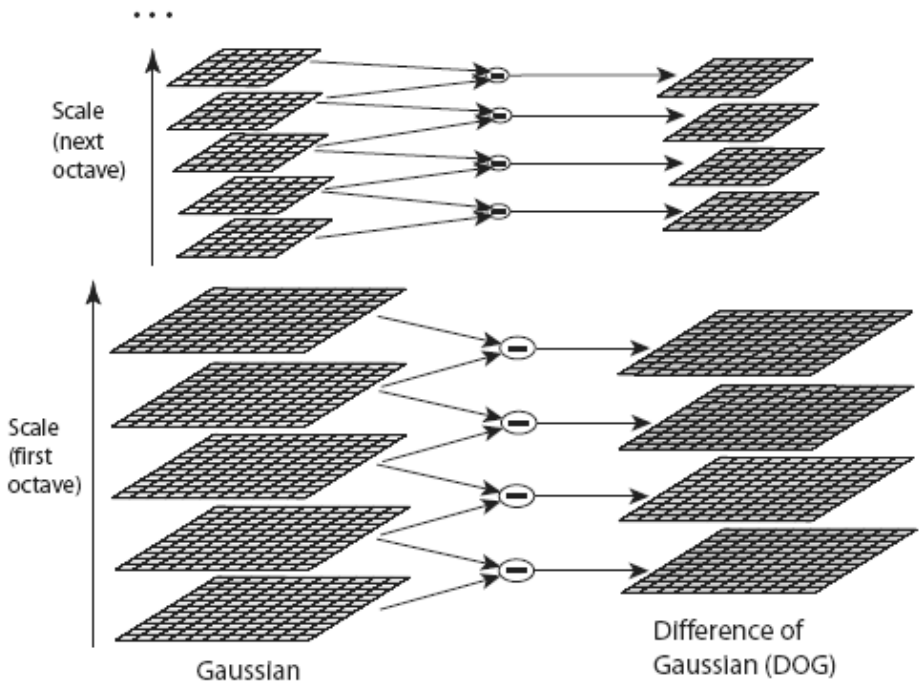


Figure 2.3 – Difference of Gaussian operation

In each octave, the first image is convolved by a series of Gaussians which produces the scale space images. Each image in an octave differs by a scale factor k from its previous image. Subtracting the adjacent images in an octave produces the difference-of-Gaussian images on the right. After processing an octave completely, the Gaussian blurred image is downsampled by 2 to create the initial image of the next octave [18].

2.2.1.2 Detecting local extrema

To obtain stable keypoints, the local maxima and minima of Gaussian function $D(x, y, \sigma)$ are searched. To detect them, each point is compared with its 8 neighbors on the same image and 9 neighbors on the one scale above and one scale below images (see Figure 2.4). The point is selected as a candidate keypoint if it is larger than or smaller than all its neighbors.

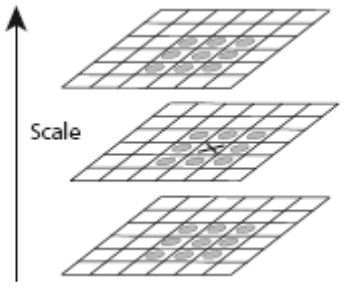


Figure 2.4 – Neighborhood of the candidate keypoint
To find local minima and maxima, each point (marked with X) is compared with its 26 neighbors, 8 on the same image, 9 on the scale above and 9 on the scale below [18].

The sampling frequency for extrema detection introduces a tradeoff between efficiency and completeness. The best choice can be experimentally determined according to the task.

2.2.2 Keypoint Localization

After determining local extrema, which are the candidate keypoint locations, the keypoints should be more accurately localized for enhanced stability. To achieve this, a detailed function is fit to the nearby pixels of the candidate keypoints. This

step brings some improvements to the algorithm such as; sub-pixel accuracy, elimination of low contrast points and elimination of edge points.

This technique, fitting a 3D quadratic function to the neighboring region of the sample point, is developed by Brown [4]. In his approach, the Taylor series expansion of the scale-space function $D(x, y, \sigma)$ is used.

$$D(\bar{x}) = D + \frac{\partial D^T}{\partial \bar{x}} \bar{x} + \frac{1}{2} \bar{x}^T \frac{\partial^2 D}{\partial \bar{x}^2} \bar{x} \qquad (2.17)$$

The function includes up to the quadratic terms and is shifted to have origin at the sample point. D and its derivatives are calculated at the sample point and $\bar{x} = (x, y, \sigma)^T$ is the offset from the sample point.

The extrema location $\hat{x}$ can be evaluated by taking the derivative of this equation and equating it to zero, which results in;

$$\hat{x} = -\frac{\partial^2 D}{\partial x^2}^{-1} \frac{\partial D}{\partial x} \qquad (2.18)$$

If $\hat{x}$, which represents the offset of the accurate keypoint location from the sample point location, is greater than 0.5 in any direction; that means the accurate location is closer to another pixel. In such a case, the sample point is changed and the operations are repeated for the new one. The final $\hat{x}$ value is added to the location of the sample point, which provides sub-pixel localization.

2.2.2.1 Eliminating low contrast points

The points which have low contrast in their neighborhood typically provide unstable keypoints so they should be removed. To obtain them, equation (2.18) can be substituted into equation (2.17), which results;

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \overline{x}}\hat{x} \tag{2.19}$$

Assuming image pixel values are in the range [0,1], the points that have $|D(\hat{x})|$ less than 0.03 are discarded.

2.2.2.2 Eliminating edge points

The locations of keypoints which are localized along an edge are generally quite unstable and sensitive to noise, so they should be eliminated for enhanced stability. To find out the points positioned along an edge, the principal curvatures of the difference-of-Gaussian function are examined. For a point lying along an edge, the DOG function will have a small principle curvature along the edge direction and a large principal curvature in the perpendicular direction. On the other hand, a stable point is expected to have comparable curvatures in both directions. The principal curvatures can be evaluated using the Hessian matrix;

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

The eigenvalues of the Hessian matrix are proportional to the principal curvatures of the DOG function. Since only the ratio of the eigenvalues is needed, explicit computation of them can be avoided. This approach is the main idea behind the Harris Corner Detector [9].

Let α and β be the eigenvalues of **H**; then trace of **H** gives the sum of α and β, and determinant of **H** gives their multiplication.

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \tag{2.20}$$

$$Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta \tag{2.21}$$

Let *r* be the ratio of the eigenvalues, such that $\alpha = r\beta$,

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \qquad (2.22)$$

depends only on r, but not on α and β.

The value $\frac{(r+1)^2}{r}$ takes its minimum value when *r*=1, i.e., when the two eigenvalues are equal, and increases as their ratio *r* increases. Thus, checking this value against a threshold is enough to observe the ratio of the principal curves. The Lowe paper [18] uses a value of *r*=10 as a threshold.

2.2.3 Orientation Assignment

Now that the keypoints are selected, unstable ones are eliminated and their accurate locations are found, a descriptor should be assigned to each keypoint to characterize it. This descriptor should be scale and rotation invariant. The scale invariance is achieved by selecting the Gaussian smoothed image *L(x,y)*, with the scale closest to the scale of the keypoint, and performing all operations on this image. To achieve rotation invariance, an orientation is assigned to each keypoint and the keypoint descriptor is computed relative to this orientation.

For a sample image *L(x,y)*, the magnitude *m(x,y)* and orientation *θ(x,y)* of the gradient can be calculated as;

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \qquad (2.23)$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \qquad (2.24)$$

Orientations around a keypoint are added up to a histogram for each keypoint. The histogram has 36 bins for 360° range. Each sample is weighted by the gradient magnitude and also by a Gaussian function with a scale 1.5 times of the scale of the keypoint, and added to the histogram. Peaks of the histogram are taken as the dominant direction of the gradient. Moreover, the local peaks that are larger than 80% of the highest peak are also accepted and a new keypoint is created for them. Therefore, there can be keypoints that have the same location but different orientations, which contribute considerably to matching stability. As a last step, a parabola is fit to the 3 histogram values around the peak to achieve better localization.

2.2.4 Constructing a keypoint descriptor

Up to this point, the keypoints are detected based on the local extrema, and location, scale and orientation values are assigned to each of them. These parameters provide a local 2D coordinate system for the local image region, thus provide translation, scale and rotation invariance. Next, a distinctive descriptor should be assigned to each keypoint to characterize it, which can later be used for matching.

The first step is to compute the local image gradient magnitudes and orientations around each keypoint. For this operation, the Gaussian blurred image with the scale of the keypoint is used, providing scale invariance. For rotation invariance, the gradient orientations and descriptor coordinates are rotated to the assigned keypoint orientation.

In Figure 2.5, (a) shows an example of local gradient magnitudes and orientations. The magnitudes of the gradient are weighted by a Gaussian with scale 1.5 times the width of the descriptor window, and added up to a histogram to construct the descriptor. A sample descriptor can be seen in (b). Here, the gradient samples are taken in an 8x8 neighborhood. Then 4x4 regions are used for histogram computation and eight direction bins are used for histogram entries.

The gradient samples contribute to both of its adjacent histogram entries with a weight factor of *(1-d)* where *d* is the distance between the original orientation of the sample and the central value of the bin.

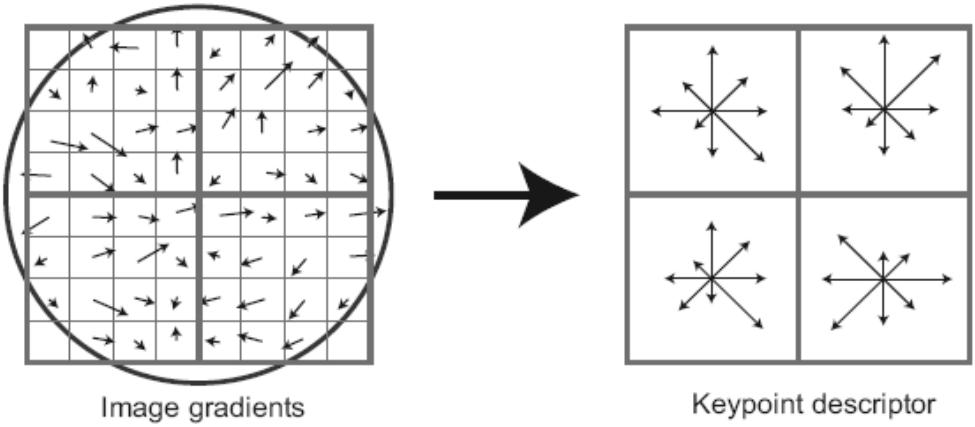


Figure 2.5 – Construction of the keypoint descriptor

This figure shows the construction of keypoint descriptor. (a) shows the local image gradient magnitudes and orientations in a 8x8 neighborhood of the keypoint. The circle represents a Gaussian weighting applied to the gradient magnitudes. (b) shows the histograms of eight directions, each created by 4x4 regions. The magnitudes of the histogram entries correspond to the sum of the gradient magnitudes near that direction [18].

After creating the histograms, each histogram entry is taken as an element of the descriptor vector. In Figure 2.5, the gradient computations are performed on an 8x8 region, resulting in a 2x2 histogram array, thus producing a 32 dimension descriptor vector. Though, the implementation of Lowe utilizes 16x16 neighborhoods, resulting in a 4x4 histogram arrays and descriptor vectors of length 128.

In this thesis, the SIFT++ code by A. Vedaldi is utilized as the SIFT code [30].

2.3 Random Sample Consensus (RANSAC)

Random Sample Consensus (RANSAC) is an algorithm to fit a mathematical model to a set of observed data which contains "a significant percentage of gross erros" [6]. It is a non-deterministic algorithm that iteratively tries to find a model that fits to as much of the data as possible. The probability of having a realistic model increases as the number of iterations increase.

The inputs to the RANSAC algorithm are:

- the observed data,
- a parametric model to be fit to the observed data (the model parameters will be estimated by RANSAC),
- *n*: the number of data points that will be used for the initial estimate,
- *t*: the error threshold to decide if a point fits well to a model,
- *d*: the minimum number of inliers for a model to be accepted,
- *k*: the number of iterations.

The algorithm first selects *n* random points from the data set which are called *hypothetical inliers*. The model parameters are calculated from these hypothetical inliers. Then all the remaining points in the data set are tested against this predicted model. The points fitting well to the predicted model, that is, with an error less than a threshold *t*, are added to the hypothetical inliers set and this new set is named the *consensus set*. If the number of elements in the consensus set is larger than *d*, the model has a probability to be a reasonable one. At this step, the model parameters are recomputed using the elements in the consensus set since the former parameters were based on only the initial hypothetical inliers. To evaluate the accuracy of the model, the error of the inliers relative to the error is computed. If this error value is smaller than the current lowest error value, this model is saved as the current best model. Then another iteration is performed again starting with an initial random subset of the data set. The procedure

continues this way until *k* iterations are made. The current best model is updated every time a model with a lower error is found. When the iterations finish, the model saved as the current best model constitutes the estimate of the RANSAC algorithm.

The RANSAC parameters *n*, *t* and *d* are problem specific parameters that should be given to the algorithm. On the other hand, the parameter *k*, i.e., the number of iterations, can be estimated. As stated above, the algorithm starts with selecting *n* random points from the data set. Let the probability of selecting an inlier from the data set be w,

$$w = \frac{number\ of\ inliers\ in\ the\ data\ set}{total\ number\ of\ elements\ in\ the\ data\ set} \tag{2.25}$$

In many cases, the number of inliers in the data set may not be known but a rough estimate can be used. Using this value, the probability of all the selected *n* points be inliers becomes w^n and the probability of having at least one outlier in selected *n* points becomes $1-w^n$. In *k* iterations, the algorithm will select at least one outlier in all k iterations with a probability of $(1-w^n)^k$, and will have at least one iteration with all inliers with probability $1-(1-w^n)^k$. Estimating the model from a data set consisting of all inliers will give a reasonable result, so this value also represents the probability of RANSAC finding a good model. If we set it to a constant value, in other words to the desired probability, we obtain:

$$P = 1 - \left(1 - w^n\right)^k \tag{2.26}$$

$$k = \frac{\log\left(1-P\right)}{\log\left(1-w^n\right)} \tag{2.27}$$

This *k* value is the maximum number of iterations required to find a fine model with probability *P*.

CHAPTER 3

A ROBUST AND EFFICIENT SALIENT REGION DETECTOR AND TRACKER FOR VIDEO SEQUENCES

The focus of this thesis is to construct a reliable and computationally efficient salient region detector and tracker using the visual attention concept. The algorithm should be suitable for outdoor video sequences. Since the outdoor environments are normally uncontrolled, the *Visual Attention* algorithm is promising since it autonomously selects the locations that it will concentrate on.

This chapter is organized as follows. First, section 3.1 describes the video data used in the tests and section 3.2 specifies the evaluation criteria to interpret the tests. The remaining sections propose methods towards obtaining a reliable and efficient salient region detector and tracker. In section 3.3, the Visual Attention algorithm is used alone and its behavior is analyzed. In section 3.4, the SIFT features are used alone as a region tracker to observe its performance. Then, in section 3.5, SIFT features combined with the Visual Attention to achieve better performance. Section 3.6 considers using RANSAC algorithm with matched SIFT points to get more robust results. Section 3.7 studies the idea of region-based-processing to decrease the augmented computational load. Section 3.8 suggests limiting the attention channels to further decrease the complexity.

3.1 Test Data

The test data used by the tests consist of four videos which are called *Shaky*, *Planar*, *Blurred* and *CarPark*.

- *Shaky* is an outdoor video sequence recorded in the front yard of a building. Some bright and colored objects are distributed throughout the scene to act as visually attracting objects. The environment is motionless. The video is recorded by holding the camera by hand and 'shaking' it with a pattern that represents legged motion. Some sample frames can be observed in Figure 3.1.

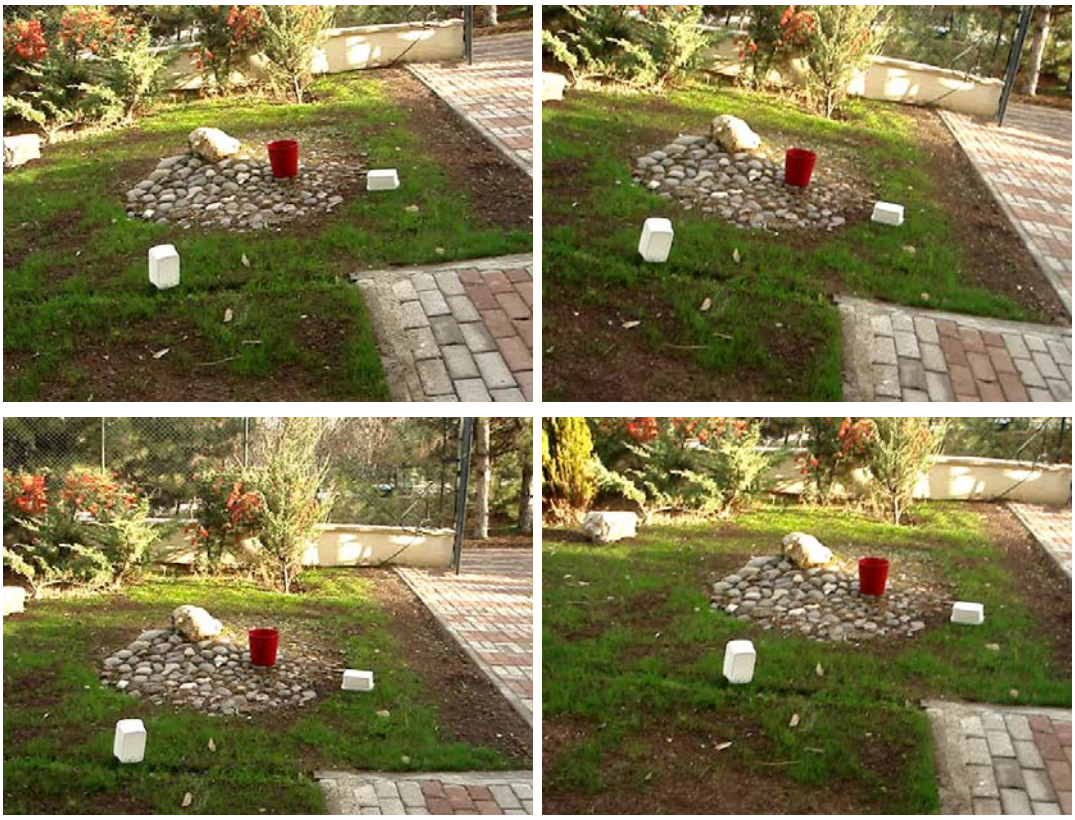


Figure 3.1 – Sample frames from the video *Shaky*

Some sample frames from the video *Shaky* which represents legged motion.

- *Planar* is recorded in the same environment. The camera is held by hand and moved slowly. The motion of the camera in this video has a feature as follows: First the tentative salient objects in the environment have been detected. Then the camera is moved such a way that the trajectories of the salient objects do not overlap throughout the video. This video is used to evaluate the tracking performances of various algorithms by comparing the actual trajectories of the objects with the computed trajectories. Since the paths of the objects do not overlap, this video is called *Planar* for the ease of comprehension. Some sample frames from the video *Planar* can be observed in Figure 3.2.

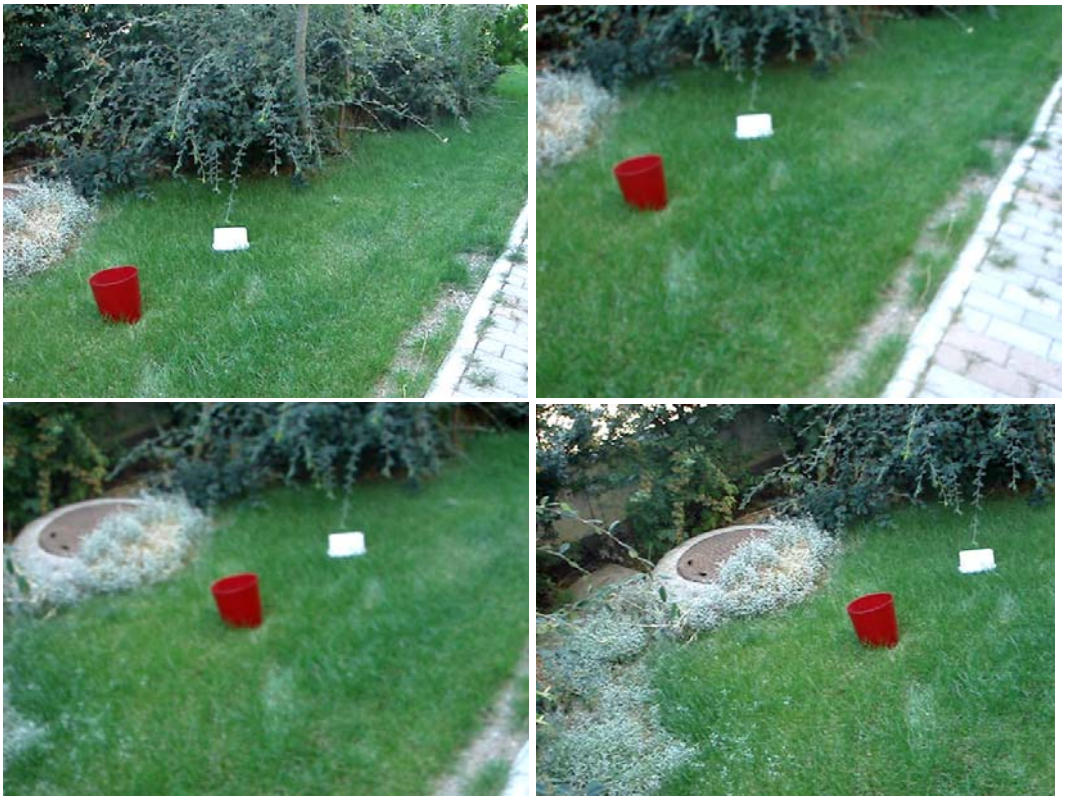


Figure 3.2 – Sample frames from the video *Planar*

Some sample frames from the video *Planar.* This is the test video for the evaluation of tracking capabilities of the algorithms.

- *Blurred* is recorded in the same environment. The camera is held by hand and moved very fast in order to obtain images which are highly motion blurred. Motion blur is an inevitable factor in many video sequences, as seen in the camera outputs of mobile robots. The aim of this sequence is to test the robustness of the algorithms against motion blur. Some sample frames from the video *Blurred* can be observed in Figure 3.3.

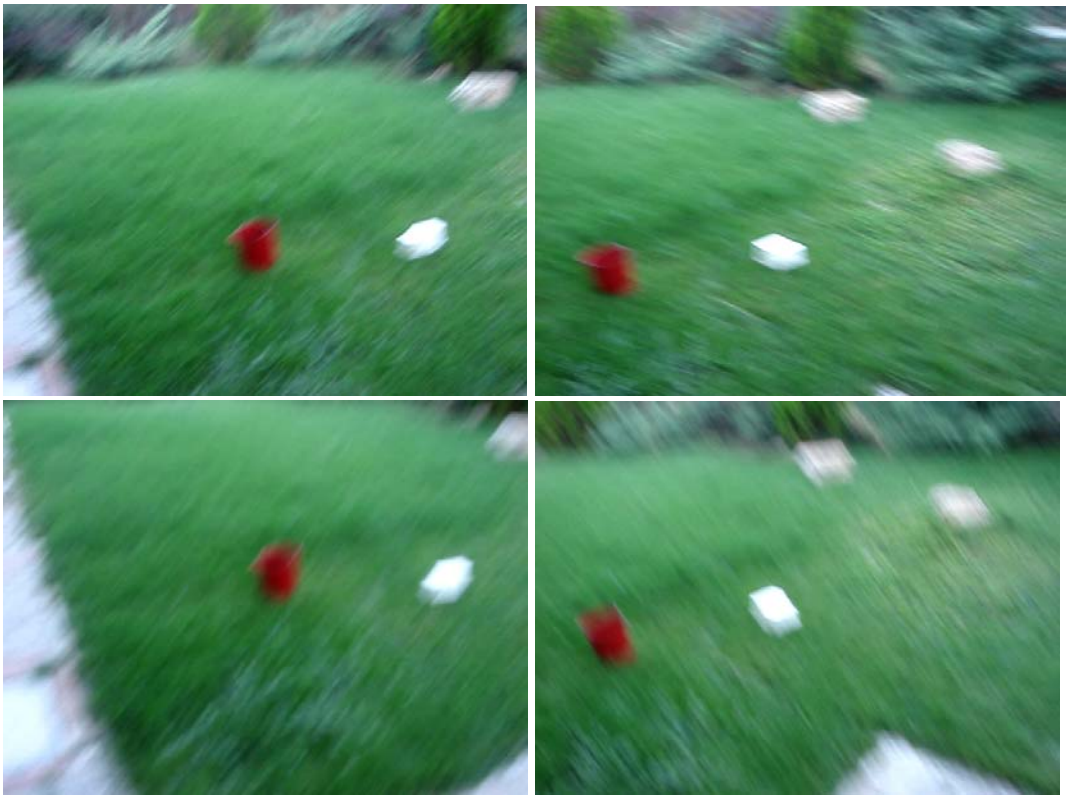


Figure 3.3 – Sample frames from the video *Blurred*

Some sample frames from the video *Blurred.* It is used for testing robustness of the algorithms against motion blur.

- *CarPark* is recorded in a small car park. The camera is held by hand and moved again simulating a legged motion. In this sequence there are no intentionally distributed salient objects around the scene so the environment is totally uncontrolled. It is used to test the behaviors of the algorithms in real life scenarios. Some sample frames from the video *CarPark* can be observed in Figure 3.4.

Figure 3.4 – Sample frames from the video *CarPark*

Some sample frames from the video *CarPark.* This video represents a real life scenario.

In the experiments of this chapter, the test videos *Shaky* and *Planar* are used. *Shaky* is used to measure the errors, so the performances, of the algorithms while *Planar* is used to visualize the actual and computed positions of the salient regions on the image in order to evaluate the tracking capabilities.

3.2 Evaluation Criteria

Performing some tests, we need to measure the tracking capabilities and the efficiencies of the methods. Intuitively, the evaluation criteria compare the algorithms based on their,

- ability to find discriminative features,
- reproducibility of these features,
- ability to track those features over time in consecutive frames,
- computational efficiency.

By taking these into account, formal evaluation criteria are produced.

First, a reference data is obtained for each test video. Let *N* be the number of frames in the sequence and *M* be the number of reference points in each frame. The m^{th} point in n^{th} frame can be represented as $\mathbf{x}_{m,n}(x, y)$, where $m \in [0, M-1]$, $n \in [0, N-1]$ and *x and y* are the horizontal and vertical locations of the point respectively. First, *M* reference points are chosen from the first frame. Then for each following frame, the point corresponding to the same location is found manually. The displacement vector between subsequent reference points is calculated as:

$$\Delta\mathbf{x}_{m,n} = \mathbf{x}_{m,n} - \mathbf{x}_{m,n-1} \tag{3.1}$$

These inter-frame displacement values are calculated by hand and recorded. They give the exact displacements in the image; hence constitute the ground-truth data. The results of the following tests are compared with these values.

When testing each algorithm, for the first frame, the coordinates of the detected objects/points are recorded. Then for each following frame, the displacement vectors are calculated by the algorithm. Finally, the calculated displacement vectors are compared with the reference data.

This procedure can be formalized as follows:

$\mathbf{x'}_{m,n}(x, y)$: The computed coordinate of the m^{th} point in the n^{th} frame,

The computed inter-frame displacement vector $\Delta\mathbf{x'}_{m,n}$ can be computed as:

$$\Delta\mathbf{x'}_{m,n} = \mathbf{x'}_{m,n} - \mathbf{x'}_{m,n-1} \tag{3.2}$$

The error between the calculated displacement vector and the ground-truth displacement vector can be calculated as:

$$E_{\Delta\mathbf{x}_{\mathbf{m},\mathbf{n}}} = \left|\Delta\mathbf{x}'_{m,n} - \Delta\mathbf{x}_{m,n}\right| \tag{3.3}$$

To have a measure of the overall error of the algorithm, one of the M reference points is selected and the mean and variance of the inter-frame errors is calculated:

$$\mu_E = \frac{\sum_{n=1}^{N} E_{\Delta\mathbf{x}_{m,n}}}{N} \tag{3.4}$$

$$\sigma_E = \sqrt{\frac{\sum_{k=1}^{N}\left(E_{\Delta\mathbf{x}_{m,n}} - \mu_E\right)^2}{N}} \tag{3.5}$$

These values give a measure of the error between the ground-truth data and the data calculated by the algorithms. In the experiments in chapters 3 and 4, μ_E is used as the measure of the performance of the algorithm. (While calculating the

error $E_{\Delta \mathbf{x}_{\mathbf{m},\mathbf{n}}}$, the error in x direction E_x, and the error in y direction E_y are computed separately and then combined according to $E = \sqrt{E_x^2 + E_y^2}$.)

At this instant, having a measure of performance, we need to measure the efficiency of the algorithms as well. To measure the computational efficiency, the average time spent for processing a frame is measured and recorded for each algorithm. The total time to process the video sequence, Δt_T can be computed as:

$$\Delta t_T = t_{end} - t_{start} \tag{3.6}$$

where t_{start} is the local time when the algorithm starts, and t_{end} is the local time when the algorithm quits.

Average time spent to process a frame, Δt_{avg}, can be easily computed as:

$$\Delta t_{avg} = \frac{\Delta t_T}{N} \tag{3.7}$$

where N is the total number of frames processed.

3.3 Using Visual Attention alone for salient region detection and tracking in video sequences (*VAonly*)

The idea is using the *Visual Attention* algorithm alone as a salient region detector and tracker. From now on, this method will be called *VAonly*. The inputs of the algorithm are the video sequence and the number of salient regions to track, *M*.

The flow of the algorithm can be visualized in Figure 3.5. For each frame, the algorithm is run and the *M* salient region positions on the image are found. The algorithm provides the center points of the visually salient areas; that is $\mathbf{x}_{m,n}(x,y)$

for the m^{th} salient region in n^{th} frame, where $m \in [0, M-1]$, $n \in [0, N-1]$, and N is the total number of frames. Then, a saliency feature vector **f** is extracted for each salient region which will later be used for matching purposes.

The details of 'saliency feature extraction' are given in Figure 3.6 (a). Remember that *Visual Attention* algorithm creates three conspicuity maps that display the visually attractive areas of the image in terms of intensity, color and orientation differences (see section 2.1.2). The feature vector $\mathbf{f}_{m,n}$ for the m^{th} salient region in n^{th} frame is defined as:

$$\mathbf{f}_{m,n} = \left(\frac{I_{m,n}}{I_{m,n}+C_{m,n}+O_{m,n}}, \frac{C_{m,n}}{I_{m,n}+C_{m,n}+O_{m,n}}, \frac{O_{m,n}}{I_{m,n}+C_{m,n}+O_{m,n}}\right) \quad (3.8)$$

where $I_{m,n}$ is the sum of the values in the intensity conspicuity map in the 16x16 region around $\mathbf{x}_{m,n}$. Similarly, $C_{m,n}$ and $O_{m,n}$ are computed from the color and orientation conspicuity maps respectively.

Notice that all the three components are normalized. The elements of feature vector represent the contribution of each channel to the saliency of the region (for instance, a salient region may have 30% of its saliency coming from intensity channel, 10% from color and 60% from orientation channel).

A salient region can be defined as $SR_{m,n}$ = ($\mathbf{x}_{m,n}$, $\mathbf{f}_{m,n}$) where **x** is its spatial location and **f** is its feature vector.

Having the salient regions defined, they should be added to the trajectories that track the salient regions over frames. If current frame is the first frame of the video sequence, then *M* initial trajectories are created from the *M* salient regions and the processing loop continues with the next frame. If it is not the first frame, there must be *M* active trajectories coming from the previous frames. The regions found on this frame are tried to be added to one of the active trajectories using the procedure given in Figure 3.6 (b). Let $SR_{m,n}$=($\mathbf{x}_{m,n}$,$\mathbf{f}_{m,n}$) be the salient region to be added to a trajectory, and H_j=($\mathbf{x}_j$,$\mathbf{f}_j$) be the head element of *trajectory-j*. $SR_{m,n}$ is added to *trajectory-j* if the two conditions hold:

$$\left|\mathbf{x}_{m,n} - \mathbf{x}_j\right| < \varepsilon_{\mathbf{x}} \tag{3.9}$$

$$\left|\mathbf{f}_{m,n} - \mathbf{f}_j\right| < \varepsilon_{\mathbf{f}} \tag{3.10}$$

where $\varepsilon_{\mathbf{x}}$ and $\varepsilon_{\mathbf{f}}$ are the threshold values to be determined in accordance with the data.

Equation (3.9) ensures that the salient region is in the neighborhood of the candidate trajectory. Since the frames are successive elements of a video sequence, the elements of the trajectory must be in continuity. And equation (3.10) ensures that the feature vectors of $SR_{m,n}$ and H_j are similar. A salient region can be added to at most one trajectory, and a trajectory can include at most one salient region from a frame.

After appending the salient regions of current frame to their trajectories, the trajectories that have not been extended in current frame are closed. If less than M active trajectories are left as a result of this action, new trajectories are started using the unassigned salient regions of current frame.

Using the active trajectories, the inter-frame displacement vectors are computed for each tracked region. These displacement vectors are recorded to be used for comparison with the actual displacement vectors, which are computed by hand (see section 3.2). The processing loop continues up to the end of the video.

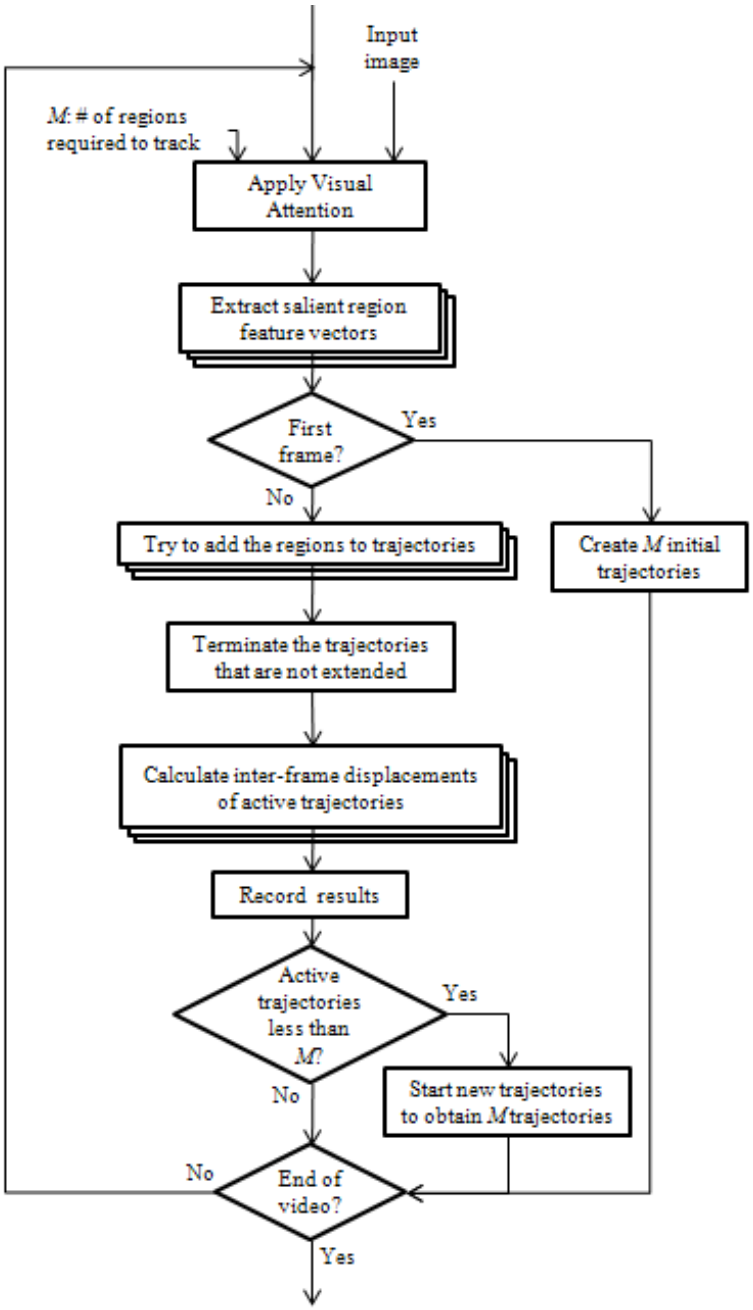


Figure 3.5 – Flowchart of *VAonly* algorithm

The Visual Attention algorithm is applied to the input frames to locate the required number of salient regions. Then these regions are tracked over frames by forming a trajectory for reach tracked region. In each frame, the displacement vectors are computed for each trajectory and are compared with the actual displacement vectors to evaluate the tracking capability of the algorithm.

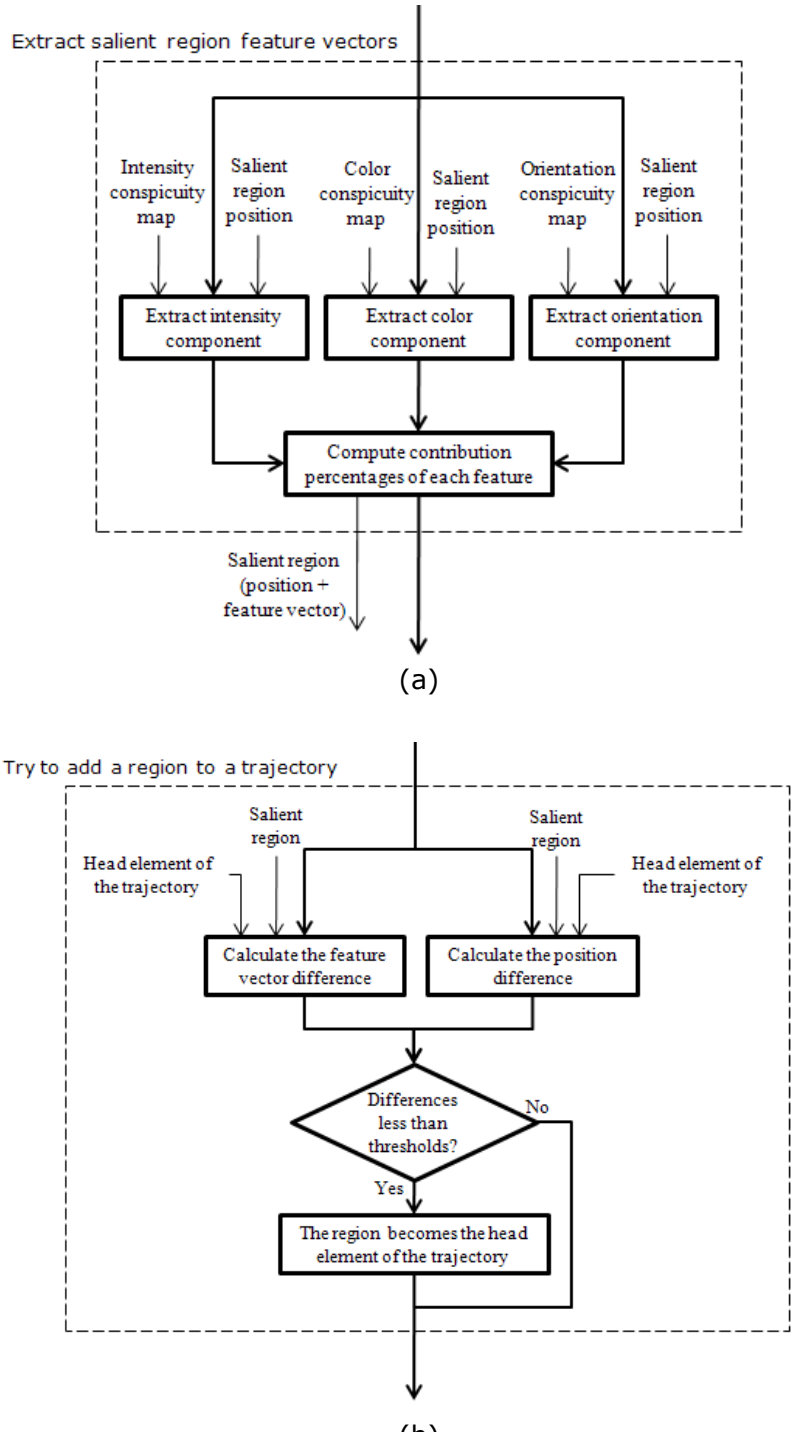


Figure 3.6 – Internal diagrams of some blocks of VAonly

(a) Extracting salient region feature vectors

(b) Adding a salient region to a trajectory

To observe the behavior of the proposed algorithm, experiments are performed. Section 3.3.1 explains the experiments conducted with different test videos.

3.3.1 Experimental Results

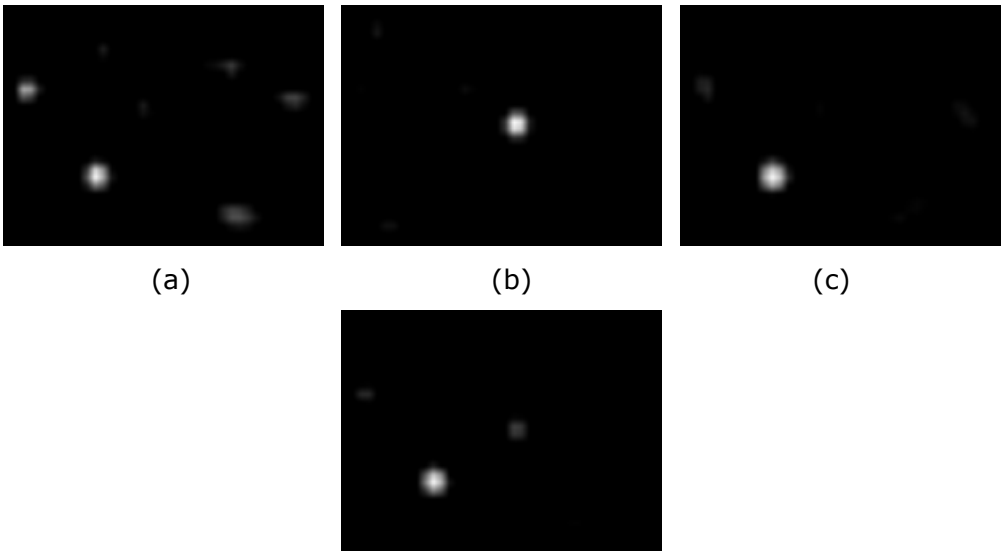
The test videos used in this experiment are:

- *Shaky*, which is an outdoor video sequence with shaky motion,
- *Planar*, which is an outdoor video sequence with non-overlapping motion,

The test videos employed here are described in detail in section 3.1.

The performance of the algorithm can best be evaluated by observing the inter-frame displacement vectors calculated by the algorithm with respect to the ground-truth inter-frame displacement vectors. The video *Shaky* is used to measure the error of the algorithm in calculating the inter-frame displacement vectors. That's why in this run, *M*, which is the number of salient regions to track, is entered as 1. A sample frame of *Shaky* sequence with the detected salient region position can be seen in Figure 3.7.

Figure 3.7 – The detected salient region position in a frame of *Shaky*



(a) (b) (c)

(d)

Figure 3.8 – Conspicuity and saliency maps of the frame in Figure 3.7.
(a) intensity conspicuity map (b) color conspicuity map
(c) orientation conspicuity map (d) saliency map

Figure 3.8 displays the conspicuity maps and the saliency map of the sample frame of *Shaky* where (a) is the intensity conspicuity map, (b) is the color conspicuity map, (c) is the orientation conspicuity map and (d) is the saliency map. Using these maps, the saliency feature vector of the most salient region (marked in Figure 3.7) is found as [56.2, 0.0, 43.8] which means that 56.2% of the region's saliency comes from intensity discrepancy while 43.8% comes from orientation discrepancy.

Figure 3.9 displays the graph of calculated versus actual displacement vectors for the video sequence *Shaky*. In this figure, the x-axis is the processed frame and the y-axis is the displacement between i^{th} and $(i+1)^{th}$ frame in pixel values. The part (a) of the graph is for the x component of the motion while the part (b) is for the y component. When Figure 3.9 is observed, it can be seen that the computed patterns only roughly follow the ground-truth data. The error is too large and the

results are extremely fluctuating. This is mainly because of the low repeatability of the points found by the *Visual Attention* algorithm. *Visual Attention* roughly gives the salient region positions but it has an algorithmic noise that results in slightly different positions in each run[1]. As a result of this poor repeatability of *Visual Attention* algorithm, the error of *VAonly* algorithm is quite large in all runs.

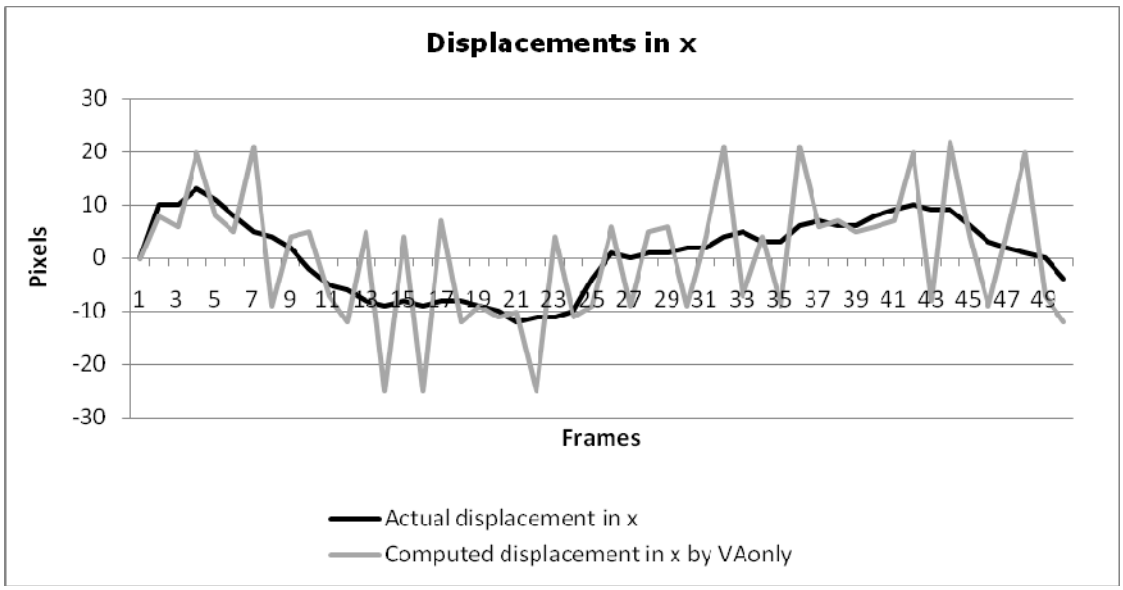
To have a better idea of the algorithm's behaviour, the errors between the actual and computed inter-frame displacement vectors are calculated. Figure 3.10 displays the error between the graphs in Figure 3.9, i.e., between the ground-truth displacement vectors and the ones calculated by the *VAonly* algorithm for the video sequence *Shaky*. The (a) part of the graph is the error in x-component of the motion, the (b) part is the error in y-component and the (c) part is the total error calculated by,

$$E_T = \sqrt{E_x^2 + E_y^2} \tag{3.11}$$

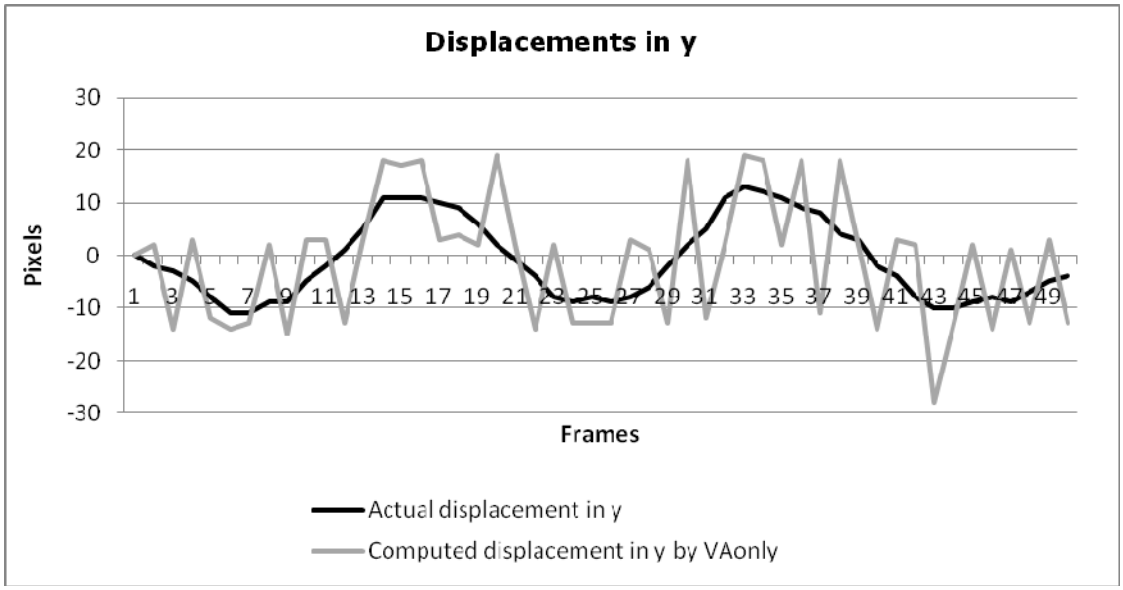
where E_x and E_y are the errors in x and y dimensions respectively.

The errors are appearantly very high, varying approximately between ±20 pixels in x and y directions. The errors are zero-mean since there is no bias in any step of the algorithm.

[1] The utilized visual attention code by iLab Neuromorphic Toolkit [10] can detect the salient positions in 16 pixel resolution and adds a random noise between +8 and -8 pixels to each detected position in order to achieve pixel resolution.



(a)

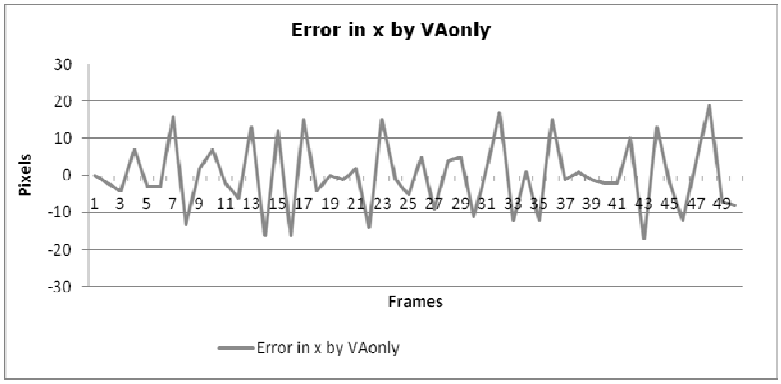


(b)

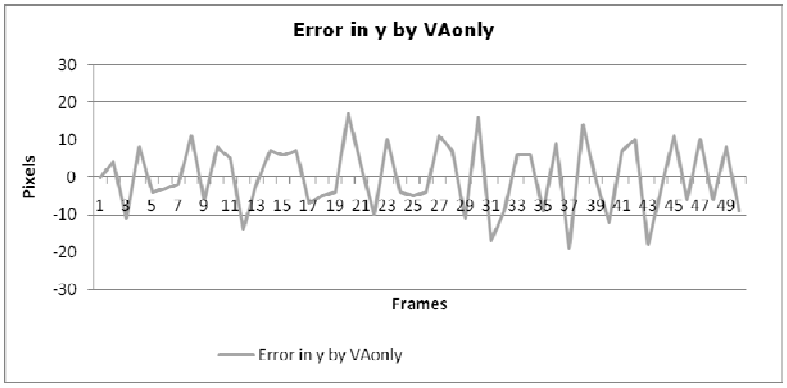
Figure 3.9 – Inter-frame displacement vectors computed by *VAonly* vs. ground truth data for *Shaky*

(a) x-component of the motion (inter-frame displacements in x direction)

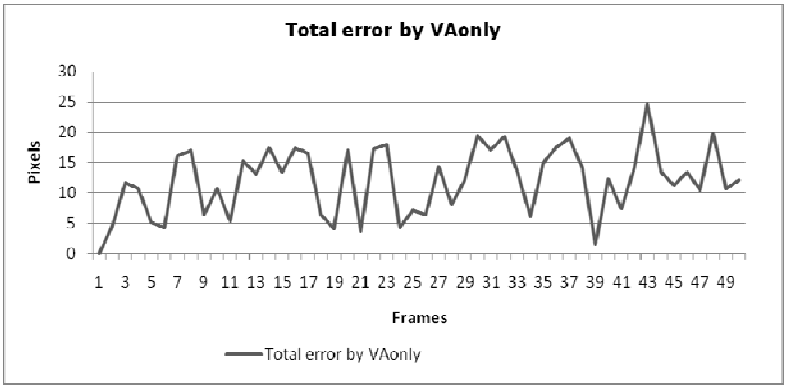
(b) y-component of the motion (inter-frame displacements in y direction)



(a)



(b)



(c)

Figure 3.10 – The error between the actual inter-frame displacement vectors and the ones calculated by *VAonly* for *Shaky*

(a) Error for the x-component of the motion

(b) Error for the y-component of the motion

(c) Total error that equals the vector sum of (a) and (b)

The mean and the variance of the total error are computed according to the equations (3.4) and (3.5) respectively. As mentioned in section 3.2, the evaluation criterion regarding the performance is the mean of the total error. Below are the mean and variance of the total error for the video *Shaky*.

μ_E = 12.13 pixels
σ_E = 5.42 pixels

The measure for computational cost, i.e., the average processing time per frame, is calculated according to the equation (3.7). Below is the average computational time per frame for video *Shaky*.

t_{avg} = 3,24 s

The performance and the computational cost of the algorithms are the two major parameters for evaluation. A good visualization can be made by placing the algorithm on the performance-cost space. Figure 3.11 shows a graph of computational cost versus algorithm performance, where x-axis is the cost and y-axis is the performance. Computational cost of the algorithm is computed as the average time spent for processing a frame, whereas the performance is taken as the mean of the total error between the actual and computed inter-frame difference vectors. In this graph, it is better to be close to the origin since in this region both the computational cost and the error of the algorithm is low, thus providing a good performance with a low cost.

The position of *VAonly* in the graph of Figure 3.11 is computed using the *Shaky* sequence. This video is the reference data set for the future experiments. It can be seen that the position of *VAonly* in performance-cost space is quite far from the origin in both coordinates, which indicates a low performance with a high cost.

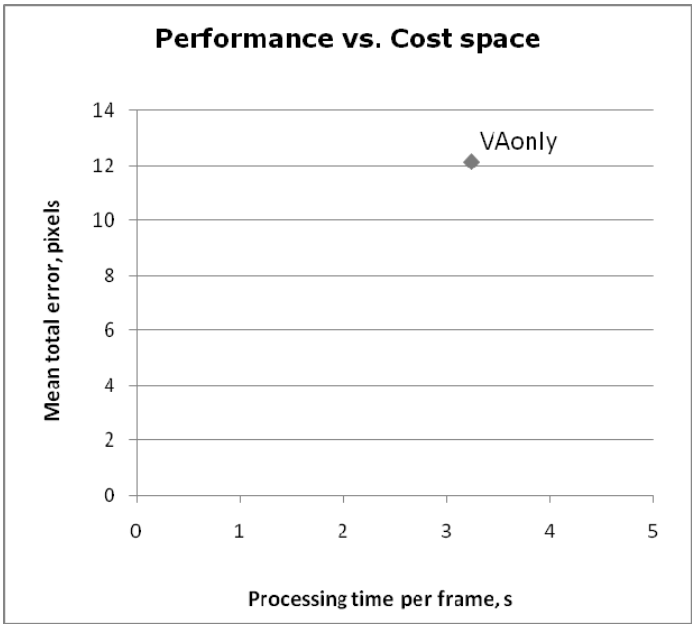


Figure 3.11 – The position of *VAonly* algorithm in "performance-computational cost" space
x-axis is the average time spent for processing a frame and y-axis is the mean of the total error between the displacement vectors computed by the algorithm and the actual ones. The region close to the origin is the sweet stop since this region represents good performance and low cost.

To have an idea of the practical error of the algorithm, the inter-frame displacement vectors can be used to compute the object locations in pixel coordinates in each frame. Then these position values can be compared with the ground-truth object locations in pixel coordinates. To achieve this, an experiment is performed using the *Planar* sequence. The reason of using this video is that, in other test videos the salient object positions overlap a lot throughout the video, thus resulting in unrecognizable graphs. Figure 3.12 shows the actual and computed positions of the tracked salient region for the video *Planar*. In this run, *M*, which is the number of salient regions to track, is entered as 2.

Examining Figure 3.12, it is seen that the computed positions give an idea about the locations of the regions, but the trajectories are very irregular with high variations.

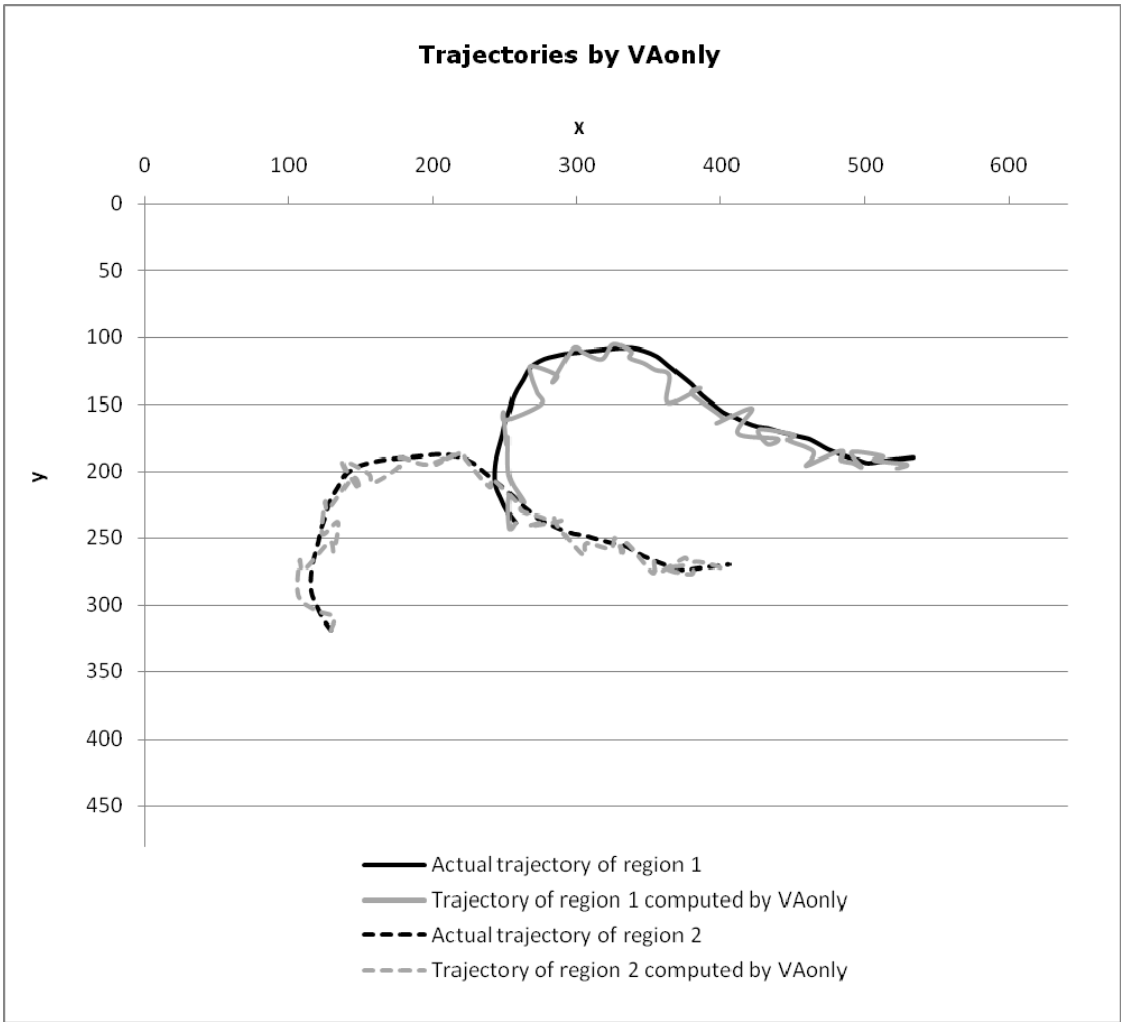


Figure 3.12 – The actual and computed positions of the tracked regions for *VAonly*

It can be seen that the error in the position of the salient object is approximately around 25 pixels. To give an idea, an error of 50 pixels corresponds to an error of 25 centimeters in the location of the salient regions in this video, which are approximately 2 meters away from the camera.

Summing up the experimental results of the *VAonly* algorithm, the error of the algorithm is too large to be used in practical applications. Moreover, the computational cost is also quite large since *Visual Attention* is a very costly algorithm especially when applied to large frames. If we concentrate on improving the performance first, it can be deduced that the performance of *VAonly* can be increased if some more robust features are computed and used for the visually attractive region. These features could possibly be Harris corners [9], SIFT features [18], SURF features [2] and so forth. Being highly robust and distinctive as well as being well accepted in the literature, SIFT features turned out to be our choice.

3.4 Using SIFT alone as a region tracker in video sequences *(SIFTonly)*

The idea here is to use SIFT to track regions in video sequences. This method will be called *SIFTonly*. The SIFT algorithm is applied to the frame and the SIFT points are extracted. Then these points are tried to be matched to the SIFT points of the previous frame.

SIFT point matching is performed according to the method of Lowe [18]. As described in section 2.2, each SIFT point has a feature vector of dimension 128. Assume that we are trying to match two SIFT point groups *group-1* and *group-2* having *n1* and *n2* elements respectively. Let the i^{th} point of *group-1* be $P_{1\text{-}i}$. To match $P_{1\text{-}i}$ to a point in *group-2*, the vector distance between the feature vector of $P_{1\text{-}i}$ and the feature vector of each point in *group-2* are calculated. The elements in *group-2* are sorted according to these vector distance values, which show its similarity with $P_{1\text{-}i}$. Then, this sorted list of *group-2* is analyzed. Let the head of the sorted list be $P_{2\text{-}1}$ and the second element in the list be $P_{2\text{-}2}$. Notice that $P_{2\text{-}1}$ is the element that is most similar to $P_{1\text{-}i}$ while $P_{2\text{-}2}$ is the second most similar one. If $P_{2\text{-}1}$ is 60 percent more similar to $P_{1\text{-}i}$ than $P_{2\text{-}2}$, then $P_{2\text{-}1}$ is accepted as a match. If the

difference between the similarity values of the head and second elements is less than 60 percent, then it is said that no match is found for $P_{1\text{-}i}$.

After matching the SIFT points of current frame to the SIFT points of previous frame, inter-frame displacement vectors should be computed. An inter-frame displacement vector can be computed for each matched SIFT point pair. To obtain a single displacement vector, these values are averaged. Then these computed displacement vectors are recorded to be compared with the actual ones. The processing loop continues up to the end of the video. The flow of the algorithm can be visualized in Figure 3.13.

Section 3.4.1 presents the experimental results of *SIFTonly*.

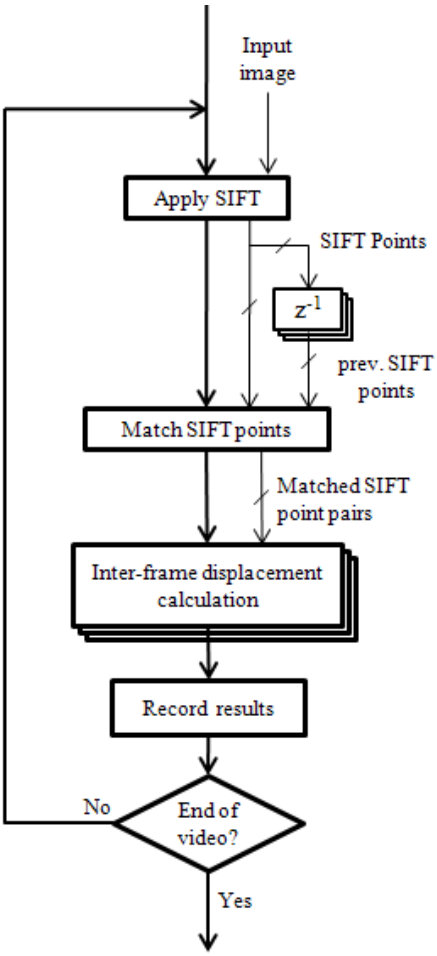


Figure 3.13 – Flowchart of *SIFTonly* algorithm

SIFT is applied to the input frames to extract the local stable features. Then the SIFT points of current frame are tried to be matched to the SIFT points of previous frame. The inter-frame displacement vectors are calculated using the matched SIFT point pairs. These values are recorded and the processing continues in this way till the end of the video.

3.4.1 Experimental Results

The test videos used in this experiment are *Shaky* and *Planar*. Similar to section 3.3.1, *Shaky* is used to evaluate the error and cost of the algorithm, while *Planar* is used to evaluate the tracking trajectories.

In this experiment, the inter-frame displacement vectors will be calculated using the SIFT points; however, the displacement values definitely depend on the depth of the region. Thus, the frames of the test videos cannot be fed to *SIFTonly* as full frames. To overcome this issue, first, a region with relatively uniform depth is segmented from the input frames by hand.

For a typical frame of the video *Shaky*, SIFT algorithm extracts points in the order of thousands. A uniform depth region from a sample frame of *Shaky* can be seen in Figure 3.14 with SIFT points marked as red points.

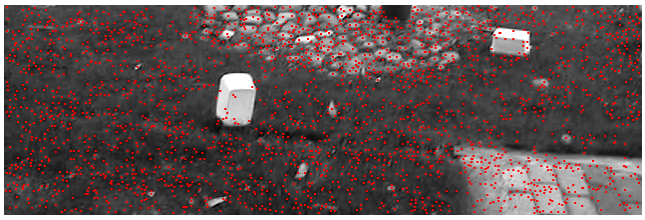
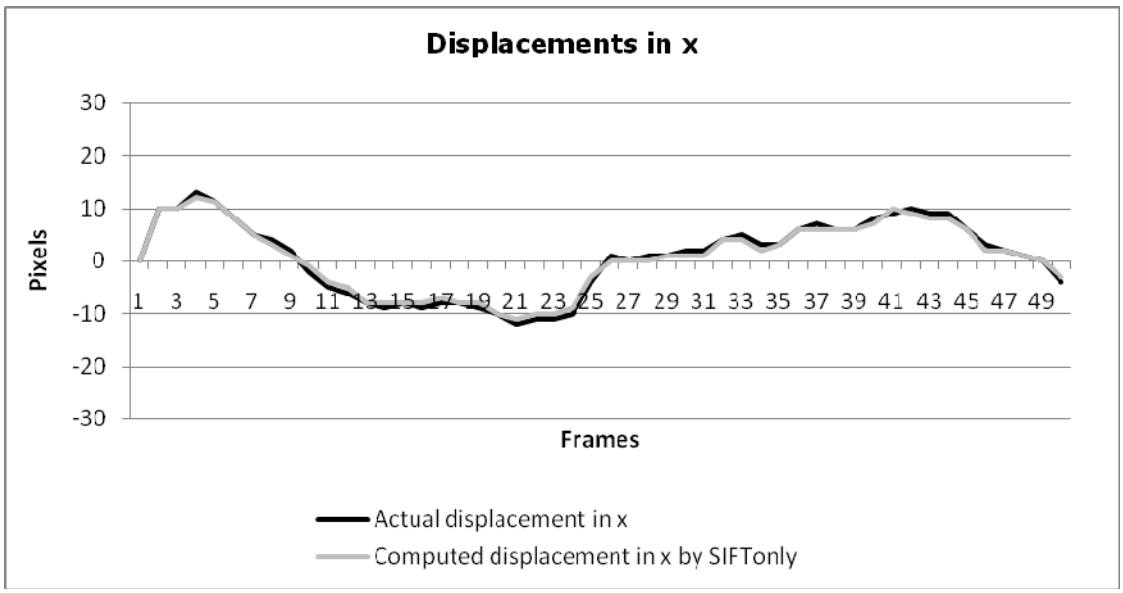
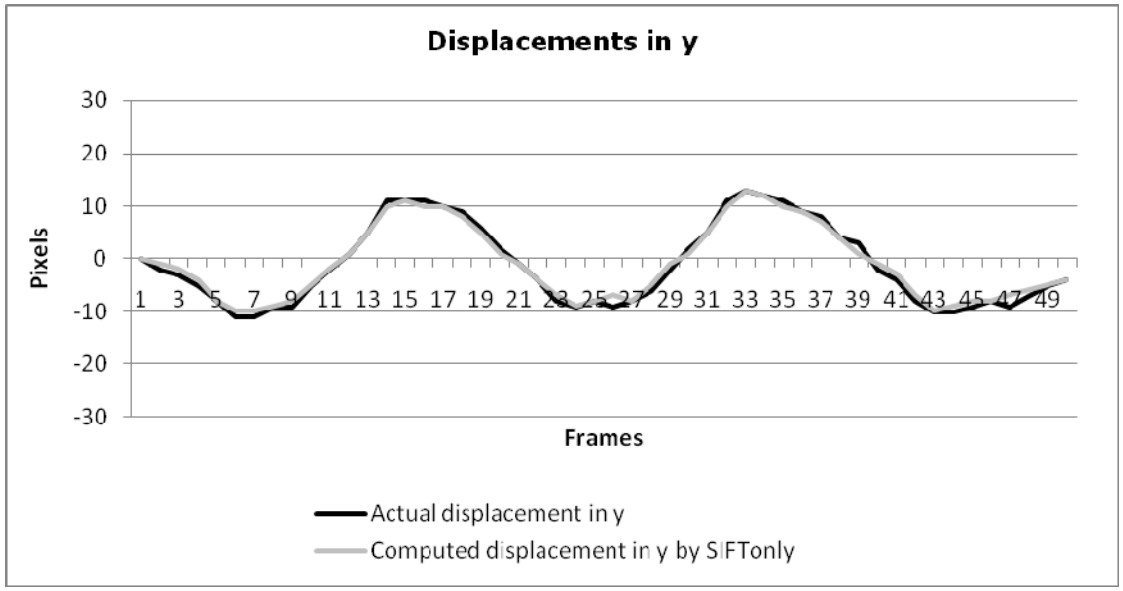


Figure 3.14 – SIFT points on a frame of *Shaky*
SIFT algorithm extracted 3017 points in this region.

Figure 3.15 is the plot of computed and actual inter-frame displacement vectors and Figure 3.16 is the plot of errors. In all graphs, pixel units are employed. The (a) parts of the figures are for the x-component of the motion while the (b) parts are for the y-component. In the error graphs, the (c) parts represent the total error computed by the equation (3.11).



(a)

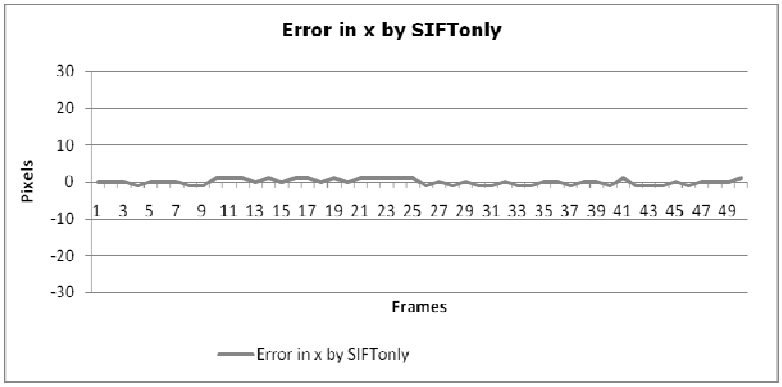


(b)

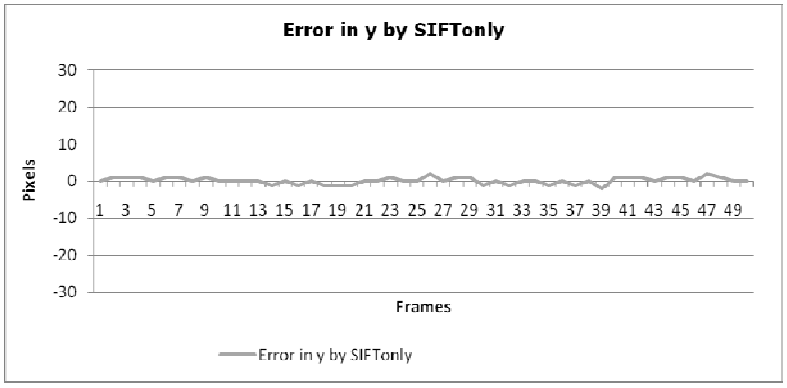
Figure 3.15 – Inter-frame displacement vectors computed by *SIFTonly* vs. ground truth data for *Shaky*

(a) x-component of the motion (inter-frame displacements in x direction)

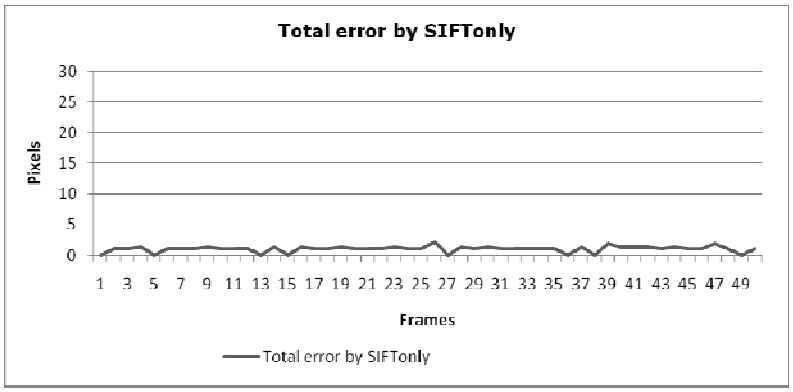
(b) y-component of the motion (inter-frame displacements in y direction)



(a)



(b)



(c)

Figure 3.16 – The error between the actual inter-frame displacement vectors and the ones calculated by *SIFTonly* for *Shaky*

(a) Error for the x-component of the motion

(b) Error for the y-component of the motion

(c) Total error that equals the vector sum of (a) and (b)

Observing Figure 3.15, the inter-frame displacement vectors computed by *SIFTonly* follow the ground truth data very well providing very precise results. Correspondingly the errors in Figure 3.16 are very low around 2 and 3 pixels.

The mean and variance of the error and the processing time per frame are computed by equations (3.4), (3.5) and (3.7) respectively.

$\mu_E = 1.03$ pixels

$\sigma_E = 0.51$ pixels

$t_{avg} = 11.9$ s

Using these values, *SIFTonly* can be placed in the performance - computational cost space as in Figure 3.17.

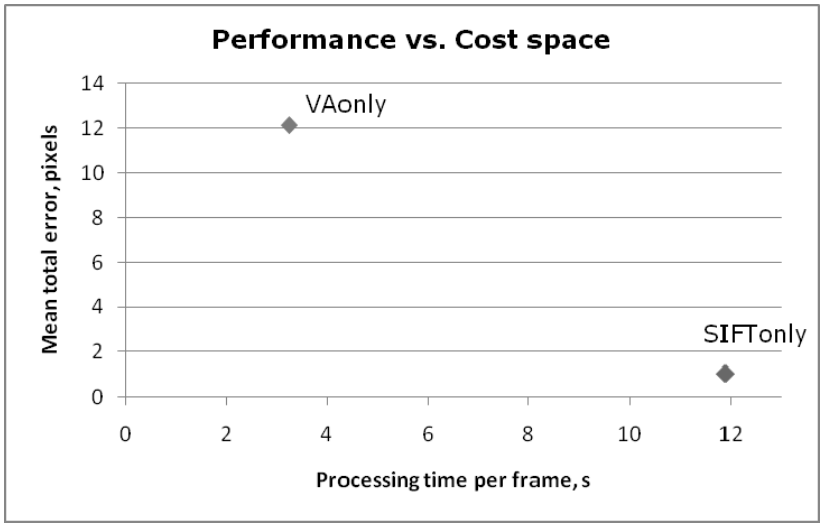


Figure 3.17 – The position of *SIFTonly* algorithm in "performance-computational cost" space
x-axis is the average time spent for processing a frame and y-axis is the mean of the total error between the displacement vectors computed by *SIFTonly* and the actual ones.

In Figure 3.17, it can be seen that *SIFTonly* demonstrate a much better performance with respect to *VAonly* but the computational cost is unacceptably high. The main reason of this is the enormous number (*~3000*) of extracted SIFT points which the makes the matching procedure very costly.

To view the error in computed object locations, the experiment with video *Planar* is used (see Figure 3.18). Since no region is defined in *SIFTonly* algorithm, the position values are computed for the center of the frame using the inter-frame displacement vectors. In Figure 3.18, the computed positions follow the ground data well but there is an accumulating error. The reason for that is the integration operation performed. Matching SIFT points and calculating the coordinate differences between the matched pairs gives us the derivative of the motion. To compute the positions in each frame, we integrate these values, thus accumulating the error.

Considering all the experimental results of *SIFTonly*, it can be said that this algorithm has a very low level of error but it is computationally too costly. A reason of this computational load is that SIFT itself is a very time-consuming algorithm. And another reason is the SIFT point matching procedure. Notice that, if we assume *n* SIFT points are extracted, the complexity of the matching procedure is $O(n^2)$; and knowing that the number of SIFT points per frame are in the order of thousands, the matching procedure is unsurprisingly very expensive. The simplest way of decreasing the time spent by SIFT and matching is decreasing the processed image size. By this way, both the SIFT algorithm will take less time and the matching procedure will be faster since the number of points will decrease significantly. Thus, it would be a useful idea to apply SIFT on specific regions of the image, namely the salient regions.

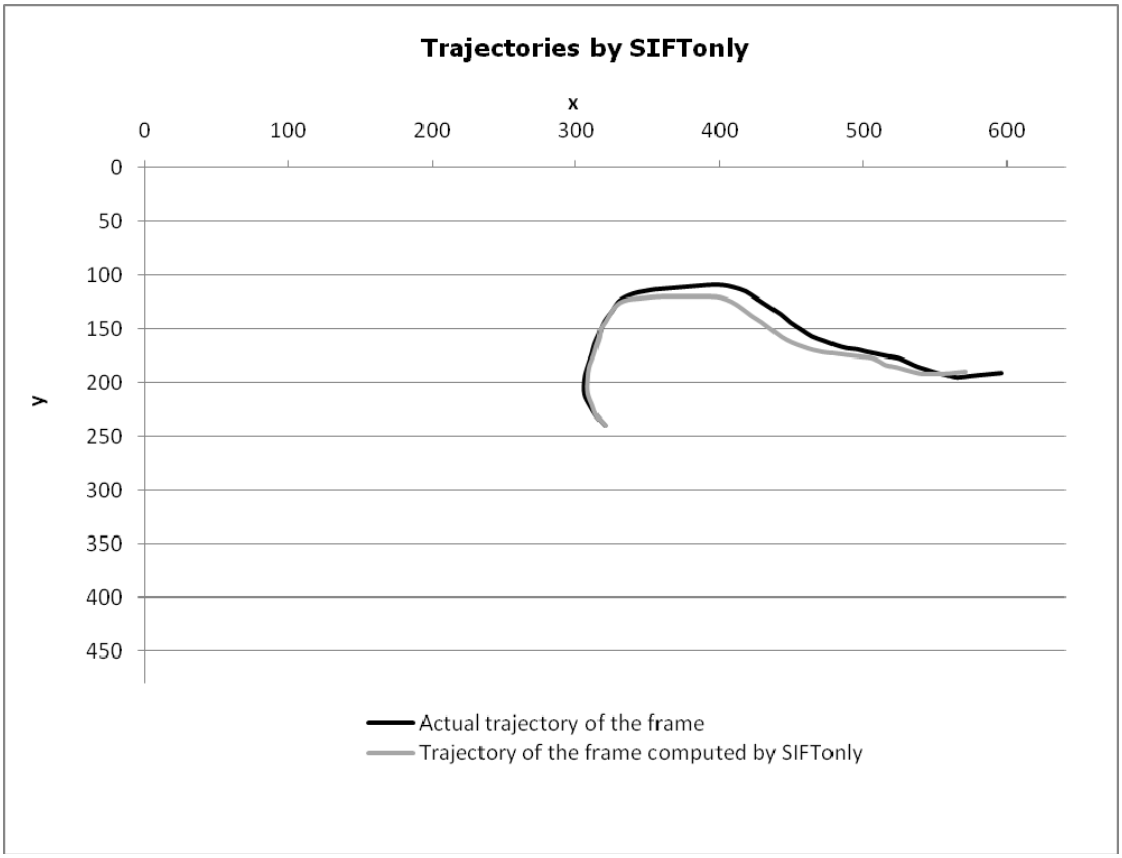


Figure 3.18 – The actual and computed positions of the frame for *SIFTonly*

The max error in position is approximately 15 pixels resulting in an error of 8 centimeters in world coordinates.

3.5 Applying SIFT on Visual Attention regions *(VA+SIFT)*

In section 3.3, we have observed that using Visual Attention algorithm alone as an salient region detector and tracker displays a poor performance. It gives a rough idea about the motion information but would be useless for lots of applications that require higher levels of accuracy. And in section 3.4, we have observed that SIFT algorithm is very good at providing accurate results but computationally very expensive when applied to large image regions. As a consequence, it would be a smart idea to use SIFT features on the visually attractive region.

The idea is to apply the SIFT algorithm to the regions around the visually salient locations of the image. This method will be called *VA+SIFT*. The flow of the execution can be seen in Figure 3.19.

The flow of the algorithm is very similar to *VAonly*. The detected salient regions are added to the trajectories and the non-extended trajectories are terminated as in the case of *VAonly*. In this method, after this step, the salient regions are segmented from the input image for further processing. When segmenting the salient part, a 100x100 region is cropped with the found region center. This cropped region is fed to the SIFT algorithm and so as to extract the SIFT feature points. Formerly we were using the center points of the salient regions to calculate the inter-frame displacement vectors of the trajectories. This time, we use the SIFT points of the regions. The SIFT points of the head of the trajectory are matched with the SIFT points of the second element of the trajectory. The matching procedure is explained in detail in section 3.4. SIFT algorithm typically provides quite a lot of points. As an outcome of this, the matched SIFT points are also several. Since every single SIFT point match may suggest a different displacement value for the frames, this information should somehow be collected together. A trivial procedure is taking the average of these values.

The experimental results of *VA+SIFT* can be found in section 3.5.1.

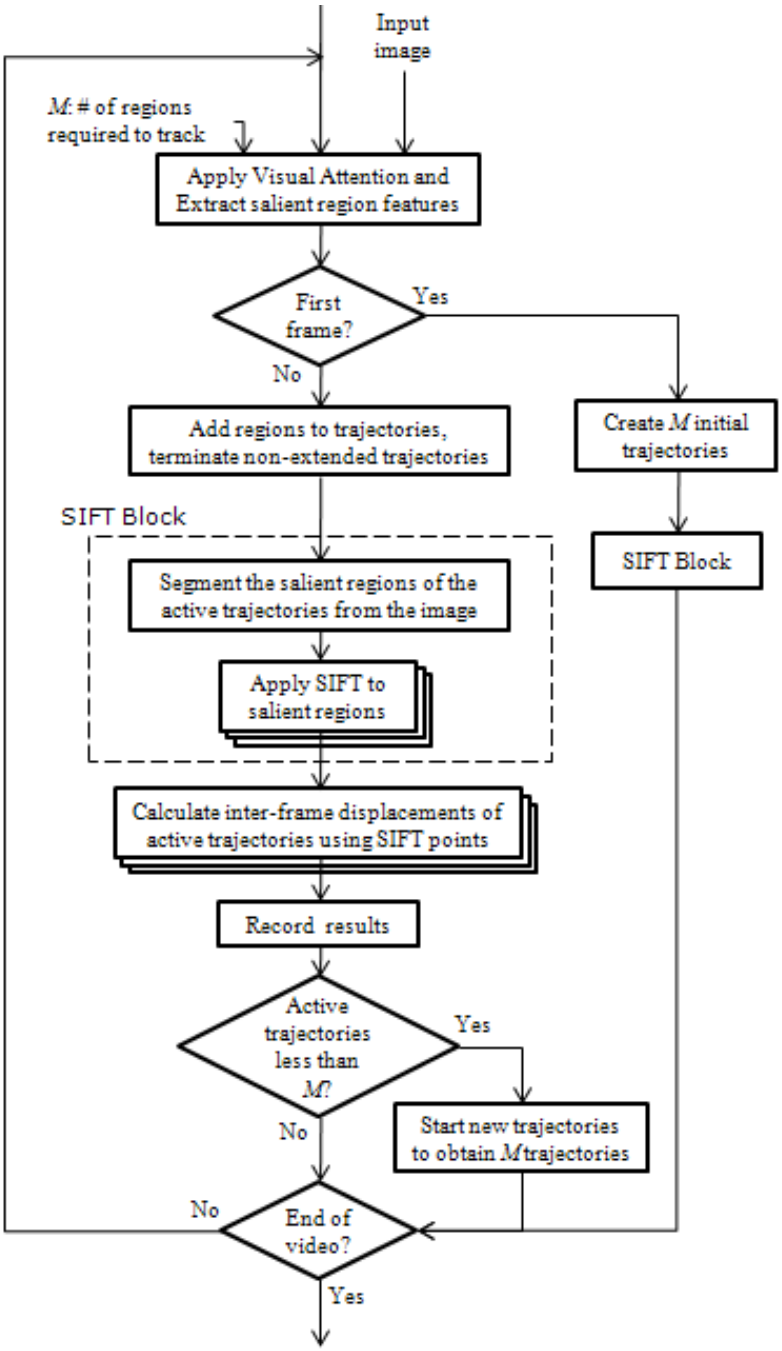


Figure 3.19 – Flowchart of *VA+SIFT* algorithm

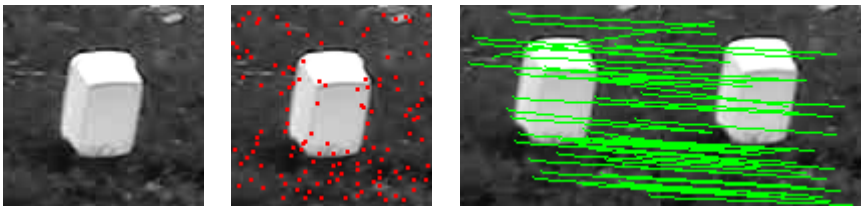
In *VA+SIFT,* unlike *VAonly*, the salient regions are cropped and SIFT algorithm is applied on them to extract more stable features so as to obtain more accurate results. The SIFT points of salient regions of consecutive frames are matched and the inter-frame displacement vectors are calculated using the matched pairs.

3.5.1 Experimental Results

Similar to sections 3.3.1 and 3.4.1, the test videos are *Shaky* and *Planar*.

In the test with *Shaky*, the number of salient regions to track is entered as 1. In Figure 3.20, (a) is an input frame from *Shaky* with the most salient region marked, (b) is the segmented region for SIFT processing, (c) displays the detected SIFT points on the region and (d) shows the SIFT point matches with the next frame. It can be seen that there exist a small number of wrong matches.

(a)



(b) (c) (d)

Figure 3.20 – Segmenting the salient region from a frame of *Shaky*

(a) The input frame. The found salient region is roughly marked.

(b) The segmented salient region

(c) The 141 SIFT points extracted from the cropped region.

(d) The matched SIFT point pairs.

The actual inter-frame displacement vectors and the ones calculated by *VA+SIFT* are plotted in Figure 3.21 for the video *Shaky*. The error in these figures is plotted in Figure 3.22. Consistently, pixel units are employed in all plots.

Remembering Figure 3.9 in section 3.3.1, the inter-frame displacement vectors computed by *VAonly* algorithm were only roughly following the actual pattern with unacceptable levels of error. Currently in Figure 3.21, it can be seen that the pattern computed by *VA+SIFT* follows the ground-truth data much better. Similarly, comparing Figure 3.22 with Figure 3.10; it is apparent that the errors decreased remarkably.

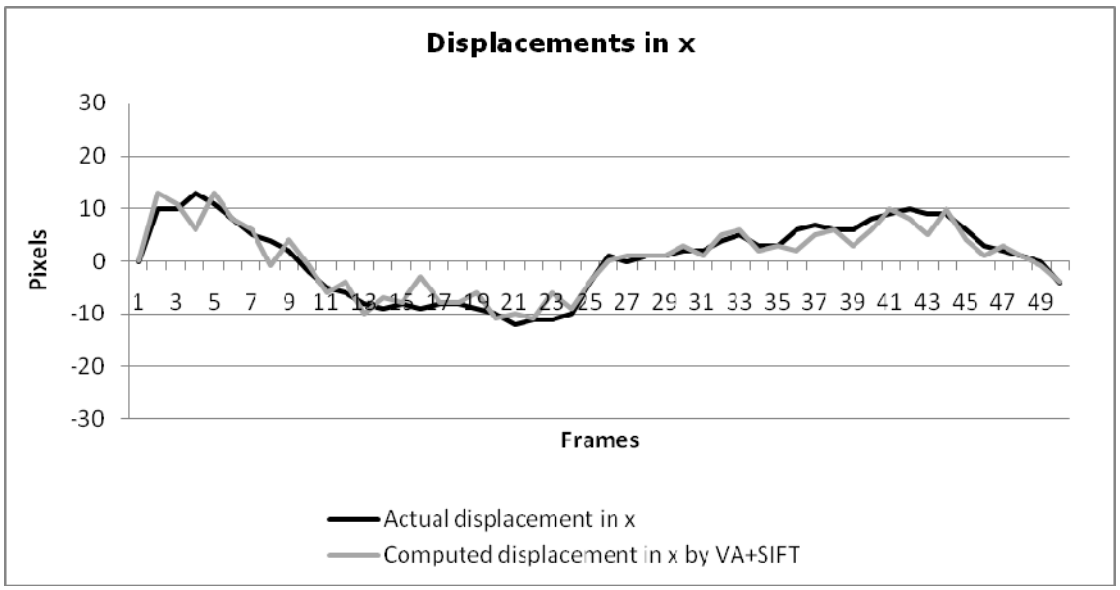
The mean and variance of the total error and the processing time per frame are computed by equations (3.4), (3.5) and (3.7) respectively.

$\mu_E = 3.10$ pixels

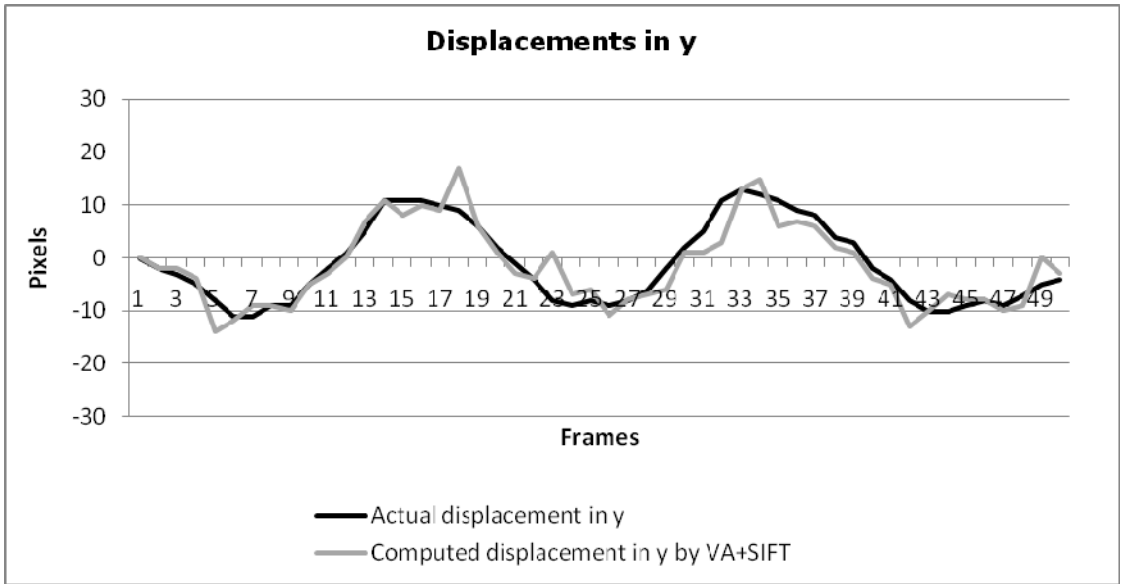
$\sigma_E = 2.17$ pixels

$t_{avg} = 3.93$ s

Using these data, VA+SIFT algorithm can be placed in the performance versus computational cost space as in Figure 3.23.



(a)

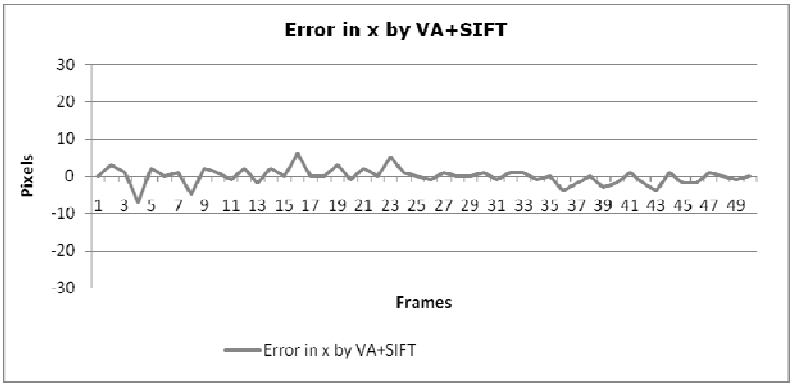


(b)

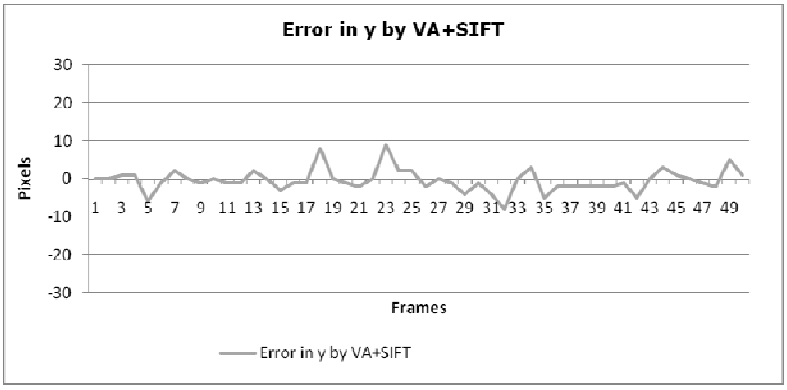
Figure 3.21 – Inter-frame displacement vectors computed by *VA+SIFT* vs. ground truth data for *Shaky*

(a) x-component of the motion (inter-frame displacements in x direction)

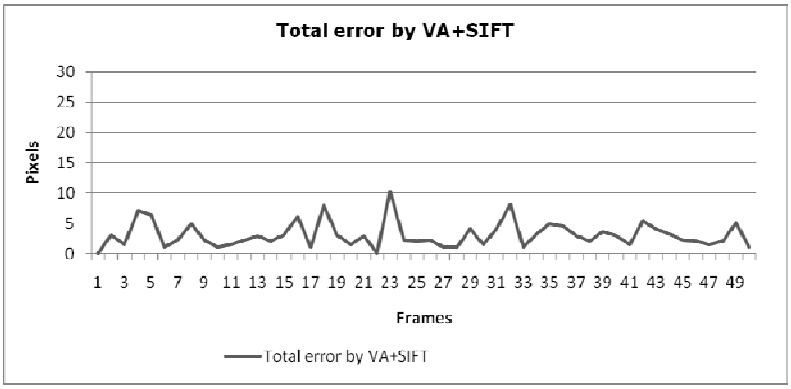
(b) y-component of the motion (inter-frame displacements in y direction)



(a)



(b)



(c)

Figure 3.22 – The error between the actual inter-frame displacement vectors and the ones calculated by *VA+SIFT* for *Shaky*

(a) Error for the x-component of the motion

(b) Error for the y-component of the motion

(c) Total error that equals the vector sum of (a) and (b)

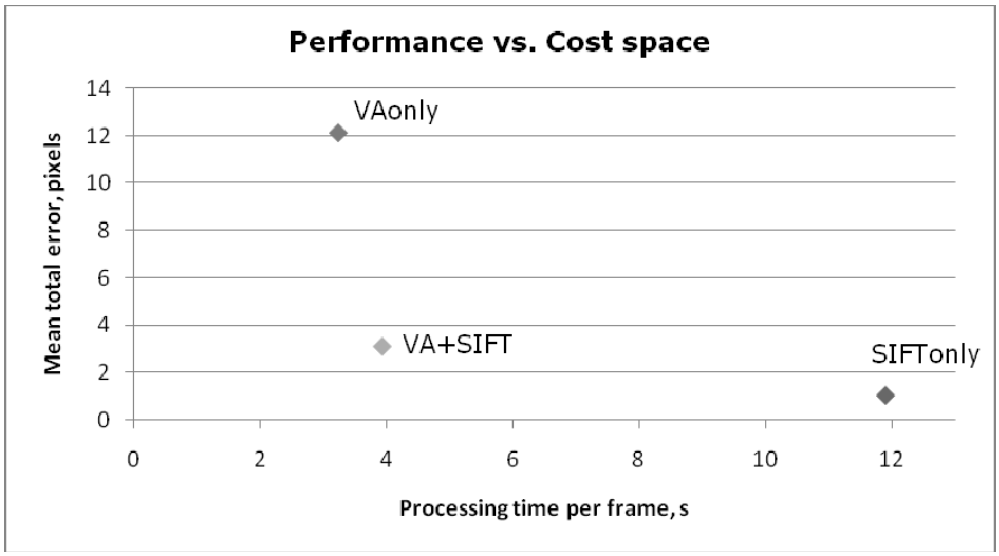


Figure 3.23 – The position of *VA+SIFT* algorithm in "performance-computational cost" space
x-axis is the average time spent for processing a frame and y-axis is the mean of the total error between the displacement vectors computed by *VA+SIFT* and the actual ones.

Figure 3.23 makes it possible to compare *VAonly*, *SIFTonly* and *VA+SIFT* in terms of the two major criteria: performance and computational cost. The plot illustrates that using the SIFT features for the salient region improves the performance significantly compared to *VAonly* while causing only a slight increase in the computational cost. *VA+SIFT* is also much more feasible with respect to *SIFTonly* seeing that the error difference between them is very small whereas the cost difference is massive.

Figure 3.24 displays the actual versus computed positions of the two tracked regions in video *Planar* in pixel coordinates. The actual and computed patterns are very much alike despite the accumulating error.

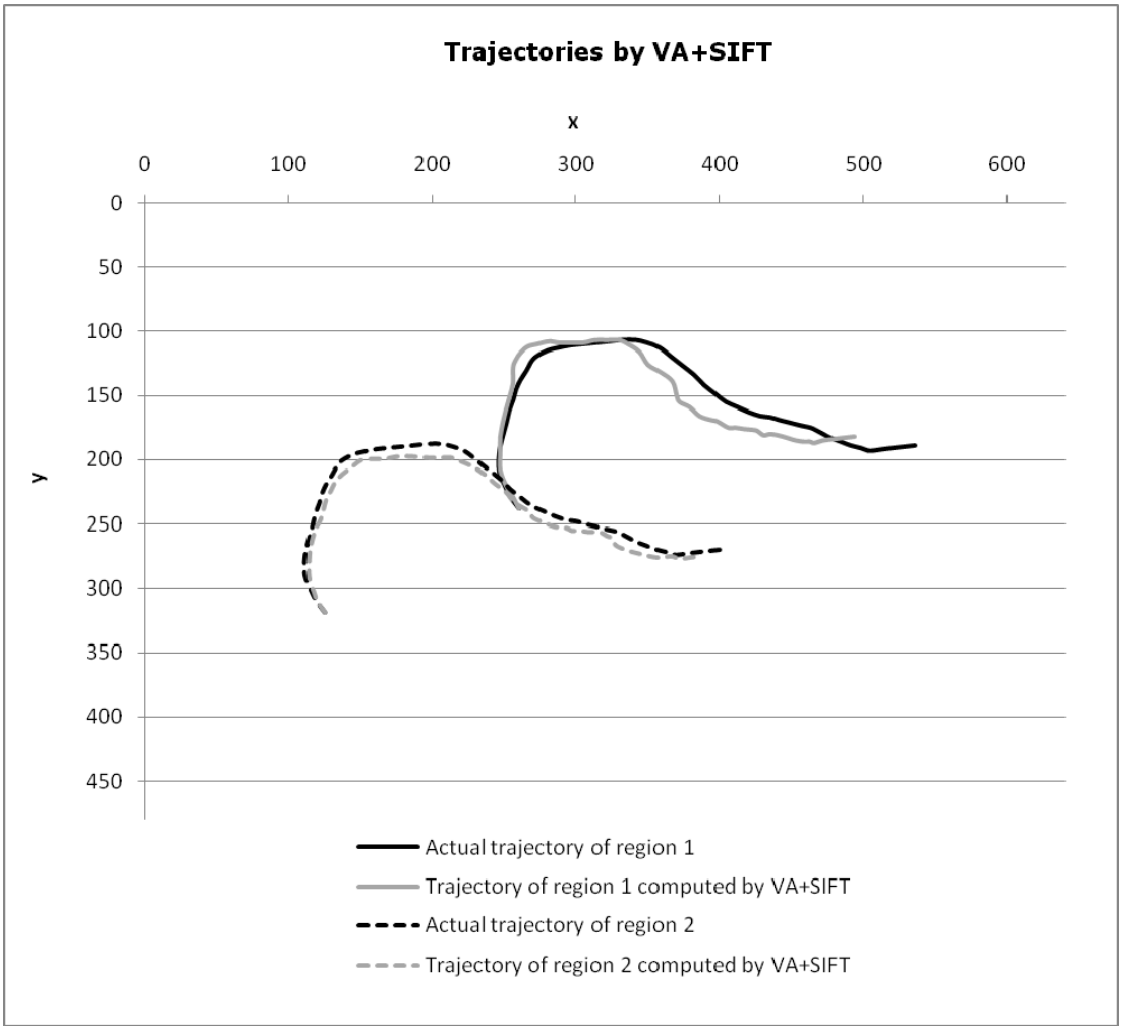


Figure 3.24 – The actual and computed positions of the tracked regions for *VA+SIFT*

The max error in position is approximately 40 pixels resulting in an error of 20 centimeters in world coordinates.

The experimental results of *VA+SIFT* displayed that using the distinctive SIFT features around the visually salient regions to compute precise displacement vectors provide a good tracking performance. However, the computational cost is still too high to be practical. Before concentrating on lowering the computational time, first we will search if we can further enhance the performance.

3.6 Using RANSAC to eliminate the outliers *(VASR)*

Please note that in section 3.5, the SIFT points of the salient region of current frame were matched with the SIFT points of the salient region of previous frame, yielding matched pairs in the order of tens. Each of these pairs may obviously indicate a different displacement vector. In section 3.5, these values were averaged to get a single displacement vector per salient region. A more intelligent method could be employed taking in mind that these values generally contain outliers. RANSAC is an outlier elimination method described in section 2.3.

The idea here is to use RANSAC algorithm while calculating the inter-frame displacement vectors from the matched SIFT point pairs. This method will be called *VASR* as an abbreviation of *VA+SIFT+RANSAC*. The flow of the algorithm is same with Figure 3.19 with the exception of employing RANSAC in the "Inter-frame displacement calculation" block.

To eliminate the outliers using RANSAC, first a parametric system model is needed to be defined. Here the mean and the variance of the displacement vectors will be employed as the model parameters, resulting in 4 parameters:

- The mean of the x component of the displacement vectors,
- The variance of the x component of the displacement vectors,
- The mean of the y component of the displacement vectors,
- The variance of the y component of the displacement vectors.

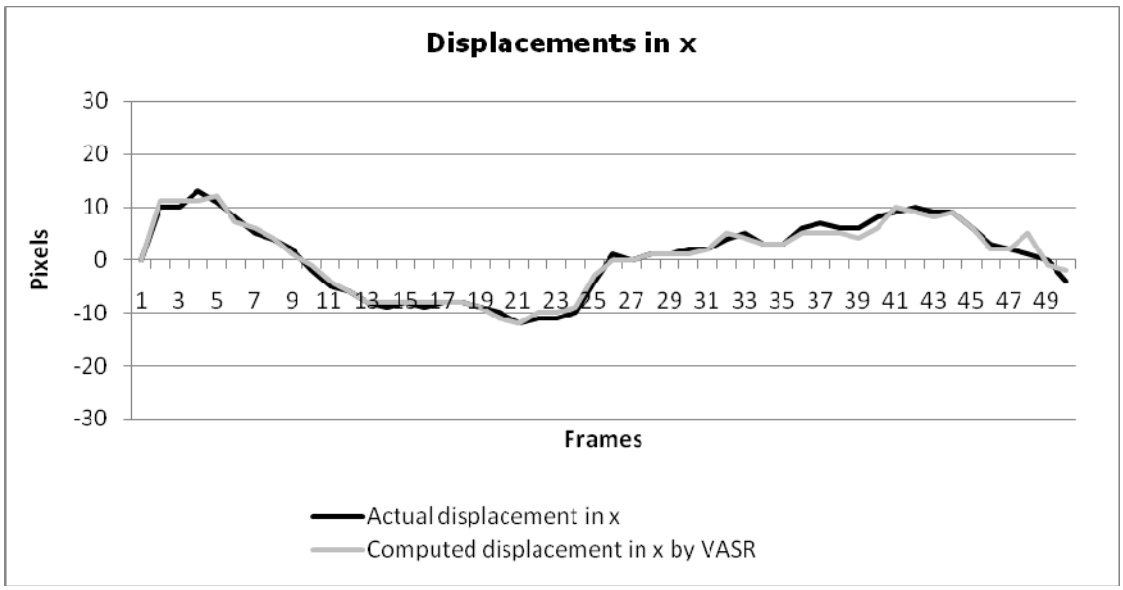
The usage of this model in RANSAC is such that:

- First, a number of random samples are chosen from the displacement vectors, which are calculated using the matched SIFT point pairs.
- A mean and a variance are computed for them.
- Then the remaining samples are tested by these mean and variance values. If the distance between the sample and the computed mean is smaller than the computed variance, the sample is accepted as an inlier; otherwise it is an outlier.
- After testing all samples, if a good number of them are classified as inliers, the estimated model may be a successful one. If so, the mean and variance are re-calculated taking all inliers into account.
- Finally, an error value is computed for the inliers by averaging their distances with the mean value.
- The model with the lowest error is accepted when the predetermined number of iterations are completed.

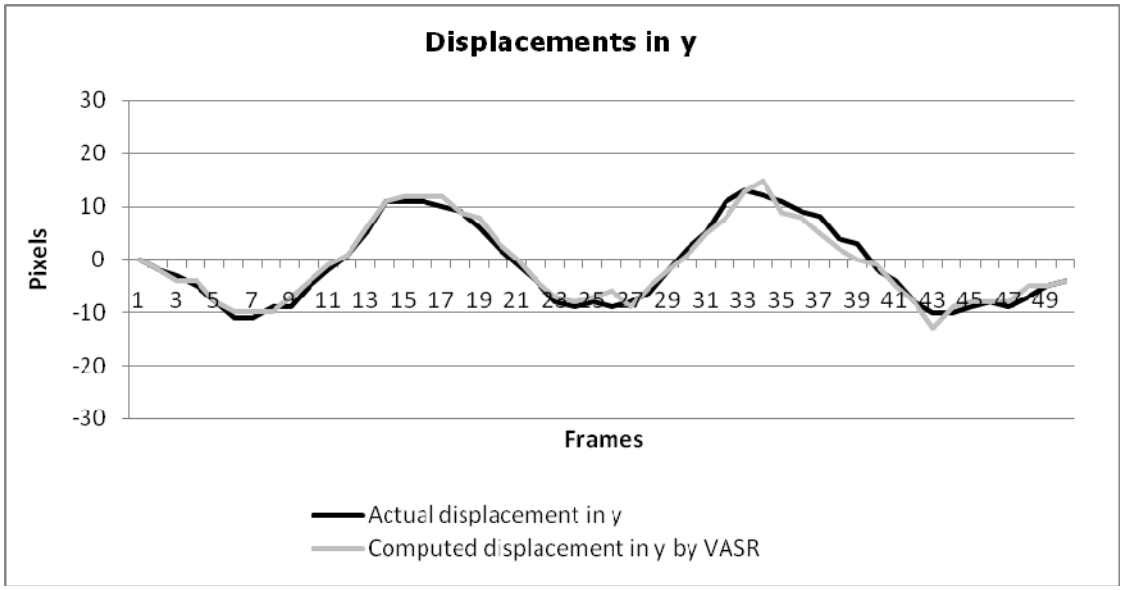
The experimental results of *VASR* can be found in section 3.6.1.

3.6.1 Experimental Results

Figure 3.25 is the plot of the inter-frame displacement vectors that are computed by *VASR*. It can be seen that now the computed pattern almost exactly follows the ground-truth data. In the same way, the errors between the actual and computed displacement vectors, plotted in Figure 3.26, are extremely low.



(a)

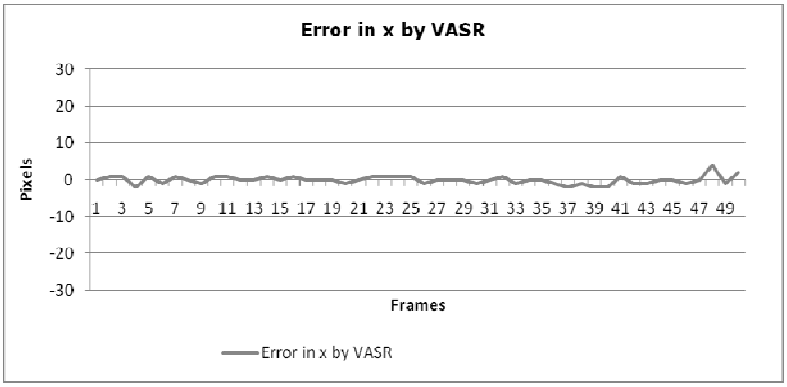


(b)

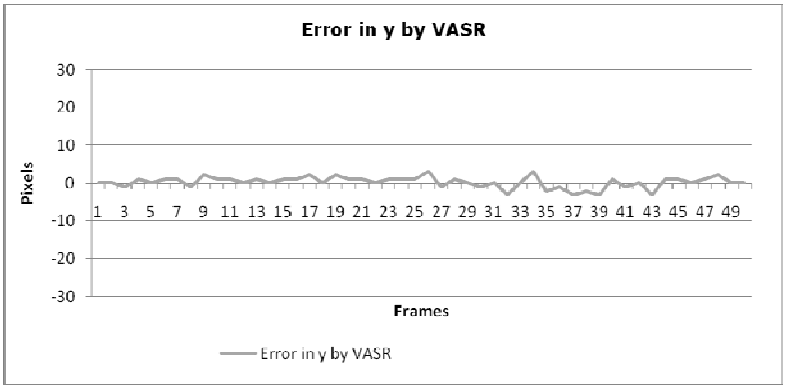
Figure 3.25 – Inter-frame displacement vectors computed by *VASR* vs. ground truth data for *Shaky*

(a) x-component of the motion (inter-frame displacements in x direction)

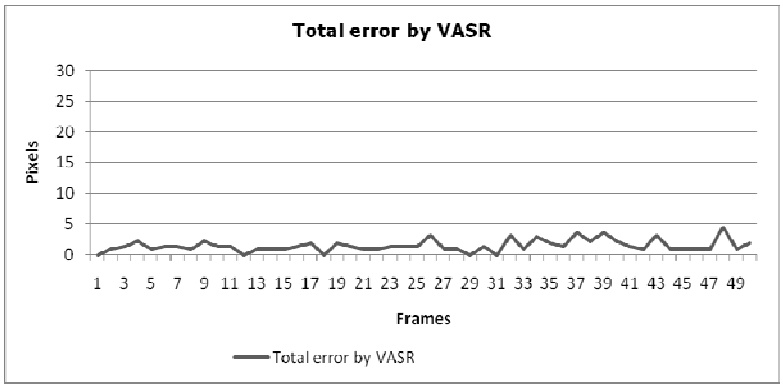
(b) y-component of the motion (inter-frame displacements in y direction)



(a)



(b)



(c)

Figure 3.26 – The error between the actual inter-frame displacement vectors and the ones calculated by *VASR* for *Shaky*

(a) Error for the x-component of the motion

(b) Error for the y-component of the motion

(c) Total error that equals the vector sum of (a) and (b)

Using equations (3.4), (3.5) and (3.7), the mean and variance of the total error and the processing time per frame are computed.

$\mu_E = 1.56$ pixels

$\sigma_E = 0.96$ pixels

$t_{avg} = 4.16$ s

Using these statistics, *VASR* is placed in performance-cost space in Figure 3.27 (*SIFTonly* is not displayed in the plots henceforth as it is on the far right side).

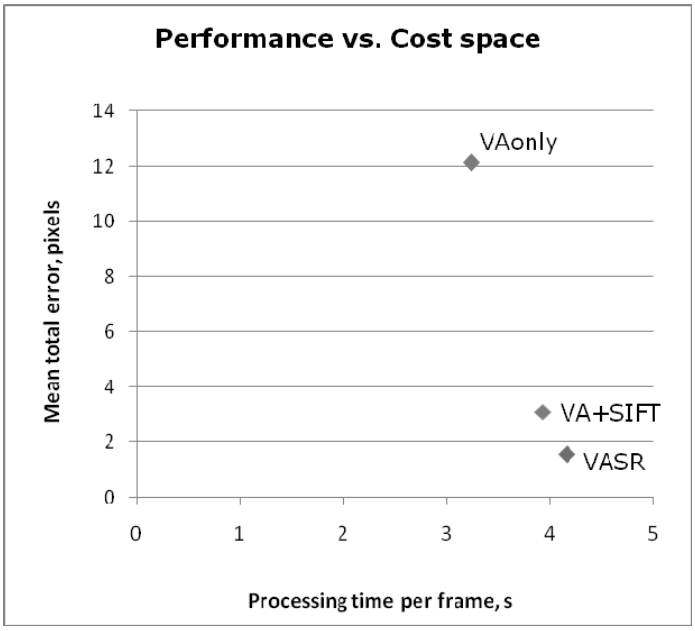


Figure 3.27 – The position of *VASR* algorithm in "performance-computational cost" space
x-axis is the average time spent for processing a frame and y-axis is the mean of the total error between the displacement vectors computed by *VASR* and the actual ones.

The position of *VASR* in Figure 3.27 indicates that *VASR* is a more accurate algorithm compared to VA+SIFT at the cost of a minor increase in complexity.

Figure 3.28 is the plot of actual and computed positions of the two tracked regions in video *Planar* in pixel coordinates. It can be seen that the error between the actual and computed pattern has decreased with respect to *VA+SIFT*.

Evaluating the progress from *VAonly* to *VASR*, it can be concluded that at this point, the performance is significantly enhanced by *VASR* algorithm and has reached an acceptable level. Nevertheless, the complexity of the algorithm is even higher. Decreasing the computational cost of *VASR* in some way would be very beneficial on the road to obtaining a practically valuable algorithm. Section 3.7 proposes a way of decreasing the costs.

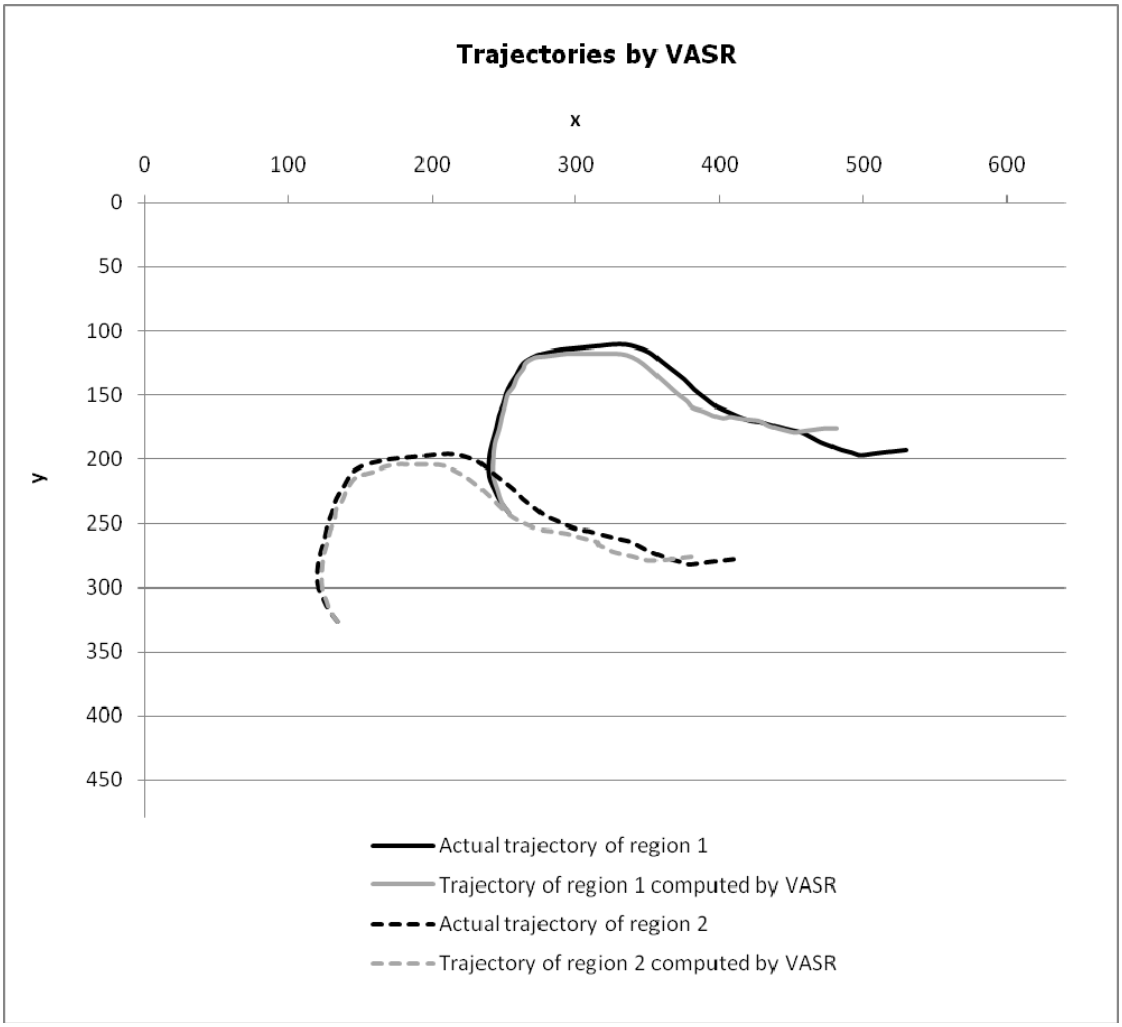


Figure 3.28 – Actual and computed positions of the tracked regions for *VASR*
The actual and computed patterns are very similar though an accumulation in the error can be observed.

3.7 Adding Region-Based Processing *(R-based VASR)*

At this point, having an algorithm that is substantially acceptable in performance (*VASR*), our aim is to reduce the computational cost. If we analyze the flowchart of *VASR* in Figure 3.19 and search for the time consuming blocks, the *Visual Attention* block stands out. In fact, the experimental statistics reveal that this step takes about 80 percent of the processing time. The time it takes to process a frame by *Visual Attention* depends on several parameters but the mostly on the frame size. Hence, if we can reduce the size of the input images fed to the *Visual Attention* block, the time spent per frame is expected to decrease significantly.

Since the input frames are the elements of a video sequence, there is continuity in between. In other words, if the position of the m^{th} salient region is $(x_{m,n},y_{m,n})$ in the n^{th} frame and $(x_{m,n+1},y_{m,n+1})$ in the $(n+1)^{th}$ frame; then $(x_{m,n+1},y_{m,n+1})$ must be in the vicinity of $(x_{m,n},y_{m,n})$. Thus, if the position of a salient region is known in n^{th} frame, it is unnecessary to search everywhere in the $(n+1)^{th}$ frame for that region. It is just sufficient to process the region around its earlier position.

Adapting this idea to *VASR*, the idea is, once the salient region locations are computed in the first frame, only the neighboring regions can be searched for those salient regions in the following frames. This method will be called *R-based VASR* as an abbreviation of *Region-based VASR*. The flow of the algorithm is in Figure 3.29. There exists a decision block at the first step to decide whether the current frame will fully be processed or only its specific regions will be processed. The internal structure of this decision block can be found in Figure 3.30.

A frame is fully processed if any of the three cases below holds:

- If it is the first frame of the input video sequence,
- If the number of active trajectories (regions being tracked) is less than the required number, (Notice that the number of active-trajectories is not completed to the required number as in the case of full frame processing.)
- If the time for periodic full frame processing has reached.

The periodic full frame processing scheme aims to avoid diverging as the number of processed frames increase.

Depending on the outcome of the decision block, a frame can either be fully or regionally processed. If full processing is required, the flow of the algorithm is same with *VASR*. If the current frame will be processed in a region-based manner, the algorithm starts with determining the regions to be processed. Remember that the salient region positions of the current frame are expected to be in the neighborhood of the salient region positions of the previous frame. Thus the salient region centers of the previous frame are used to crop the processing regions of the current frame. Let the center of m^{th} salient region in n^{th} frame be $(x_{m,n},y_{m,n})$. The search region of $(n+1)^{th}$ frame is computed solely by taking $(x_{m,n},y_{m,n})$ as the center and cropping the region around it with constant size in both directions (100 pixels, resulting in a 200x200 square region). A brighter idea would be using the motion information that can be extracted from a number of past frames.

After segmenting the regions, they are fed to the *Visual Attention* block to locate the salient region positions and extract the saliency features. After finding the new salient positions and adding them to their matched trajectories, the regions around them are re-cropped (100x100) in order to apply SIFT. The remaining part of the algorithm is same with the full frame processing scheme except that; if the number of active trajectories remain less than the required number (M), they are not completed but the forthcoming frame is fully processed to obtain M salient regions.

A problem that is worth mentioning is that assume n^{th} frame is fully processed. As a result, its saliency feature vectors (contribution of each of intensity, color and orientation channels to the saliency value) are computed using the full image. When the next frame, which is $(n+1)^{th}$, is regionally processed, this time the feature vectors are computed using the region image. Since Visual Attention algorithm is not solely pixel-orientated but involves global normalization operations, these two feature vectors appear to be quite different and do not

satisfy equation (3.10). Therefore the feature vectors cannot be matched and the found region cannot be added to its trajectory. This problem is solved by processing the n^{th} frame once more in a region-based manner. The 200x200 regions around the salient positions are fed to the *Visual Attention* algorithm once more to compute the saliency feature vectors.

Section 3.7.1 presents the experimental results of *R-based VASR*.

3.7.1 Experimental Results

In the test with the video *Shaky*, the number of salient regions to track is entered as 1 as usual. In Figure 3.31, (a) is the 200x200 search region in an input frame of *Shaky*. The search region is derived from the salient region location in the previous frame. Figure 3.31 (b) is the found salient region. It can be seen that the search region is indeed quite large. This is because of the fact that, when estimating the search region, no information is gathered about the dynamics of the motion, but only the salient region of the previous frame is taken and expanded in all directions.

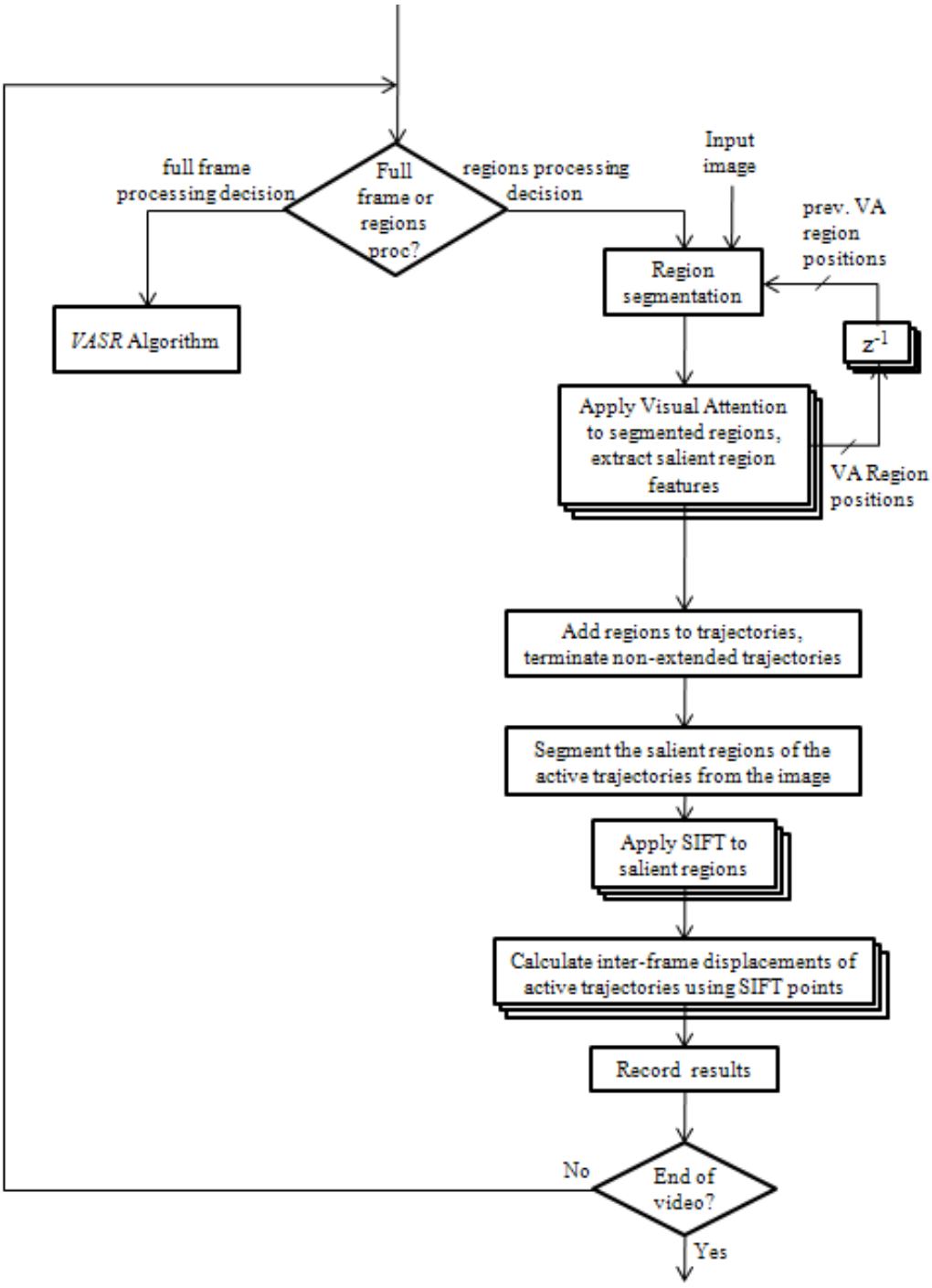


Figure 3.29 – Flowchart of *R-based VASR* algorithm

Depending on the outcome of the initial decision block, a frame is either fully or regionally processed.

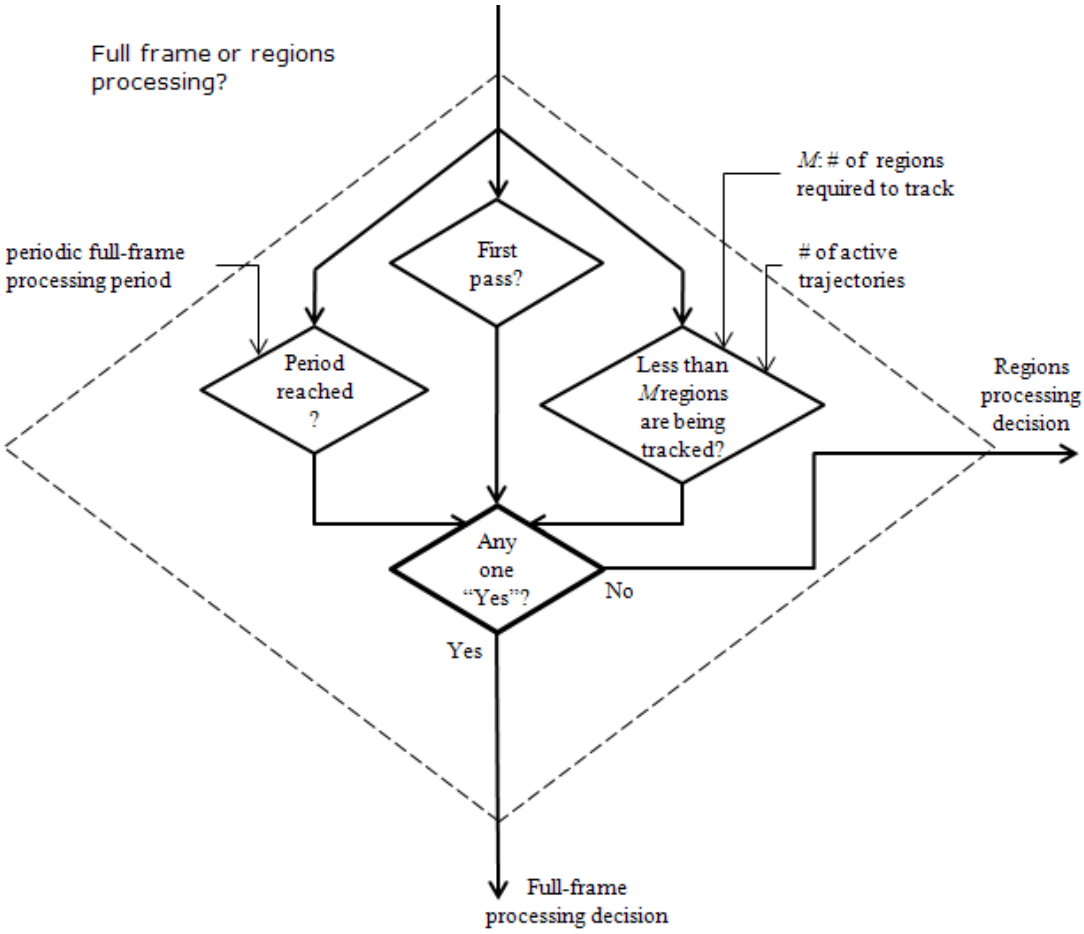


Figure 3.30 – Flowchart of the decision block of *R-based VASR* algorithm

If current frame is the first frame of the video sequence, or if less than required number of regions are being tracked, current frame will be fully processed. Moreover, a periodical full frame processing scheme exists as well to avoid diversion.

(a) (b)

Figure 3.31 – The search region and the found salient region in a frame of *Shaky*

(a) The search region derived from the previous position of the salient region

(b) The salient region found in the search region

The inter-frame displacement vectors computed by *R-based VASR* are displayed in Figure 3.32 for the video *Shaky*. Figure 3.33 displays the errors between these actual and computed displacement vectors. The error values did not change much with respect to the errors of *VASR*.

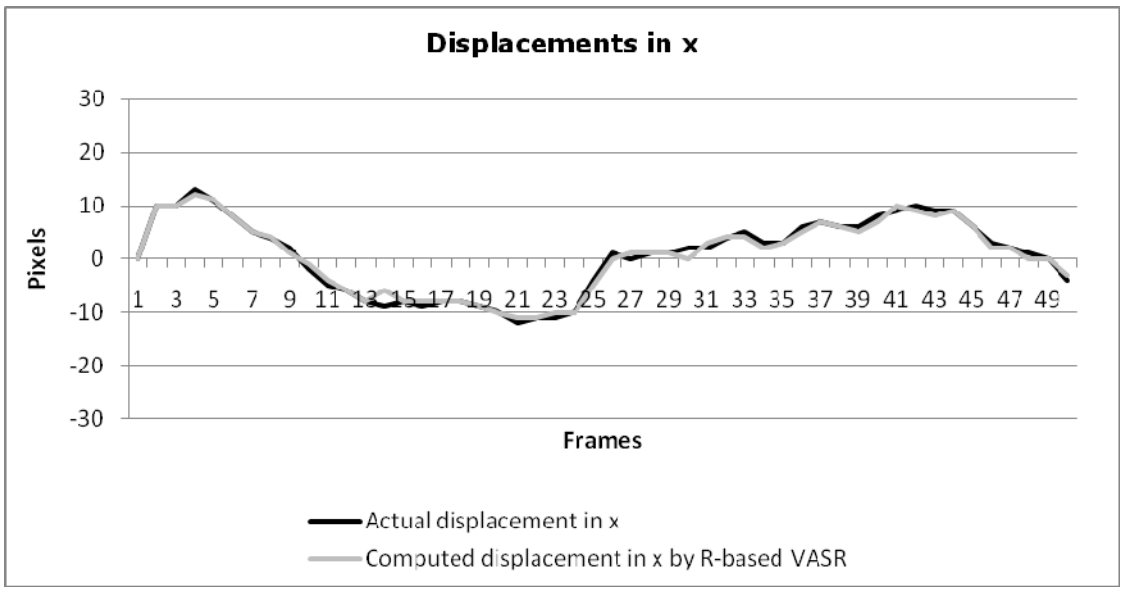
Computing the mean and the variance of the total error, and the average time spent per frame using equations (3.4), (3.5) and (3.7) we get:

$\mu_E = 1.45$ pixels

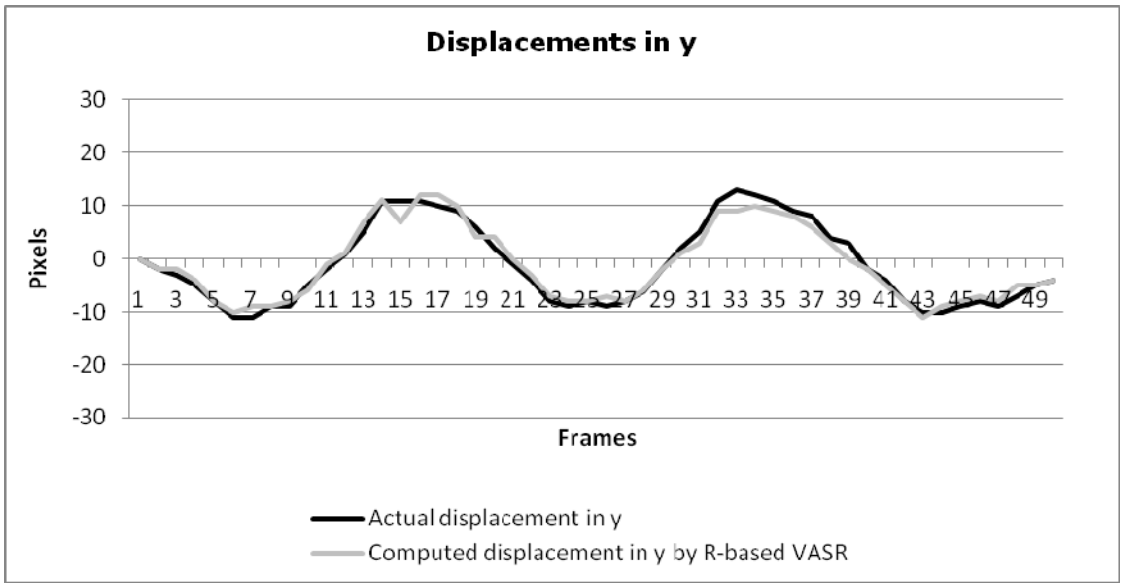
$\sigma_E = 0.94$ pixels

$t_{avg} = 1.43$ s

Using these values to place *R-based VASR* in performance-cost space, Figure 3.34 is obtained. The vital outcome of Figure 3.34 is that region based processing scheme decreased the computational cost dramatically while preserving the algorithm performance. The position *R-based VASR* is the current best position obtained on this graph (closest to origin).



(a)

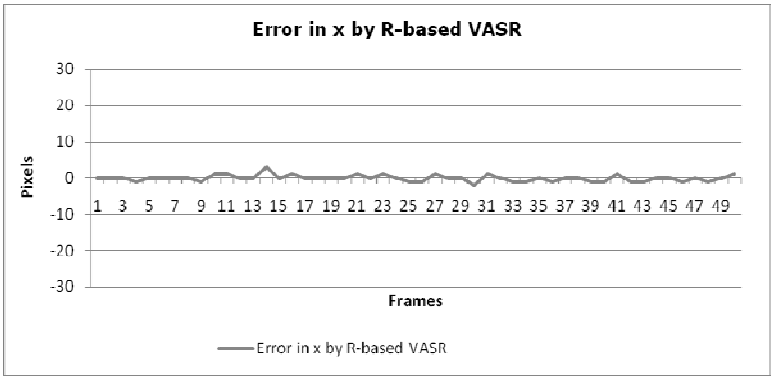


(b)

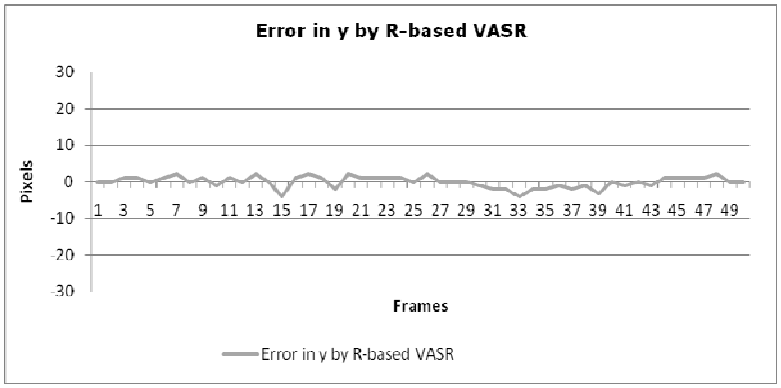
Figure 3.32 – Inter-frame displacement vectors computed by *R-based VASR* vs. ground truth data for *Shaky*

(a) x-component of the motion (inter-frame displacements in x direction)

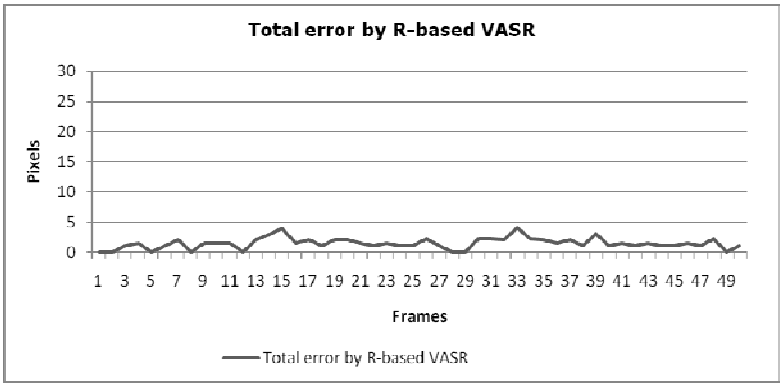
(b) y-component of the motion (inter-frame displacements in y direction)



(a)



(b)



(c)

Figure 3.33 – The error between the actual inter-frame displacement vectors and the ones calculated by *R-based VASR* for *Shaky*

(a) Error for the x-component of the motion

(b) Error for the y-component of the motion

(c) Total error that equals the vector sum of (a) and (b)

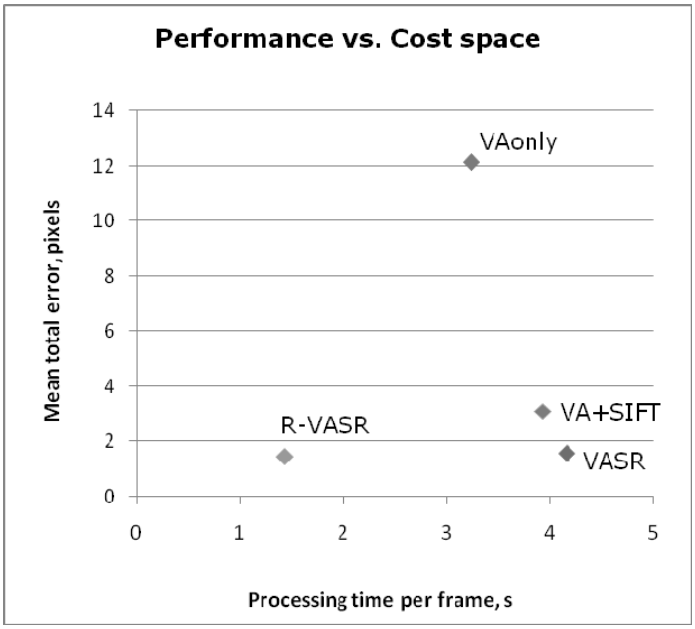


Figure 3.34 – The position of *R-based VASR* algorithm in "performance-computational cost" space

x-axis is the average time spent for processing a frame and y-axis is the mean of the total error between the displacement vectors computed by *R-based VASR* and the actual ones.

Figure 3.35 is the plot of actual and computed positions of the two tracked regions in video *Planar* in pixel coordinates using *R-based VASR*. It can be seen that the error between the actual and computed patterns is small. However, region-based processing has such an effect that the tracked regions are sometimes lost throughout the video. When this happens, the next frame of the video is fully processed to complete the required number of regions being tracked. This causes breaks in the trajectories.

Evaluating *R-based VASR*, the computational cost of the algorithm is less than any other proposed method so far but still not very low. Accordingly, section 3.8 suggests a way of further decreasing the cost.

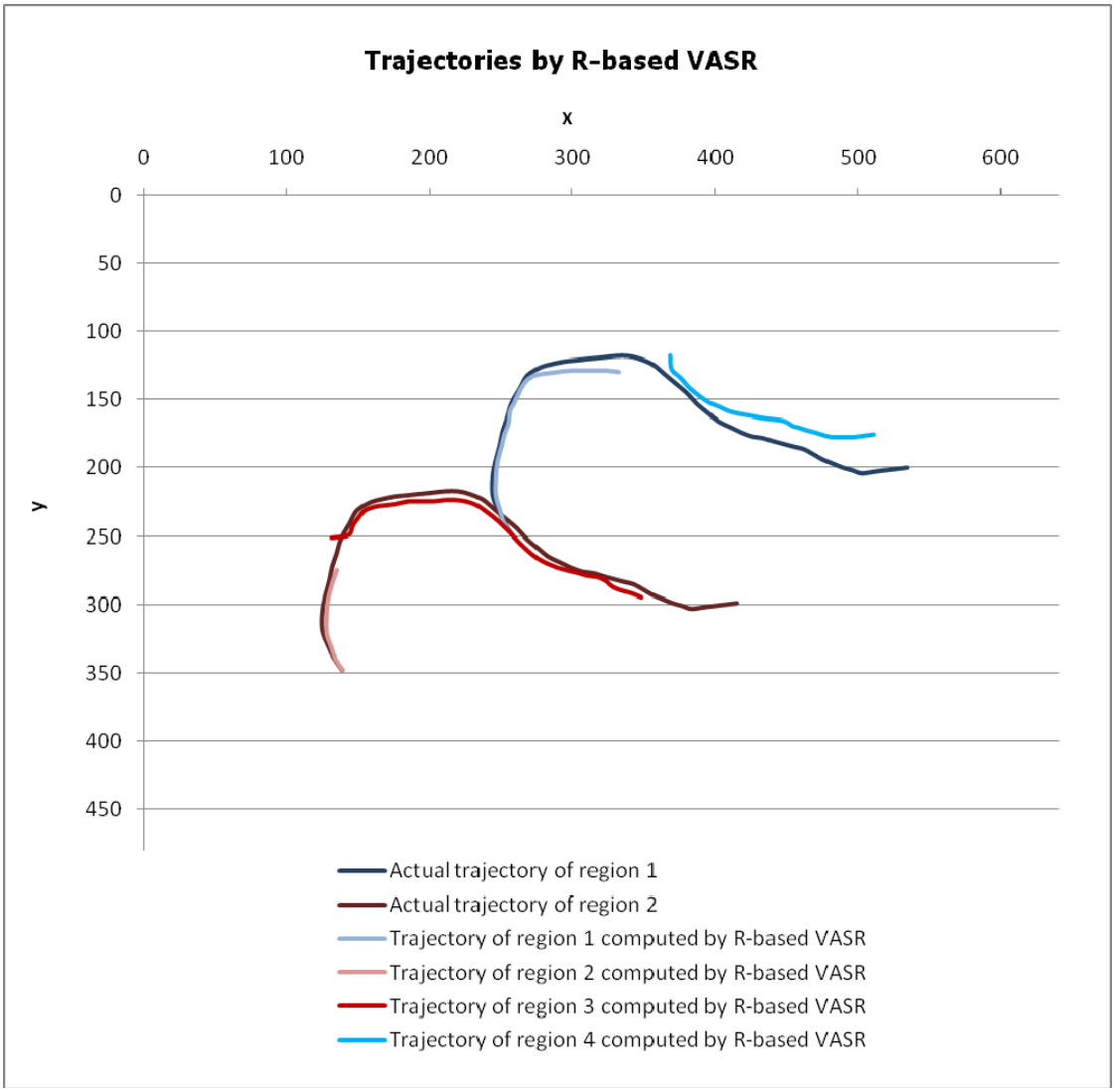


Figure 3.35 – Actual and computed positions of the tracked regions for R-based *VASR*

3.8 Limiting Visual Attention Channels (*RC-based VASR*)

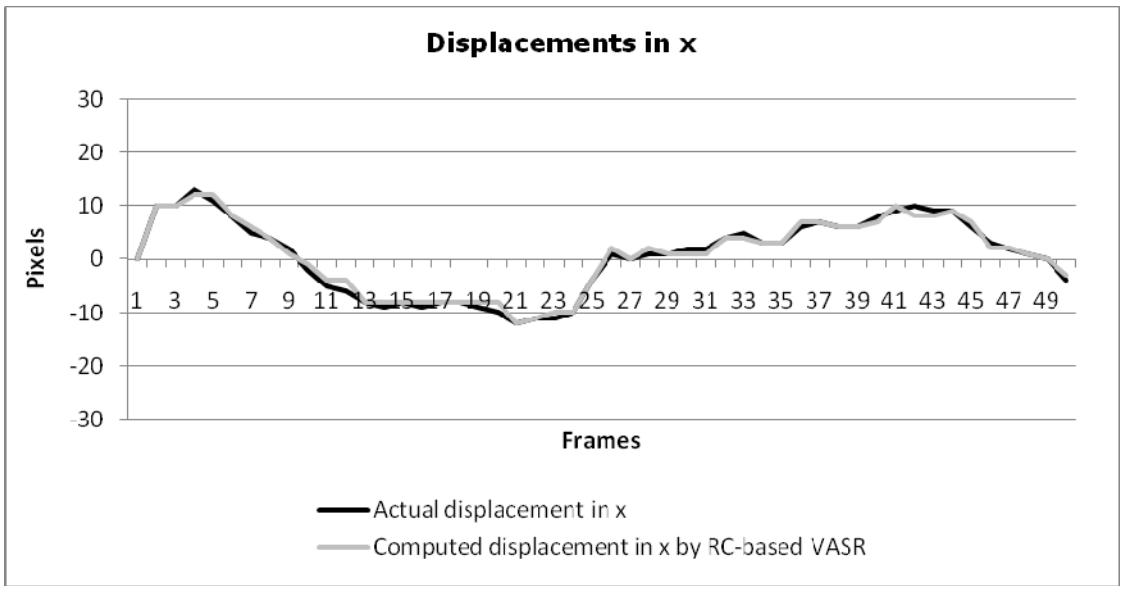
Region-based processing concept has decreased the computational cost significantly but still about 65 percent of the processing time is spent for the *Visual Attention* processing. Remember that, *Visual Attention* algorithm has three channels; namely intensity, color and orientation. When it is invoked, it computes the saliency values of the regions with respect to these three channels. A question that comes in mind is that could fewer channels be sufficient? When salient regions in a frame are detected, in the following frame, the same salient regions are searched to continue tracking. In addition, recall that we have a salient region feature vector keeping information about the contributions of each channel to the region's saliency. Having these in mind, it can be deduced that similar to limiting the search regions, attention channels can also be limited.

The idea is, once a saliency feature vector is computed for a salient region, that region can be searched in the subsequent frame using only its most dominant attention channel. This method will be called "*RC-based VASR"* as an abbreviation of "*Region and Channel based VA+SIFT+RANSAC*". The flowchart of the algorithm is same with R-based VASR except for the *Visual Attention* block (see Figure 3.29). Channel limiting is only performed in the regions processing branch. When full frame is processed, the attention channels are not be limited in order to find new salient regions.

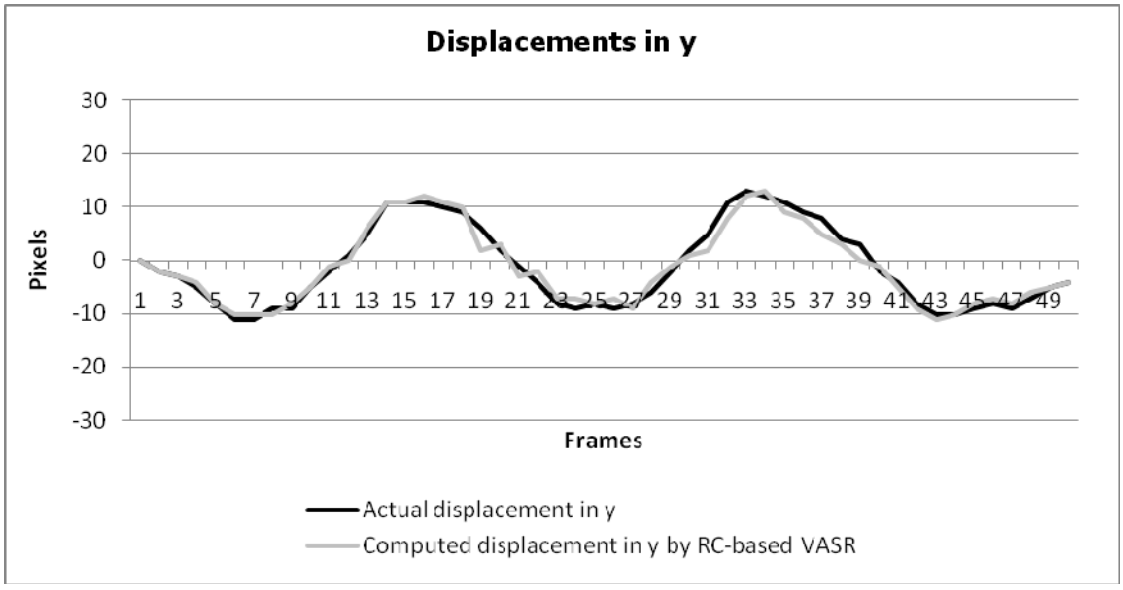
Experimental results of RC-based VASR can be found in section 3.8.1.

3.8.1 Experimental Results

The inter-frame displacement vectors computed by *RC-based VASR* are displayed in Figure 3.36 for the video *Shaky*. Figure 3.37 display the errors between the actual and computed displacement vectors. The mean and the variance of the total error, and the average time spent per frame are:



(a)

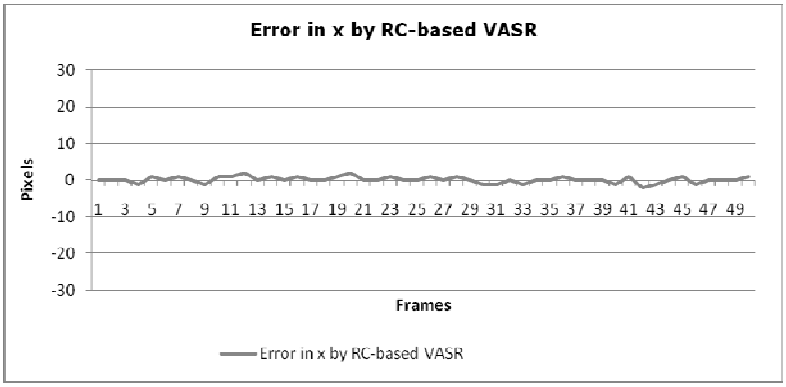


(b)

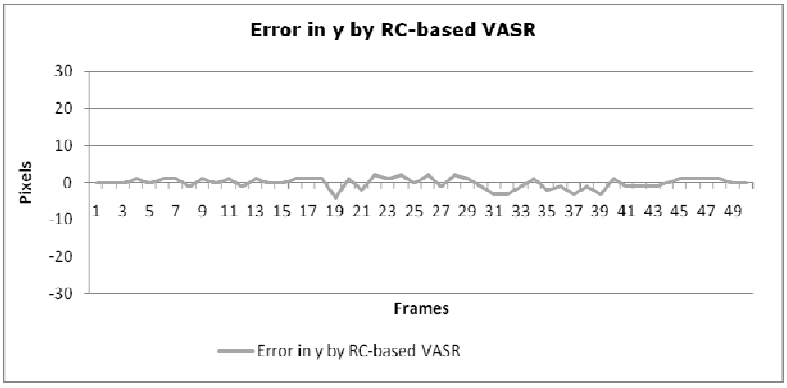
Figure 3.36 – Inter-frame displacement vectors computed by *RC-based VASR* vs. ground truth data for *Shaky*

(a) x-component of the motion (inter-frame displacements in x direction)

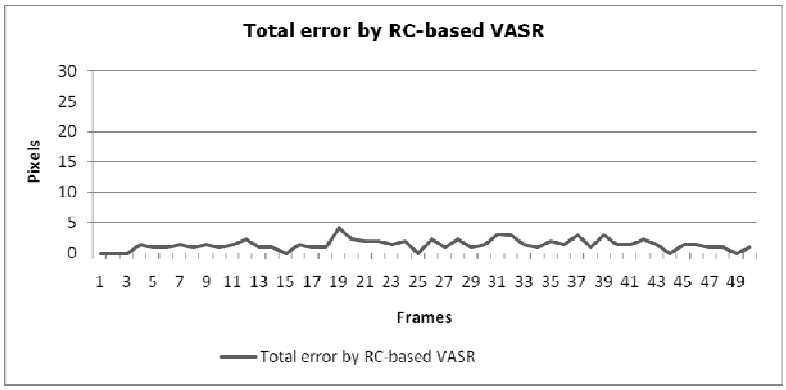
(b) y-component of the motion (inter-frame displacements in y direction)



(a)



(b)



(c)

Figure 3.37 – The error between the actual inter-frame displacement vectors and the ones calculated by *RC-based VASR* for *Shaky*

(a) Error for the x-component of the motion

(b) Error for the y-component of the motion

(c) Total error that equals the vector sum of (a) and (b)

$\mu_E = 1.43$ pixels

$\sigma_E = 0.88$ pixels

$t_{avg} = 1.43$ s

Placing *RC-based VASR* to the performance-cost space based on these values gives the plot in Figure 3.38.

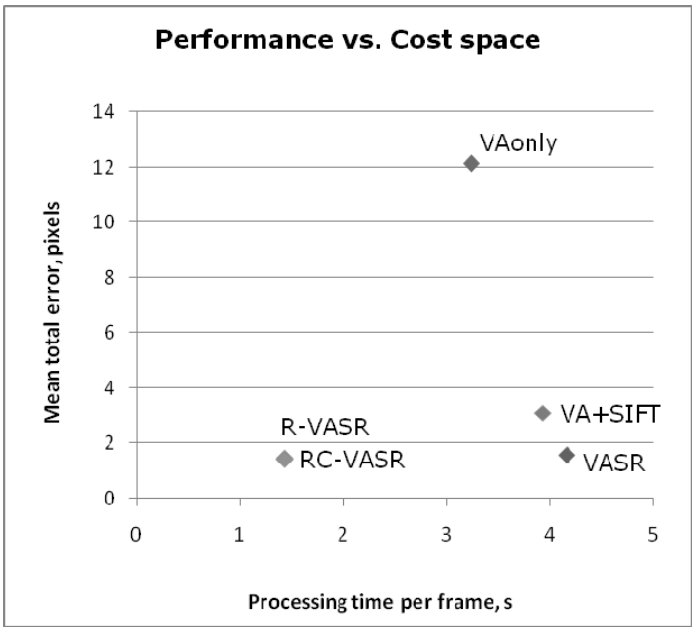


Figure 3.38 – The position of *RC-based VASR* algorithm in "performance-computational cost" space

x-axis is the average time spent for processing a frame and y-axis is the mean of the total error between the displacement vectors computed by *RC-based VASR* and the actual ones.

The positions of *RC-based VASR* and *R-based VASR* in Figure 3.38 are quite on top of each other. However, comparing the numerical statistics reveal that *RC-based VASR* is slightly better than *R-based VASR* in both performance and cost. The reason of the performance gain is that limiting the attention channels increases the possibility of finding the same salient region, thus decreases the possibility of wrong matches. The gain in cost was expected due to the reduced operations in Visual Attention block. Indeed the cost gain is much less than expected.

Figure 3.39 is the plot of actual and computed pixel coordinates of the two tracked regions in video *Planar* using *RC-based VASR*. It can be seen that the error between the actual and computed patterns is again very small. Furthermore, limiting the *Visual Attention* channels removed the breaks on the trajectories. Recall that the trajectories in Figure 3.35 were split since region-based processing caused losing the tracked regions a few times. Limiting the attention channels increased the probability of finding the same salient region thus making longer trajectories possible.

Evaluating *RC-based VASR*, the computational cost and the performance of the algorithm are quite satisfactory.

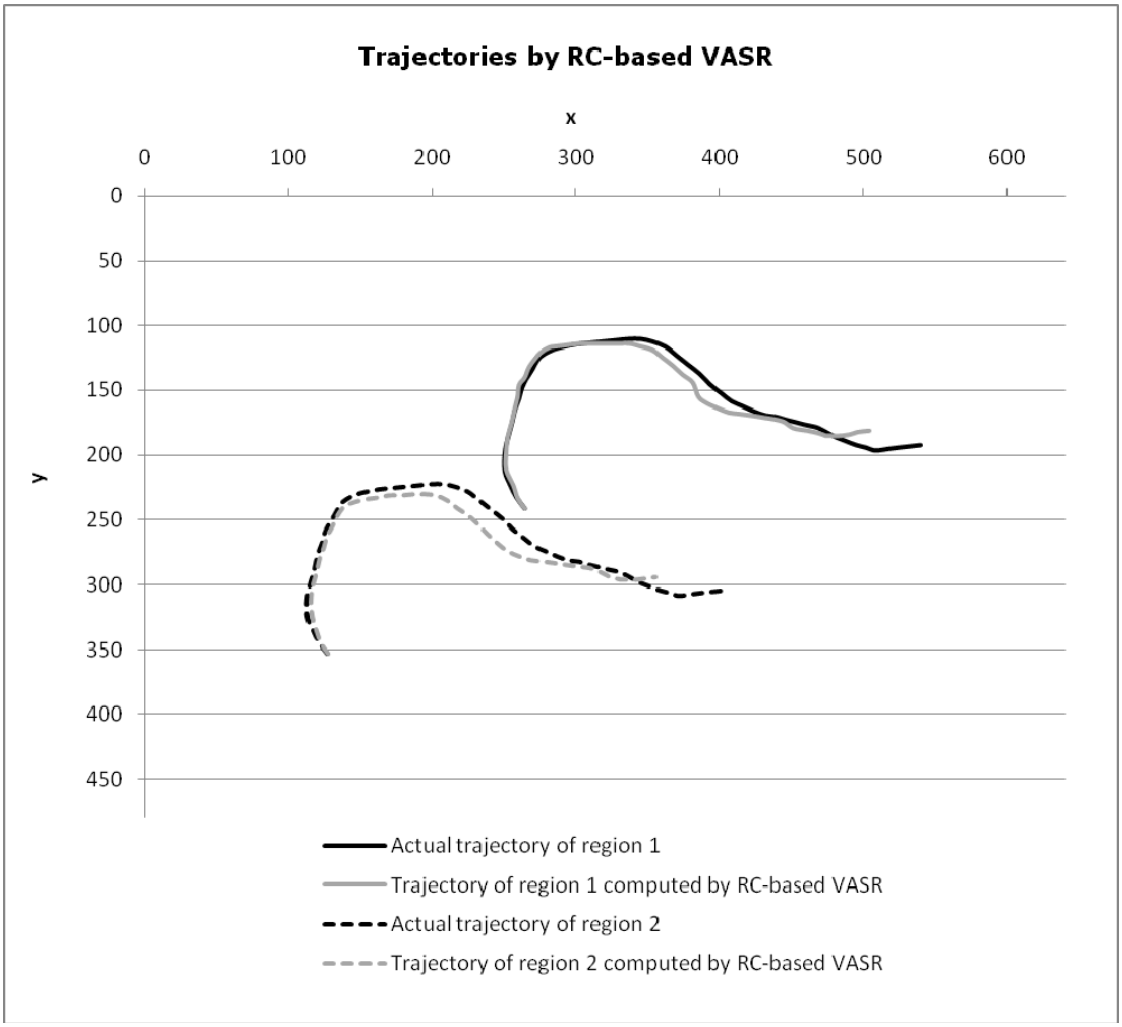


Figure 3.39 – Actual and computed positions of the tracked regions for RC-based *VASR*

3.9 Narrowing the search regions (*MoReC*)

In section 3.7, the idea of region based processing is introduced to decrease the computational cost. When a salient region position is determined in a frame, only the neighboring region is searched in the next frame. In section 3.7, the search region is defined as the 200 pixel x 200 pixel area around the previous position of the salient region. In this section, the search region is tried to be narrowed by using the motion information in order to further decrease the computational load.

The idea is to extract the motion information of the tracked salient region by using the past frames and then to estimate the region's next position, thus to obtain a much narrower search area. This method will be called *MoReC* as an abbreviation of *"Motion-Region-Channel-based VASR"*.

The motion extraction is performed in its simplest way such that, the displacement is calculated between the two previous frames. Let the position of m^{th} salient region is $(x_{m,n},y_{m,n})$ in the n^{th} frame and $(x_{m,n+1},y_{m,n+1})$ in the $(n+1)^{th}$ frame. When trying to compute a search region for the $(n+2)^{th}$ frame, first we should estimate the motion information using the n^{th} and $(n+1)^{th}$ frames. The displacement between n^{th} and $(n+1)^{th}$ frames, call $\Delta\mathbf{x}_{n+1}$, can be computed by:

$$\Delta\mathbf{x}_{m,n+1}(\Delta x_{m,n+1},\Delta y_{m,n+1}) = \mathbf{x}_{m,n+1}(x_{m,n+1},y_{m,n+1}) - \mathbf{x}_{m,n}(x_{m,n,},y_{m,n}) \qquad (3.12)$$

It can be assumed that the time intervals between frames are equal since they are part of a video sequence. Accordingly, with the assumption that the camera has *uniform translational motion*, i.e., moving on a straight line with constant speed, it can be supposed that the displacement between the $(n+1)^{th}$ and $(n+2)^{th}$ frame will be equal to $\Delta\mathbf{x}_{m,n+1}$. Thus, the estimated position of the salient region in $(n+2)^{th}$ frame, call $\mathbf{x'}_{m,n+2}$, can be written as below:

$$\mathbf{x}'_{m,n+2}(x_{m,n+2},y_{m,n+2}) = \mathbf{x}_{m,n+1}(x_{m,n+1},y_{m,n+1}) + \Delta\mathbf{x}_{m,n+1}(x_{m,n+1,},y_{m,n+1}) \qquad (3.13)$$

In this implementation, the search region of m^{th} salient region in $(n+2)^{th}$ frame is defined as the 100 pixel x 100 pixel square centered at $\mathbf{x'}_{m,n+2}$. Notice that this is one-fourth of the search region defined in section 3.7.

Experimental results of *MoReC* can be found in section 3.9.1.

3.9.1 Experimental Results

Figure 3.40 shows examples of the estimated search regions for the video *Shaky*. The first two images are examples of the worst calculated search regions and the third one is a typical one. It can be observed that even the worst cases are sufficiently successful.

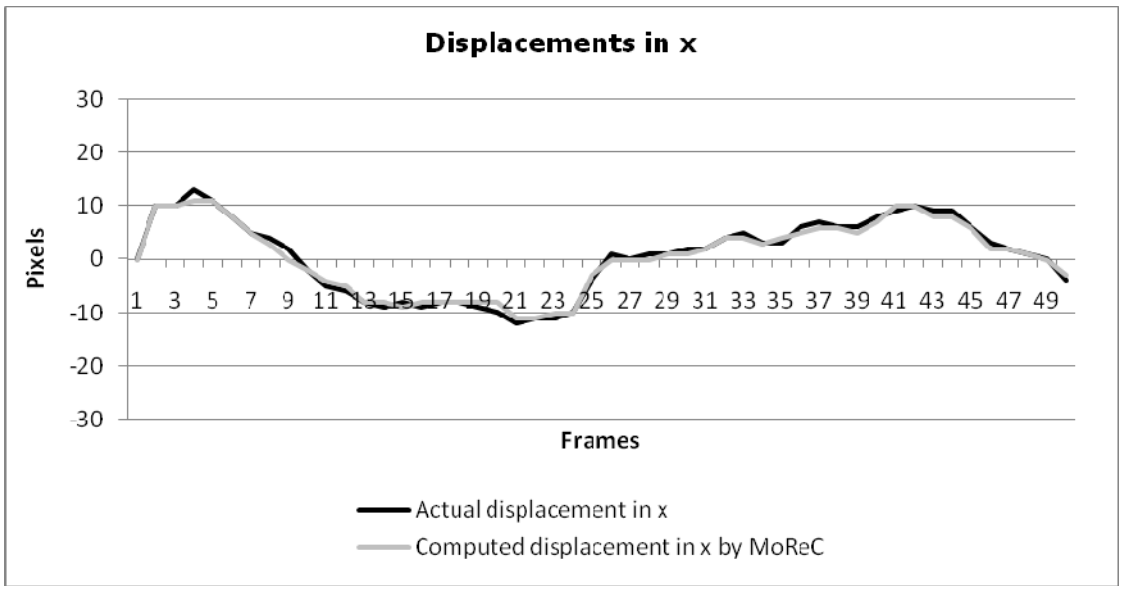
Figure 3.40 – Examples of search regions in *Shaky*

The inter-frame displacement vectors computed by *MoReC* are displayed in Figure 3.41 for the video *Shaky*. Figure 3.42 display the errors between the actual and computed displacement vectors. The mean and the variance of the total error, and the average time spent per frame are:

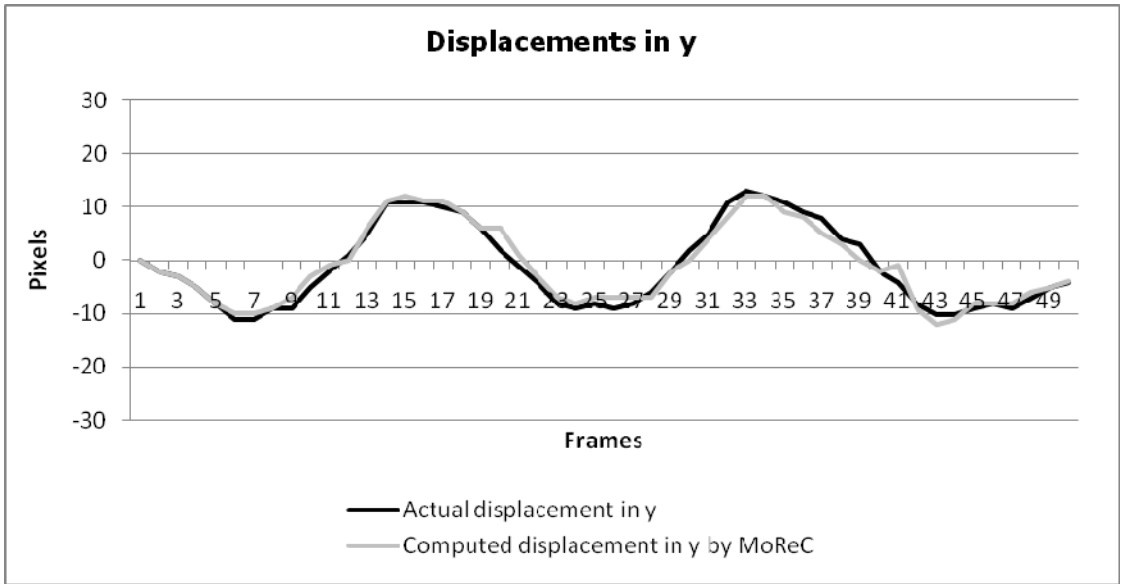
$\mu_E = 1.35$ pixels

$\sigma_E = 0.97$ pixels

$t_{avg} = 1.17$ s



(a)

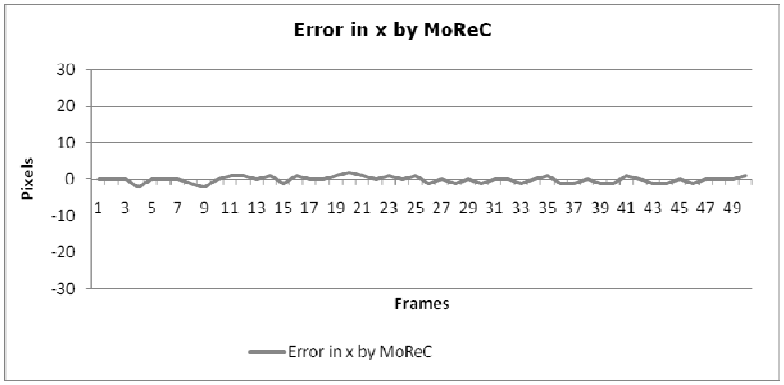


(b)

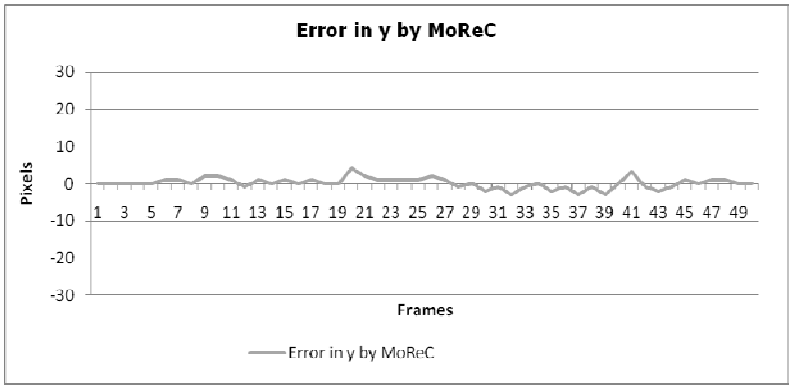
Figure 3.41 – Inter-frame displacement vectors computed by *MoReC* vs. ground truth data for *Shaky*

(a) x-component of the motion (inter-frame displacements in x direction)

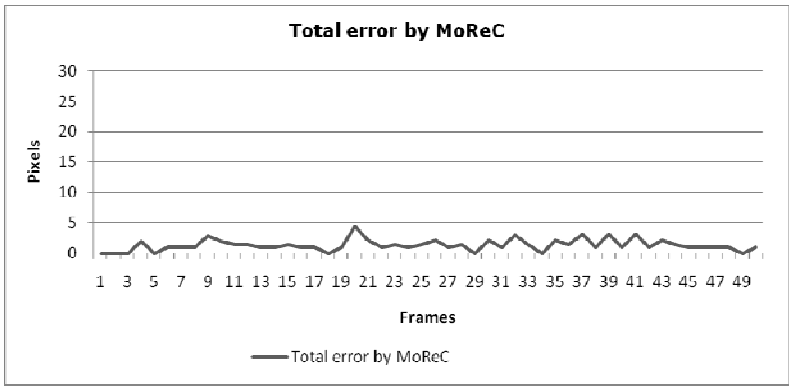
(b) y-component of the motion (inter-frame displacements in y direction)



(a)



(b)



(c)

Figure 3.42 – The error between the actual inter-frame displacement vectors and the ones calculated by *RC-based VASR* for *Shaky*

(a) Error for the x-component of the motion

(b) Error for the y-component of the motion

(c) Total error that equals the vector sum of (a) and (b)

Placing *MoReC* to the performance-cost space gives the plot in Figure 3.43. It can be seen that *MoReC* has considerably lower average processing time per frame with respect to *RC-based VASR*. This means that, narrowing the search regions resulted in a noticeable decrease in computational load while preserving the performance.

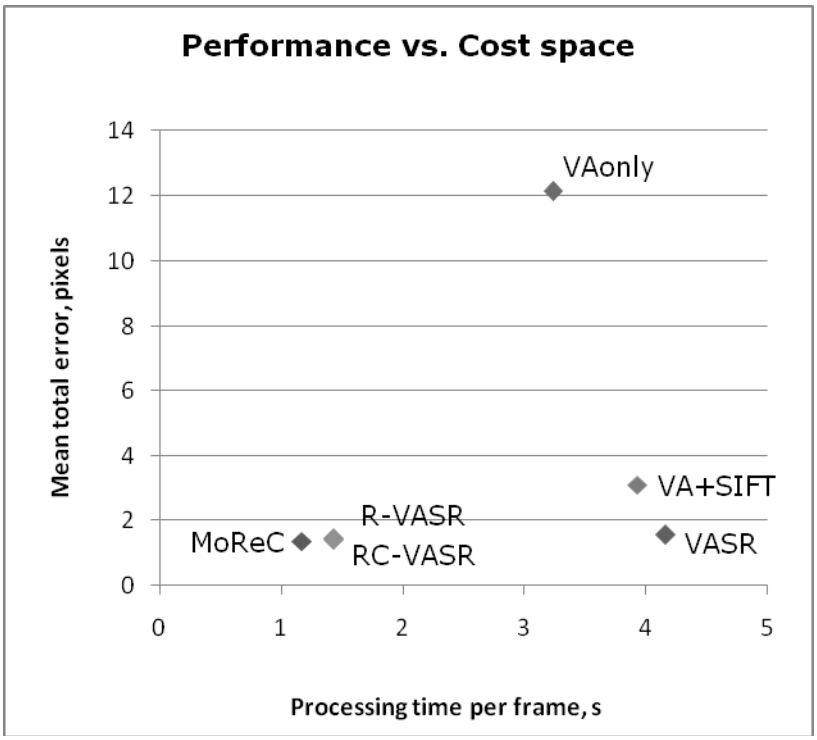


Figure 3.43 – The position of *MoReC* algorithm in "performance-computational cost" space
x-axis is the average time spent for processing a frame and y-axis is the mean of the total error between the displacement vectors computed by *MoReC* and the actual ones.

Figure 3.44 is the plot of actual and computed pixel coordinates of the two tracked regions in video *Planar* using *MoReC*. It can be seen that the error between the actual and computed positions of the salient regions is again small. This proves that the narrowed search regions did not result in a degraded tracking performance.

Evaluating *MoReC*, the computational cost and the performance of the algorithm are the best results obtained in this chapter.

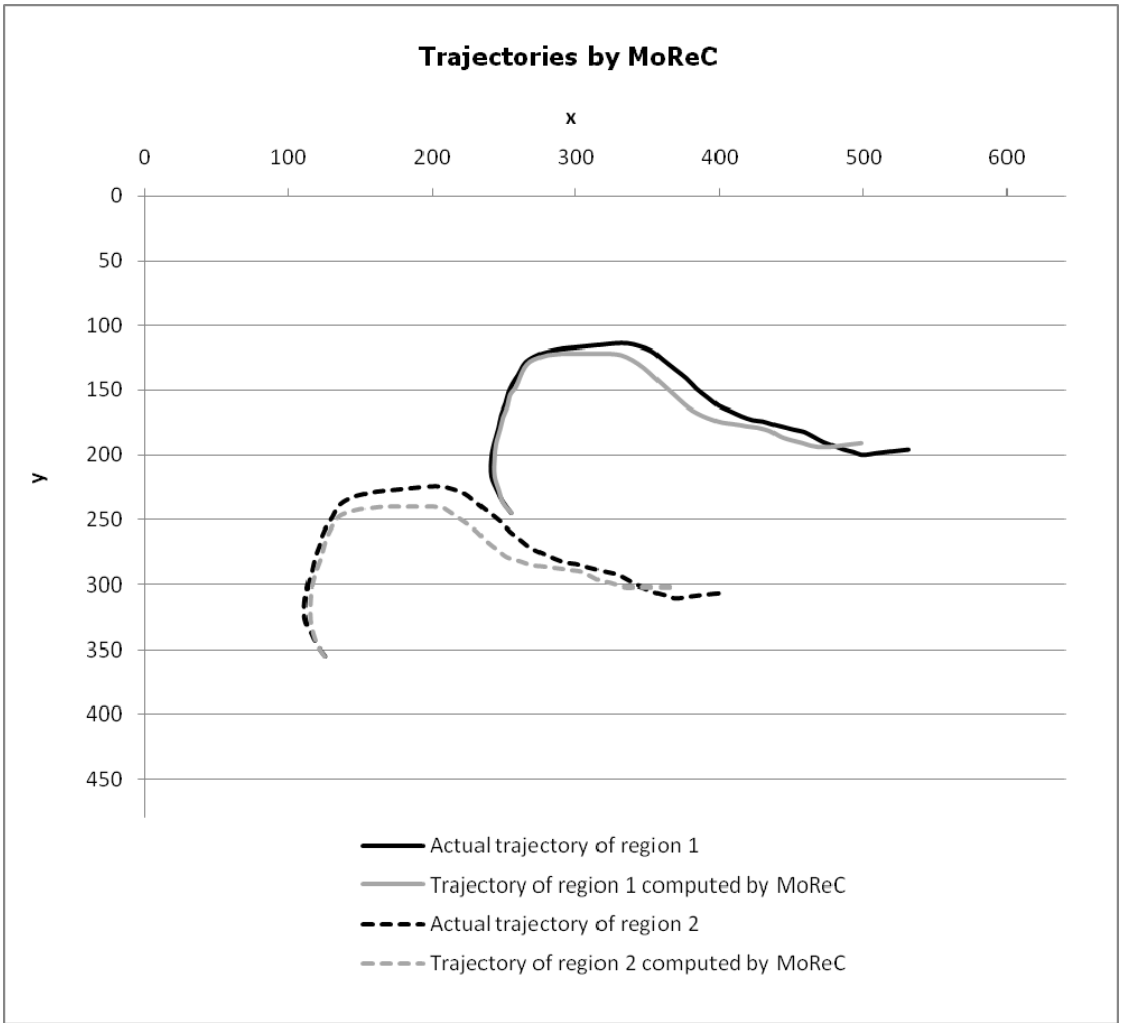


Figure 3.44 – Actual and computed positions of the tracked regions for *MoReC*

CHAPTER 4

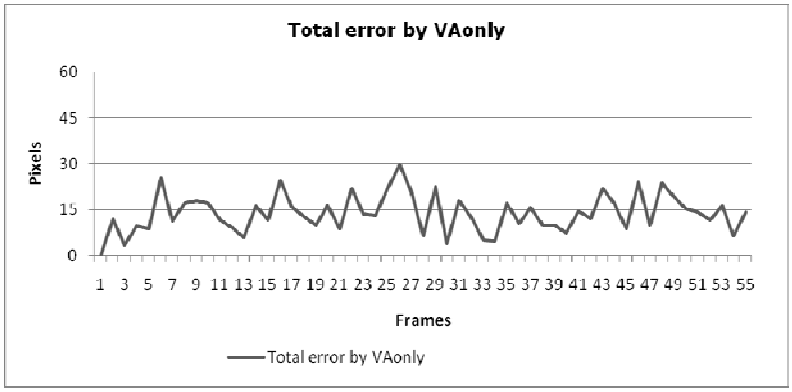
EXTENSIVE EXPERIMENTS

In chapter 3, several methods are proposed in an incremental manner for detecting and tracking salient regions. However, the test videos used in chapter 3 are specifically chosen to clearly display the gains with each proposed method. As a result of this, they were far from real life scenarios on account of the intentionally placed salient objects around the scene. In this chapter, the proposed methods are tested on more realistic video sequences. In section 4.1, the video *Blurred* (see section 3.1 for the test videos) is used to test the algorithm performances against motion blur. In section 4.2, the video *CarPark* is used to test the performances in an uncontrolled real life environment.

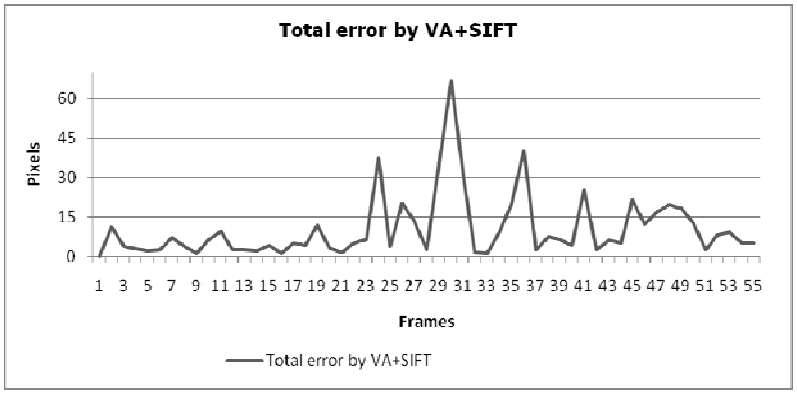
4.1 Motion Blur

The goal of this thesis is to obtain a salient region detector and tracker which can be used in mobile platforms. An inevitable phenomenon of the visual inputs of mobile systems is the motion blur. Thus, the methods proposed in chapter 3 should be tested against motion blurred videos.

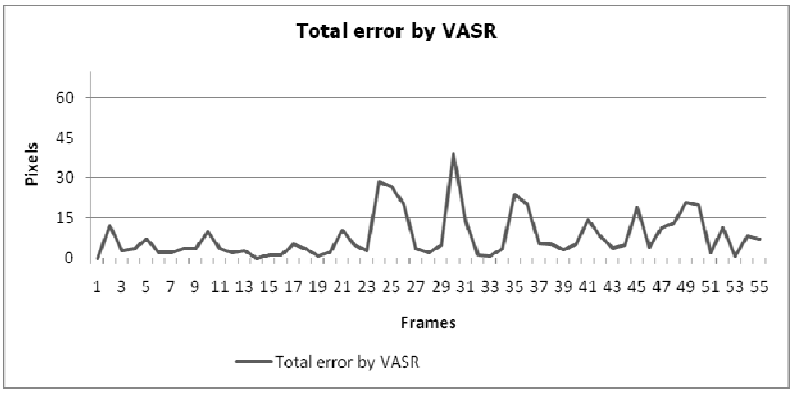
In this section the video *Blurred*, a highly motion blurred outdoor video, is used as the test video. Figure 4.1 displays the total errors between the actual inter-frame displacement vectors and the ones computed by the proposed algorithms.



(a) Total error of *VAonly* for *Blurred*



(b) Total error of *VA+SIFT* for *Blurred*

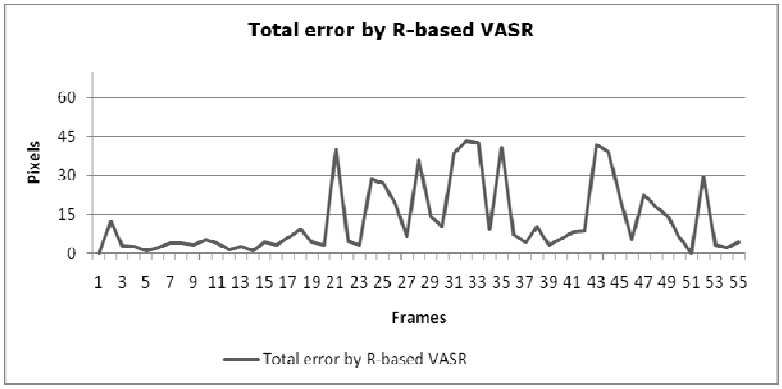


(c) Total error of *VASR* for *Blurred*

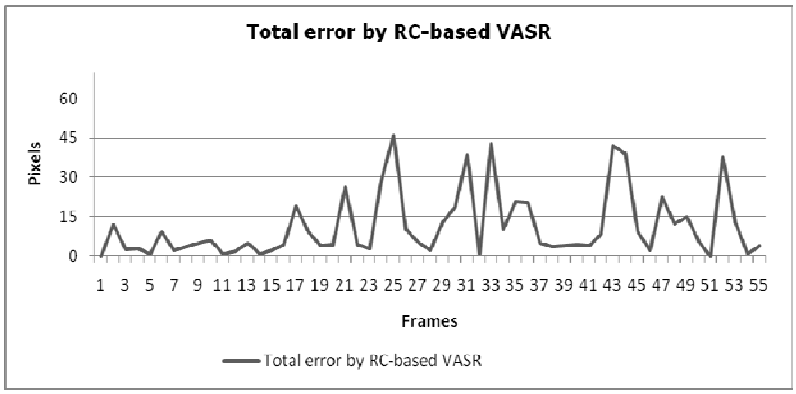
Figure 4.1 – Total errors of the proposed algorithms for video *Blurred*

(a) *VAonly* (b) *VA+SIFT* (c) *VASR*

(d) *R-based VASR* (e) *RC-based VASR*



(d) Total error of *R-based VASR* for *Blurred*



(e) Total error of *RC-based VASR* for *Blurred*

Figure 4.1 – Total errors of the proposed algorithms for video *Blurred* (continued)

Examining the plots in Figure 4.1, unlike the tests in chapter 3, the error of *VAonly* is not undoubtedly the highest one. The main reason of this is that SIFT algorithm is more sensitive to noise than *Visual Attention* algorithm.

Visual Attention detects salient region based on intensity, color and orientation discrepancies, which do not change much with motion blur. On the other hand, SIFT detects blob-like (Mexican hat-like) structures, which definitely change with motion blur. This can easily be easily realized by observing Figure 4.2, Figure 4.3 and Figure 4.5. In Figure 4.2, there are two sample frames, (a) without motion

blur and (b) with motion blur. Let the sharp frame be called *Frame A* and the blurred frame be called *Frame B*. Figure 4.3 (a) and (b) shows the SIFT points detected on these frames respectively. In *Frame A*, 7418 SIFT points are detected while in *Frame B* only 589 SIFT points are detected. These results reveal that SIFT algorithm is sensitive to motion blur. In fact most of the remaining points in *Frame B* can truly be matched to their counterparts in *Frame A*, as can be seen in Figure 4.4. Nevertheless, the decreasing number of points may cause problems especially when processing regions instead of full frame. On the other hand, the effects of motion blur on *Visual Attention* can be observed in Figure 4.5. (a), (b) and (c) parts of the figure are the intensity, color and orientation conspicuity maps of *Frame A* respectively. (d) is the saliency map of *Frame A*. Similarly, (e)-(h) are the conspicuity and saliency maps of *Frame B*. As it can easily be noticed, (a)-(d) are extremely similar to (e)-(h), telling that motion blur do not affect *Visual Attention* much.

As a result of these facts, the error levels of *VAonly* for video *Blurred* are comparable to the results of video *Shaky* (see section 3.3.1) while the errors of other algorithms have apparently increased.

Calculating the mean errors of the total errors in Figure 4.1 (a) to (f), and measuring the average processing times per frame, the algorithms can be placed on the performance-computational cost space as in Figure 4.6. In this figure, comparing *VAonly*, *VA+SIFT* and *VASR*, *VASR* displays the best performance as expected. However, adding SIFT features did not decrease the error enormously as in the video *Shaky*. The computational times of these three algorithms are almost equal. On the other hand, *R-based VASR* and *RC-based VASR* have slightly higher errors with much lower computational costs, meaning that region-based processing has increased the errors while decreasing the cost. It can be seen that limiting the attention channel in *RC-based VASR* has decreased the errors with respect to *R-based VASR*.

To sum up, *RC-based VASR* and *VASR* have the two best results in Figure 4.6. VASR stand out with low error value, while *RC-based VASR* stands out with low

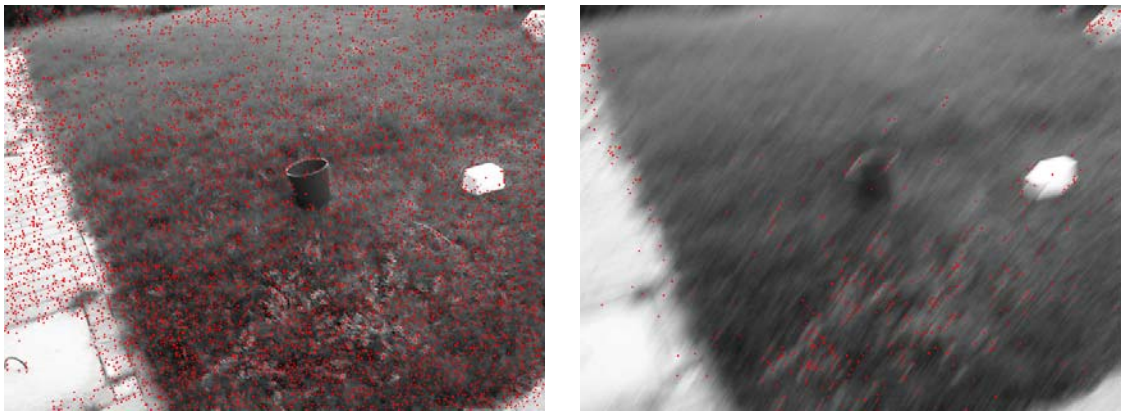
cost. The average error levels of the algorithms are between 8 and 14 pixels which correspond to 4 to 7 centimeters for the objects in the video which are approximately 3 meters away. In the light of these results, it can be concluded that the proposed algorithms, especially *RC-based VASR* and *VASR*, are proved to be functional with motion blurred videos.

(a) (b)

Figure 4.2 – Sample frames

(a) Without motion blur (Frame A)

(b) With motion blur (Frame B)



(a) (b)

Figure 4.3 – SIFT points

(a) SIFT points of Frame A

(b) SIFT points of Frame B

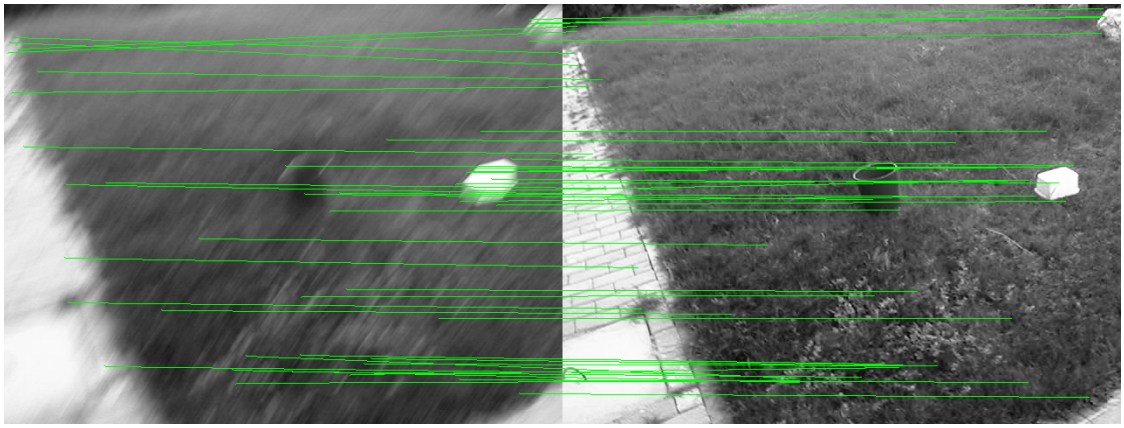


Figure 4.4 – SIFT point matches between *Frame A* and *Frame B*

Most of the SIFT points in *Frame B* can truly be matched to *Frame A*.

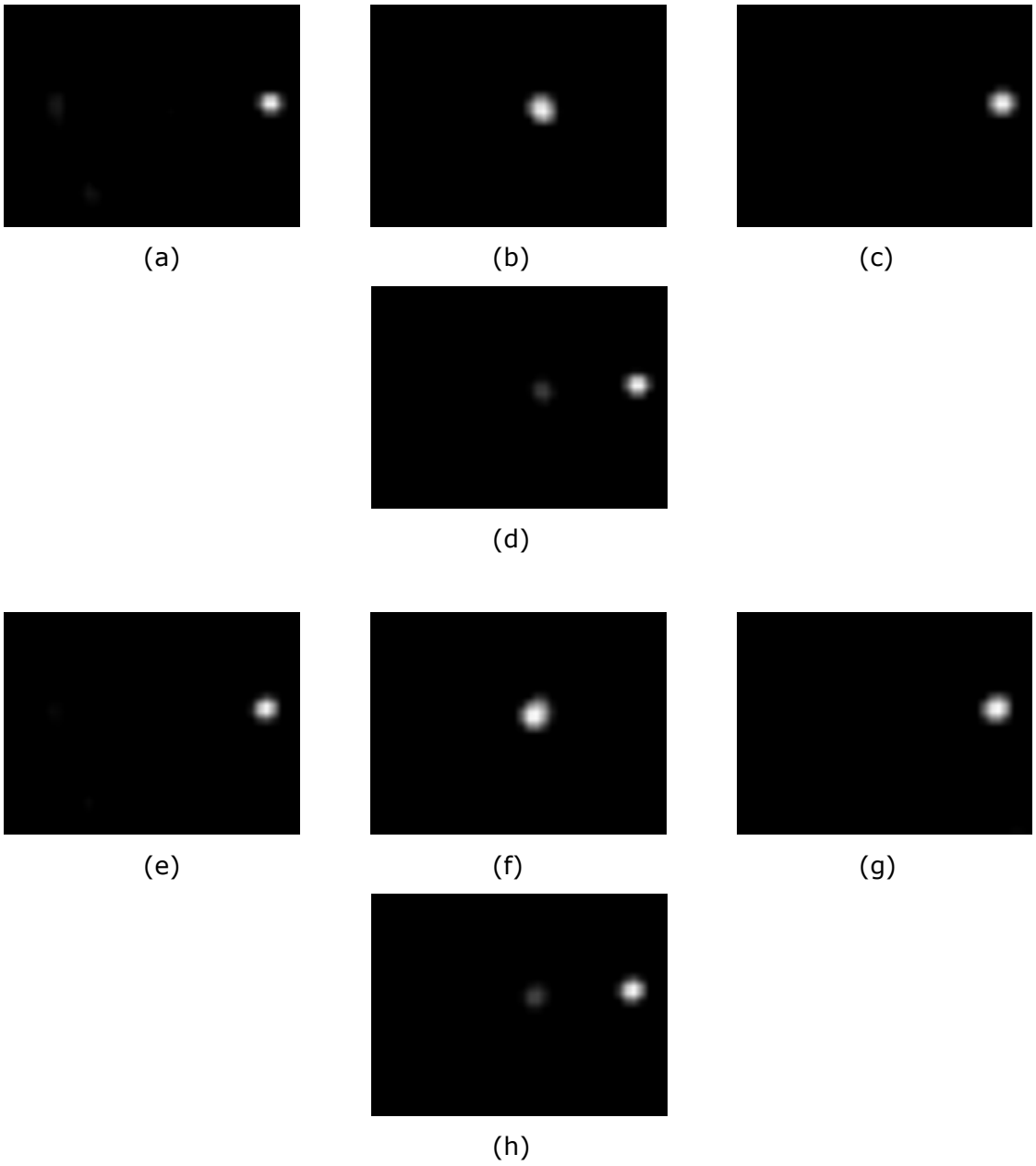


Figure 4.5 – Conspicuity and saliency maps

(a) Intensity conspicuity map of Frame A
(b) Color conspicuity map of Frame A
(c) Orientation conspicuity map of Frame A
(d) Saliency map of Frame A
(e) Intensity conspicuity map of Frame B
(f) Color conspicuity map of Frame B
(g) Orientation conspicuity map of Frame B
(h) Saliency map of Frame B

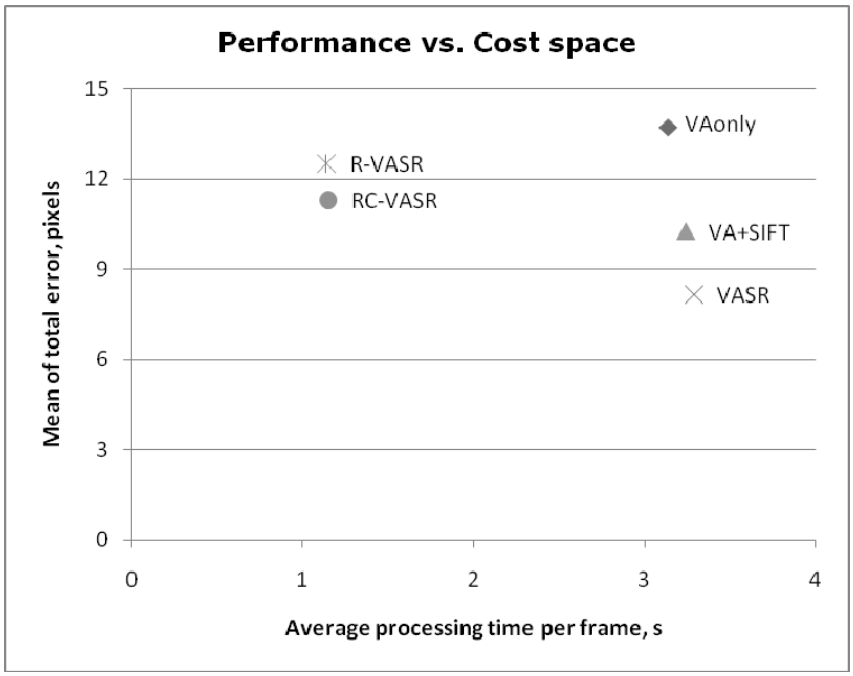


Figure 4.6 – The position of the algorithms in "performance-cost" space for video *Blurred*
x-axis is the average time spent for processing a frame and y-axis is the mean of the total error between the computed displacement vectors and the actual ones.

4.2 Real Life Environment

In this section, the test video is *CarPark*, which is an outdoor video sequence recorded in an uncontrolled environment (a car park with no intentionally placed objects). The aim of this experiment is to test the algorithms on more realistic videos.

In Figure 4.7, a sample frame of *CarPark*, its saliency map and examples of the tracked salient regions can be found.

The chief point in this test is that, the salient region tracking trajectories are generally much shorter with respect to other test videos; that is, the salient regions that are being tracked change many times throughout the video. This is because of the fact that the saliency values of many regions are comparable; thus, some of the tracked regions got lost while new regions turn out to be dominant.

In Figure 4.8, the trajectories of the tracked salient regions can be found for the test *RC-based VASR* on video *CarPark*. In this run, the number of salient regions to track is entered as 3. The positions of the tracked salient regions on image plane can be viewed throughout the frames. If the axis showing the frames is followed, it can be observed that there are 3 tracked salient regions at all times. Once a region gets lost during tracking, a new region is started to be tracked by creating a new trajectory for it. Since *RC-based VASR* is used in this run, region based processing is performed. Also a periodic full frame processing scheme exists with the period defined as 10, meaning full frame is processed once in every 10 frames. An interesting point is that most of the trajectory losses are occurred when full processing occurs, i.e., when frame count is a multiple of 10. From this result, it can be deduced that region-based processing in fact increases the continuity of the trajectories.

(a)

(b)

(c)

Figure 4.7 – Samples from video *CarPark*

(a) A sample frame from video *CarPark*

(b) The saliency map of the sample frame

(c) Examples of the salient regions that are tracked throughout the video

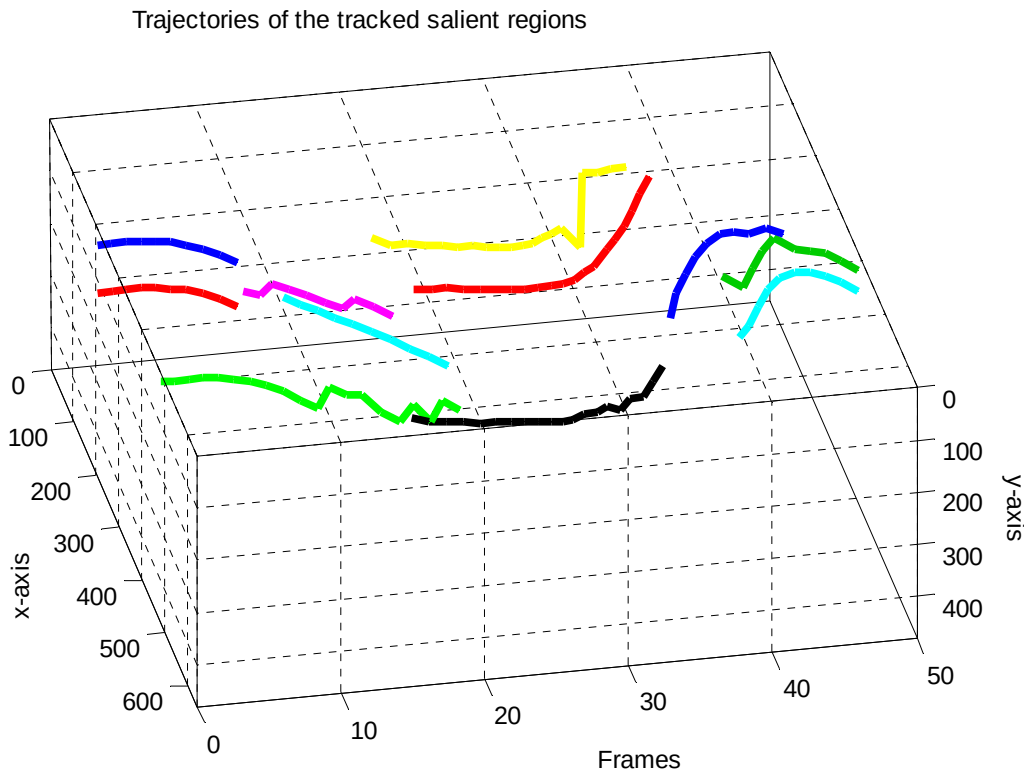


Figure 4.8 – Trajectories of the salient regions of *CarPark*, tracked by *RC-based VASR*
The number of salient regions to track is 3. This figure shows the positions of the tracked regions on image plane throughout the frames.

The algorithms are tested on the video *CarPark* and are placed on the performance-cost space as in Figure 4.9. It can be seen that adding SIFT and RANSAC to Visual Attention decreased the error while slightly increasing the cost. Region-based processing concept decreased the cost remarkably while increasing the error. Limiting the visual attention channels slightly decreased the augmented error. Consequently, *RC-based VASR* can be deduced to have the best statistics. Having an average error of 9 pixels and processing time per frame of 1.7 seconds, it can be used as a salient region detector and tracker in real life videos such as *CarPark*.

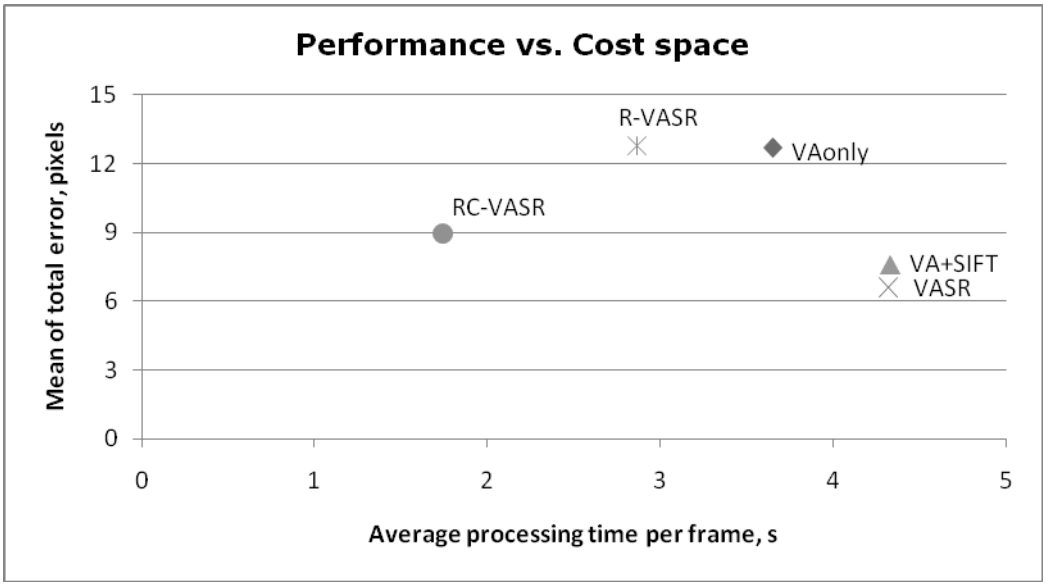


Figure 4.9 – The position of the algorithms in "performance-cost" space for video *CarPark*
x-axis is the average time spent for processing a frame and y-axis is the mean of the total error between the computed displacement vectors and the actual ones.

CHAPTER 5

CONCLUSION

Visual Attention is an interesting theory that tries to detect the regions of the image that are most likely to draw attention of human beings. Having designed in accordance with human judgment, Visual Attention is expected to be widely used in autonomous systems in near future. In this work, an efficient salient region detection and tracking method is proposed that can be used for visual processing in mobile platforms.

An efficient algorithm is incrementally obtained through a series of tests. First, the *Visual Attention* algorithm by Itti et al. [14] is used alone as a salient region detector and tracker. This algorithm detects the parts of the image which are different from their surroundings in terms of intensity, color and orientation. The visually attractive parts of the images are extracted with the help of the *Visual Attention* algorithm and a feature vector is extracted for each region. These regions are tracked across frames with the help of their positions and feature vectors. The tests showed that using *Visual Attention* is promising but its single performance and cost are not appropriate for practical usage.

Next, SIFT [18] features are used within the salient region to achieve more precise tracking results. The SIFT points of the salient region of current frame are matched with the ones in previous frame to compute the displacements. The tests displayed that this step significantly enhanced the tracking performances. What is

more, since SIFT algorithm is only applied to the salient region, the computational complexity of the algorithm only slightly increased.

Next, the performance is further improved by eliminating the wrong SIFT point matches using RANSAC algorithm [6]. This addition is proved to be functional with the help of the experimental results.

At this point, having achieved a satisfactory performance, the augmented computational cost became the weak point. The previously mentioned methods and tests are analyzed to find the most time consuming blocks. It is noticed that applying the *Visual Attention* algorithm takes the majority of the processing time. Since the time it takes to process a frame by *Visual Attention* most depends on the image size, possible ways of decreasing the image sizes are investigated. Since the input frames are parts of a video sequence, there is continuity in between. Thus the salient region may not move enormously between adjacent frames. Using this information, once finding the salient regions in frame, search regions are defined for the following frames to look for the salient regions. This region based processing scheme significantly enhanced the timings of the method. In addition, the performance is not affected considerably.

Next, techniques are searched to further decrease the computational cost. The proposed idea is limiting the visual attention channels. Similar to limiting the search regions, visual attention channels can also be limited due to the continuity of the frames. Once a salient region is found and its feature vector is extracted, the contributions of each channel to the saliency value are known. The idea is to use only the most dominant visual channel of a salient region when searching for it in subsequent frames. Surprisingly, this method did not decrease the computational costs significantly. However, as a surprising effect, it enhanced the performance. The reason of this is that limiting the visual attention channels increased the probability of finding the same salient region in successive frames. Last, the search regions are further limited using the extracted motion information. This final algorithm, which is obtained incrementally by adding a module in each step, is called *MoReC* as an abbreviation of "Motion and Region

and Channel based Visual Attention+SIFT+RANSAC". It displayed the best experimental results and accepted as a conclusion.

Having found a promising algorithm, additional tests are performed using more realistic data sets. First the algorithms are tested against motion blur. The results revealed that *Visual Attention* algorithm do not seem to be affected much by the motion blur while SIFT algorithm is significantly affected. As a result of this, the proposed methods displayed larger error values with the blurred data with respect to the previous results. However, the statistics of *RC-based VASR* were good enough so it can be deduced that this algorithm is useful even with motion blur.

Next, *RC-based VASR* is tested with a realistic video which is recorded in a car park. The characteristic of this video is that the environment is uncontrolled, with no intentionally placed objects. The experimental results showed that in an environment like that, the salient regions that the algorithm tracks changes more throughout the video. However, the algorithm managed to track given number of salient regions at all times thus achieving the task. In addition, the error level of the algorithm was acceptable, thus proving to be useful.

5.1 Future Work

As future work, the computational cost of the algorithm will tried to be decreased further since in mobile platforms the processing power is usually very limited. A method could be using a better motion estimation method, for example fitting a parabola. This would make it possible to further limit the search regions by estimating the region's future positions more accurately.

As another enhancement, the selection of salient regions to track can be performed in more intelligent ways. Currently, the salient regions that are visually most attractive are selected. This process can be improved by selecting the regions with some criteria that can be defined in relation with the intention.

As explained in section 1.1, there are several methods to extract the visual attention regions. Itti's method in [14] is utilized in this thesis as it is highly accepted in the literature and is regarded as "one of the best known attention systems available" [7]. As future work, a number of the other promising saliency detection algorithms can be utilized and evaluated in the proposed tracking scheme. In particular, the real time visual attention method using integral images, proposed in [7], can be tried to speed up the process and the method in [1] can be tried to obtain saliency maps at the original image resolution, thus to enhance the accuracy.

REFERENCES

[1] Achanta R., Estrada F., Wils P. and Süsstrunk S., "Salient Region Detection and Segmentation". *International Conference on Computer Vision Systems (ICVS)*, 2008.

[2] Bay H., Ess A., Tuytelaars T. and Gool L. V., "SURF: Speeded Up Robust Features". *Computer Vision and Image Understanding (CVIU),* Vol. 110, No.3, pp. 346-359, 2008.

[3] Bonaiuto J. and Itti L., "Combining attention and recognition for rapid scene analysis". *Proc. IEEE-CVPR Workshop on Attention and Performance in Computer Vision (WAPCV'05), San Diego, California,* pp. 1-6*,* 2005.

[4] Brown M. and Lowe D. G., "Invariant features from interest point groups". *British Machine Vision Conference, Cardiff, Wales,* pp. 656-665*,* 2002.

[5] Duncan J., "Selective attention and the organization of visual information". *Journal of experimental psychology. General*, Vol. 113, No. 4, pp. 501-517*,* 1984.

[6] Fischler M. and Bolles R., "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography". *Communications of the ACM,* vol. 24, No. 6, pp. 381-385*,* 1981.

[7] Frintrop S., Klodt M. and Rome E., "A Real-time Visual Attention System Using Integral Images". *The 5th International Conference on Computer Vision Systems,* 2007.

[8] Gao K., Lin S., Zhang Y., Tang S., Ren H., "Attention Model Based SIFT Keypoints Filtration for Image Retrieval". *Seventh IEEE/ACIS International Conference on Computer and Information Science*, 2008.

[9] Harris C. and Stephens M., "A combined corner and edge detector". *4th Alvey Vision Conf.* pp. 147-151, 1988.

[10] iLab Neuromorphic Vision C++ Toolkit (iNVT). 28 April 2009 21:53:01 PDT. Available from http://ilab.usc.edu/toolkit/home.shtml. Retrieved 5 October 2008.

[11] Itti L., "Automatic Foveation for Video Compression Using a Neurobiological *Model of Visual Attention". IEEE Transactions on Image Processing,* Vol. 13, No. 10, pp. 1304-1318, 2004.

[12] Itti L., Gold C. and Koch C., "Visual Attention and Target Detection in Cluttered Natural Scenes". *Optical Engineering,* Vol. 40, No. 9, pp. 1784-1793*,* 2001.

[13] Itti L. and Koch C., "Target Detection using Saliency-Based Attention" *Proc. RTO/SCI-12 Workshop on Search and Target Acquisition (NATO Unclassified), Utrecht, The Netherlands, RTO-MP-45 AC/323(SCI)TP/19,* pp. 3.1-3.10, 1999.

[14] Itti L., Koch C. and Niebur E., "A model of saliency-based visual attention for rapid scene analysis". *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI),* Vol. 20, No. 11, pp. 1254-1259, 1998.

[15] Koch C. and Ullman S., "Shifts in selective visual attention: Towards the underlying neural circuitry". *Human Neurobiology,* Vol. 4, pp. 219-227, 1985.

[16] Li Y., Ma Y. F. and Zhang H. J., "Salient region detection and tracking in video". *International Conference on Multimedia and Expo, ICME '03. Proceedings*, 2003.

[17] Li Z. and Itti L., "Visual attention guided video compression". *Proc. Vision Science Society Annual Meeting (VSS08),* 2008.

[18] Lowe D., "Distinctive image features from scale-invariant keypoints". *International Journal of Computer Vision (IJCV)*, 2004.

[19] Ma Y. F. and Zhang H. J., "Contrast-based image attention analysis by using fuzzy growing". *ACM Multimedia 2003*, pp. 374–381*,* 2003.

[20] Mikolajczyk K., "Detection of local features invariant to affine transformations". *PhD. Thesis, Institut National Polytechnique de Grenoble, France*, 2002.

[21] Navalpakkam V. and Itti L., "An Integrated Model of Top-down and Bottom-up Attention for Optimal Object Detection". *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 2049-2056, 2006.

[22] Ouerhani N. and Hugli H., "MAPS: Multiscale attention-based pre-segmentation of color images". *4th Intternational Conference on Scale-Space theories in Computer Vision, Springer Verlag, Lecture Notes in Cotnputer Science (LNCS),* Vol. 2695, pp. 537-549*,* 2003.

[23] Ouerhani N. and Hugli H., "A model of dynamic visual attention for object tracking in natural image sequences". *International Conference on Artificial and Natural Neural Network (IWANN), Springer Verlag, Lecture Notes in Computer Science (LNCS).* Vol. 2686, pp. 702-709, 2003.

[24] Ouerhani N., Hugli H., Gruener G. and Codourey A., "A visual attention-based approach for automatic landmark selection and recognition". *Attention and Performance in Computational Vision, WAPCV 2004, Lecture Notes in Computer Science, Springer Verlag, LNCS 3368,* pp. 183-195, 2005.

[25] Rosbergen E., Pieters R. and Wedel M., "Visual Attention to Advertising: A Segment-Level Analysis". *The Journal of Consumer Research*, Vol. 24, No. 3, pp. 305-314, 1997.

[26] Salience (neuroscience). 29 August 2009, 20:10 UTC. *Wikipedia: The Free Encyclopedia*. Wikimedia Foundation Inc. Encyclopedia on-line. Available from http://en.wikipedia.org/wiki/Salience_(neuroscience). Internet. Retrieved 1 September 2009.

[27] Siagian C. and Itti L., "Biologically Inspired Mobile Robot Vision Localization". *IEEE Transactions on Robotics,* Vol. 25, No. 4, pp. 861-873, 2009.

[28] Siagian C. and Itti L., "Biologically-Inspired Robotics Vision Monte-Carlo Localization in the Outdoor Environment". *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2007.

[29] Siagian C. and Itti L., "Gist: A Mobile Robotics Application of Context-Based Vision in Outdoor Environment". *Proc. IEEE-CVPR Workshop on Attention and Performance in Computer Vision (WAPCV'05), San Diego, California,* pp. 1-7*,* 2005.

[30] Vedaldi A., (n.d.) *A. Vedaldi – Code – SIFT++*. Available from http://www.vlfeat.org/~vedaldi/code/siftpp.html. Retrieved 9 December 2008.

[31] Walther D., Itti L., Riesenhuber M., Poggio T. and Koch C., "Attentional Selection for Object Recognition - A Gentle Way". *Lecture Notes in Computer Science,* Vol. 2525, pp. 472-479*,* 2002.

[32] Walther D., Rutishauser U., Koch C. and Perona P., "On the Usefulness of Attention for Object Recognition". *Workshop on Attention and Performance in Computational Vision at ECCV,* pp. 96-103, 2004.

[33] Yu Z. and Wong H. S., "A Rule Based Technique for Extraction of Visual Attention Regions Based on Real-Time Clustering". *IEEE Transactions on Multimedia*, Vol. 9, No. 4*,* 2007.