DETECTING AND TRACKING MOVING OBJECTS WITH AN ACTIVE CAMERA IN REAL TIME

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

BY

SAMET KARAKAŞ

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

SEPTEMBER 2011

Approval of the Thesis

DETECTING AND TRACKING MOVING OBJECTS WITH AN ACTIVE CAMERA IN REAL TIME

Submitted by **SAMET KARAKAŞ** 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. İlkay ULUSOY
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members

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

Assist. Prof. Dr. İlkay ULUSOY
Electrical and Electronics Engineering, METU _____

Prof. Dr. A. Aydın ALATAN
Electrical and Electronics Engineering, METU _____

Prof. Dr. Nihan Kesim ÇİÇEKLİ
Computer Engineering, METU _____

Ali Erkin ARSLAN (M.Sc.)
MGEO, ASELSAN A.Ş. _____

Date: 09.09.2011

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 : Samet KARAKAŞ

Signature :

ABSTRACT

DETECTING AND TRACKING MOVING OBJECTS WITH AN ACTIVE CAMERA IN REAL TIME

KARAKAŞ, Samet

M.Sc., Department of Electrical and Electronics Engineering
Supervisor: Asst. Prof. Dr. İlkay ULUSOY

September 2011, 80 pages

Moving object detection techniques can be divided into two categories based on the type of the camera which is either static or active. Methods of static cameras can detect moving objects according to the variable regions on the video frame. However, the same method is not suitable for active cameras. The task of moving object detection for active cameras generally needs more complex algorithms and unique solutions. The aim of this thesis work is real time detection and tracking of moving objects with an active camera. For this purpose, feature based algorithms are implemented due to the computational efficiency of these kinds of algorithms and SURF (Speeded Up Robust Features) is mainly used for these algorithms. An algorithm is developed in C++ environment and OpenCV library is frequently used. The developed algorithm is capable of detecting and tracking moving objects by using a PTZ (Pan-Tilt-Zoom) camera at a frame rate of approximately 5 fps and with a resolution of 640x480.

Key Words: Visual Surveillance, Real Time, Active Camera, Moving Object Detection, Object Tracking

ÖZ

HAREKETLİ KAMERA KULLANARAK GERÇEK ZAMANLI HAREKETLİ NESNE ALGILAMASI VE TAKİBİ

KARAKAŞ, Samet

Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Yrd. Doç. Dr. İlkay ULUSOY

Eylül 2011, 80 sayfa

Hareketli nesne algılama yöntemleri kullanılan kameraya göre sabit ve hareketli olmak üzere iki grupta incelenebilir. Sabit kamera kullanan yöntemler, hareketli nesneleri görüntü üzerinde değişim gösteren bölgeleri inceleyerek anlayabilmektedirler. Ancak, kameranın hareketli olması durumunda bu inceleme yeterli olmamaktadır. Bu sebeple hareketli kamera ile nesne algılama yöntemleri genel olarak daha karmaşık algoritmalar ve özgün yaklaşımlar gerektirmektedir. Bu tez çalışmasında, hareketli kamera ile çekilen görüntüler üzerinden hareketli nesnelerin gerçek zamanlı olarak algılanması ve takibi amaçlanmıştır. Bu amaç doğrultusunda, gerçek zaman performansı daha iyi olan öznitelik tabanlı algoritmaların kullanılmasına karar verilmiş ve öznitelik olarak SURF (Speeded Up Robust Features) seçilmiştir. C++ ortamında OpenCV kütüphanesi kullanılarak geliştirilen algoritma; bir PTZ (Pan-Tilt-Zoom) kamera üzerinde gerçek zamanlı çalışarak, yaklaşık 5fps hızında ve 640x480 çözünürlükte hareketli nesne algılama ve takibi işlemlerini gerçekleştirebilmektedir.

Anahtar Kelimeler: Görsel Gözetim, Gerçek Zamanlı, Hareketli Kamera, Hareketli Nesne Algılama, Nesne Takibi

To My Beloved Family and Lovely Wife

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere thanks to my supervisor Assist. Prof. Dr. İlkay ULUSOY for her supervision and guidance throughout this study.

I would like to thank to my colleagues for their continuous assistance and I would like to thank to ASELSAN A.Ş. for the support given during my thesis work.

I would also like to thank TUBITAK for their financial support during my MSc. study.

Lastly I would like to present my special thanks to my parents and my wife for their continuous love, encouragement and patience during my thesis study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

For a few decades, surveillance is a continuously growing application area due to the increasing needs of the society. Surveillance equipments are important tools for both military and civilian applications. Border security, target tracking, target detection, night vision applications are just a few examples on military area. Security cameras on crowded areas and traffic monitoring are examples for civilian applications of surveillance. Moreover, improvements in camera hardware and reductions in prices encourage the widespread usage of surveillance tools.

A recent survey [1] reveals that according to some human rights groups, in 2005 there were 4 million surveillance cameras in England. It is equivalent to 1 camera for every 17 people in the country. According to Dee and Velastin [1], only a tiny fraction of these videos are ever evaluated because most of these cameras are operated by humans. Beside, most of the surveillance videos are even not displayed in a screen and just recorded to watch afterwards in case of an emergency situation. Human operators, can not concentrate on the screen all the time and they suffer boredom thus it is very possible that a human operator might miss an important event on a real time video. These facts strengthen the importance of automated visual surveillance.

The main goal of automated visual surveillance is to extract specific and high level information from the input video frames without needing human operators [2]. Automated visual surveillance is a general name for a group of applications in

computer vision. Some examples of its subjects are object detection, object tracking, video stabilization and human action recognition. The number of examples can be increased in the areas of space, military, medical, and urban security applications. Automated visual surveillance algorithms possess some important advantages with respect to human operators. By using surveillance software, operating costs can be decreased drastically. A robust algorithm does not suffer any concentration loss or boredom and can be operational 24 hours a day.

This thesis mainly concentrates on object detection and object tracking in active camera. Both subjects are very popular in computer vision community and there are a wide variety of papers published in the literature. Object detection and tracking in static cameras is an older subject and relatively more effort has been expanded. Due to the nature of static camera, video processing is easier. A stable background is useful for recognizing mobile targets. However, for active cameras, moving object detection task is not trivial. A stable background can not be obtained since line of sight of the camera is continuously changing. Thus algorithmic complexity increases for active camera surveillance applications. Yet active cameras have an important advantage to be preferred. To observe a wide area, one static camera is not sufficient. In most cases, a few static cameras should be assembled in different angles in order to view the subject area completely. However with only a single pan tilt zoom (PTZ) camera, a wide area can be observed. Beside, the camera can focus and zoom on a suspicious object and more detail can be gathered compared to a static camera.

1.2 Scope of the Thesis

In this thesis, it is aimed to develop a combined autonomous detector - tracker system for an active camera which is capable of panning, tilting and zooming. Proposed algorithm is able to work real time with a video resolution of 640x480. The resolution value is superior to most of the current studies in the literature [3]. Minimum 5 Hz computation frequency is aimed. Assuming that target objects are

far from the camera and the speed of their reflection on the image plane is slow, this frequency is sufficient for this thesis work. The detector is capable of detecting moving objects while the camera is panning, tilting and zooming as well as it is stable. Detected objects can be tracked with a single object tracker. The proposed algorithm can be divided into three main parts which is demonstrated in Figure 1-1.

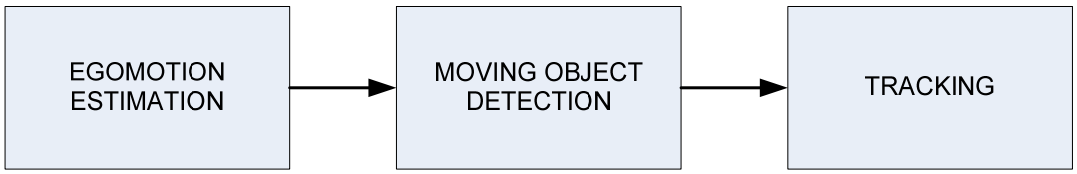


Figure 1-1 General flow diagram for the proposed algorithms

The detector determines the moving objects on the input image and locates them to user. User enables tracking and then the tracker aims to direct the camera to the target object as long as the object is in the line of sight of the PTZ camera. Moreover during tracking, tracker is expected to zoom on the target object up to a distance. The detector - tracker system is designed to be robust to typical challenges of a computer vision system as much as possible.

In this thesis, a small video database which consists of videos taken from a PTZ camera is constructed. A camera egomotion estimation algorithm is implemented and the algorithm is experimented with the videos in the database. Then two kinds of detector algorithms are implemented and compared. Finally a simple tracker algorithm is added to the system in order to direct the camera to a specific target. Some experiments are taken with the final detector – tracker system.

1.3 Outline of the Thesis

This thesis work consists of five chapters. The first chapter introduces the subject to the reader and clarifies the main aim of the thesis. The second chapter is a short

summary of the literature for the subject of moving object detection and tracking. Existing solutions for the current problem are mentioned, weak and powerful properties of each method are explained. In the third chapter, the egomotion estimation algorithms, both of the two motion detection algorithms and the tracker algorithm are explained in detail. All the steps of the final tracking system are mentioned. In the fourth chapter, the test setup, hardware and software combinations are presented and the experiments carried out are explained. Moreover the two object detector algorithms are compared in this chapter. In the final chapter, the thesis work and the results are summarized and, future work of this thesis is explained.

CHAPTER 2

MOTION DETECTION AND TRACKING FOR ACTIVE CAMERA

Motion detection and tracking has been widely studied for many years since the subject is intensively used both in commercial and military electronics. Therefore, there are a wide variety of motion detection and tracking methods in the literature. Some of the algorithms are well developed and have a very satisfactory performance; nevertheless still there are some unsolved problems in the area.

Noise in images is one of the problems for a typical tracking system. In real life scenarios, the input video may be noisy and a robust tracking system should be tolerant to noise up to some extend [4]. Blurring is also a potential problem. For instance, in a PTZ camera, if camera moves through pan or tilt direction excessively fast and if the shutter speed is relatively slow, blurring may occur. In such a case, algorithms which rely on features like blobs or corners may fail. Desired features on the image might be lost due to blurring. Changes in illumination are another challenging situation for surveillance applications [4]. Due to the angle of the light source and different type of whether conditions, pixels of the same scene may change dramatically. Thus a robust tracking system should withstand such kinds of variations.

If a tracking system aims on specific objects like cars or pedestrians, it may take advantage of the initial knowledge about the shape of the target object [4]. However in some applications there may be no priori knowledge about the target object. In such cases a tracking system should work on targets with various shapes. Moreover

the target might be a non-rigid object such that the shape of the target can change while moving. For instance while a pedestrian is walking, the shape of the pedestrian continuously changes. Thus a direct implementation of cross correlation will probably fail for pedestrians [4]. Repetitive sequences on the background can also be a problem for surveillance applications. That is because background information is particularly important while finding the egomotion of the camera in an application on active cameras.

It is very possible that an object of interest might appear behind an obstacle in a video sequence. This is called occlusion in the literature. An object of interest can be partially or fully occluded behind another object. Kalman filters or Particle filters can be employed in order to solve this problem [4]. Another approach to handle occlusion problem is using multiple cameras and relating same objects from different perspectives [4].

2.1 Motion Detection

Motion detection algorithms can be divided into some sub categories. In case of computational performance, motion detection algorithms can be divided into two categories which are online and offline. If real time performance is a necessity online algorithms should be employed. A smooth detection performance can be obtained with an algorithm which works faster than 25 Hz because this is the working frequency of an ordinary human eye. However, even with 5-6 Hz, a slowly moving object can be detected and tracked in real time [5]. Offline algorithms work more slowly. They are not suitable for real time applications nevertheless; in some applications they can be employed on formerly recorded videos.

In case of camera type, motion detection algorithms divide into two categories; algorithms for stationary cameras and algorithms for active cameras such as PTZ cameras [4].

2.1.1 Motion Detection on Stationary Cameras

Motion detection on stationary cameras is an older subject. It has been widely studied for many years and lots of improvements have been taken on the subject. Existing methods can be divided into three categories which are Temporal Differencing, Background Subtraction and Optical Flow based detection [2].

2.1.1.1 Temporal Differencing

Temporal differencing is one of the most primitive motion detection methods in image processing. Consecutive frames are directly subtracted from each other and resulting pixels above a threshold are considered to belong to a moving object [2]. The choice of the threshold value is critical in case of the performance of the algorithm. Also it is a known fact that this threshold value is application and background dependent. Temporal Differencing is superior to any other algorithms in case of computational performance. However, this algorithm is vulnerable to illumination and angle of light changes. Moreover it cannot be applied to active cameras unless there is a camera motion compensation algorithm [2].

2.1.1.2 Background Subtraction

Background subtraction is a well known common technique for motion detection [2]. The scene is examined for a few frames and statistical variations are calculated for each pixel. Then a reference image is constructed. In the reference image, there exists only the stable objects of the scene and dynamic objects are eliminated [4]. After constructing the background, frame differencing is applied between the current frame and the background image in order to spot moving objects on the scene [2]. In Figure 2-1 [4], an implementation of background image construction and frame differencing is demonstrated. Frame (a) is the current frame and frame (b) is the calculated background image. Note that the walking man does not appear

on the background image. Finally in frame (d) the moving object is detected and other parts of the scene is deleted.

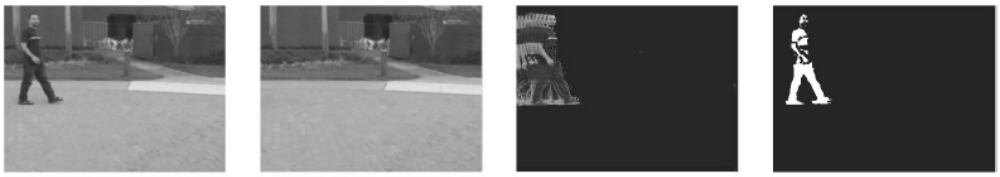
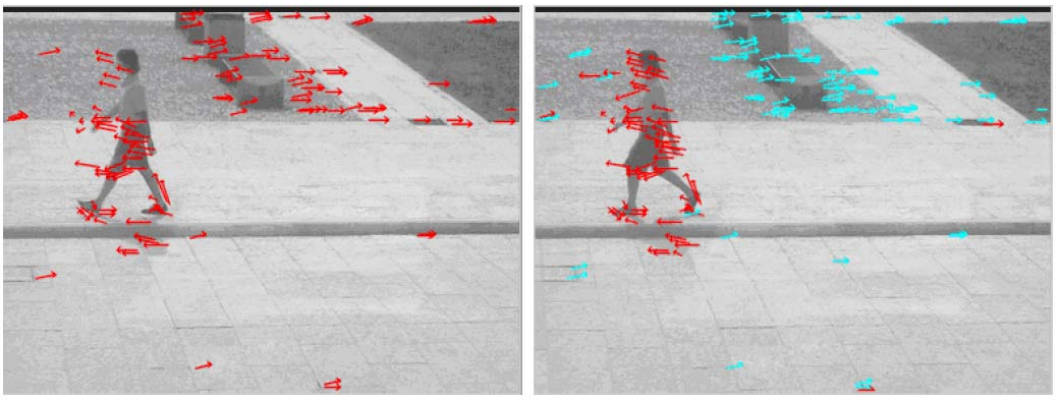


Figure 2-1 Background construction and foreground extraction example [4]

2.1.1.3 Optical Flow Based Methods

Optical flow based methods are an alternative solution of motion detection for both static and active cameras [2]. Motion vectors of the current frame are extracted. These motion vectors are clustered in case of their direction as well as their position on the image. Then moving objects are estimated based on the clustered group of motion vectors.

Figure 2-2 [5] is an example implementation of optical flow. In frame (a) motion vectors found by Lucas Kanade algorithm [6] are demonstrated and in frame (b) the motion vectors are clustered into two categories. Optical Flow will be further mentioned in part 2.1.2.2.



(a) (b)

Figure 2-2 Optical flow based object detection [5]

2.1.2 Motion Detection on Active Cameras

Since PTZ (Pan Tilt Zoom) cameras became widespread in the market, the importance of motion detection on active cameras increased considerably. Although they are not as common as static camera algorithms, there are a wide variety of motion detection algorithms for active cameras in the literature.

In an active camera, background subtraction can not be directly used as it is applied in static camera videos. For any movement of the camera, the background information totally changes and any frame differencing technique can not be employed directly [2]. In order to apply frame differencing, firstly the self movement of the camera should be figured out. In the literature, “Egomotion“ is used as another name for the self movement of the camera [5]. Egomotion information is used to reverse the movement of the next frame with respect to the previous frame. Then classical frame differencing algorithms can be employed on these two frames. Finally the resulting image is properly thresholded, filtered and also some morphological operations might be applied on the difference image to define the moving objects between these two frames. In Figure 2-3 an example procedure is given.

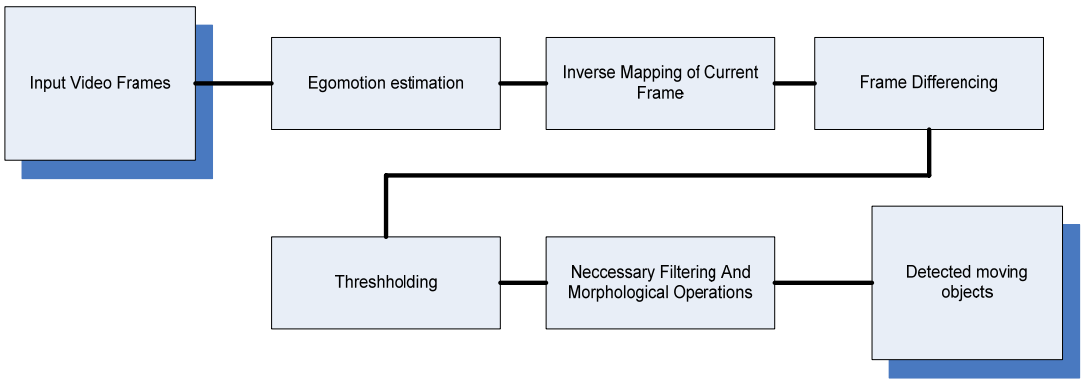


Figure 2-3 An example procedure for active camera motion detection

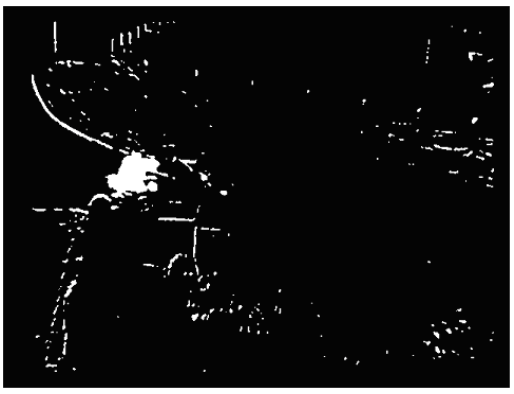
Figure 2-4 is an example MATLAB implementation of motion detection with an active camera. Frame (a) is the previous frame and frame (b) is the next frame. Only translational movement is expected and egomotion is calculated accordingly. It is given that the next frame is shifted to the left by 15 pixels thus inverse shift operation is applied to that image. Frame (c) is the inversely mapped next frame. Frame differencing is applied and the resulting image in frame (d) is found. Then a threshold is applied to the difference image such that pixels lower than the threshold, are discarded and pixels higher than the threshold are kept and assigned to a high value. In order to get rid of small point wise noise, median filter is employed in frame (e). Finally morphological opening operation is applied and frame (f) is obtained.

(a)

(b)

(c)

(d)



(e)

(f)

Figure 2-4 Matlab implementation of the example motion detection procedure

According to Kim [2], active camera motion detection algorithms can be divided into four categories in case of working mechanisms which are detection by camera geometrical properties, detection by optical flow, detection by background mosaic and detection by feature matching.

2.1.2.1 Motion Detection by Camera Geometrical Properties

In [7] and [8], camera geometrical parameters are employed to construct a stable background on PTZ camera videos. An algorithm that uses focal length data was proposed by Murray and Basu [7]. Together with the focal length data, Kang [8] uses an additional intrinsic parameter which is the size of the CCD sensor. Both algorithms also use pan and tilt data coming from the camera. Only translational movement is assumed and a background image is constructed with the help of the camera parameters. The reference image is subtracted from the background image in order to detect motion.

Usage of camera parameters for active camera motion detection is a promising concept however, it has a significant disadvantage. These types of algorithms need very accurate measurements of camera parameters such as focal length and pan/tilt angle variations [2]. However, sufficient accuracy cannot be obtained from standard commercial cameras of today and the errors in the measurements of camera parameters may cause the algorithm to fail. Only pan/tilt information of the camera is not sufficient to estimate translational shift of the image pixels. Exact focal length of the camera is also obligatory [8]. Exact measurement of this parameter needs complex hardware solutions yet lens distortions have a negative effect on the measurement. Moreover any solution of this kind will be hardware dependent and an algorithm for a camera should be recalibrated for another camera.

2.1.2.2 Motion Detection by Optical Flow

Optical flow is a promising candidate as a solution for motion detection on active camera. Sugaya and Kanatani [9] assume pure translational movement between

different frames of a PTZ camera and make use of Lukas Kanade Tracker [6] in order to find egomotion of the camera. They [9] use not only former two frames but "m" recent frames and try to improve the performance of the algorithm. Then the reference frame is inversely mapped and background subtraction is applied. Sugaya and Kanatani [9] obtain some good results on movement detection however the algorithm is off line due to algorithmic complexity.

Cucchiara [5] suggests a pyramidal implementation of KLT algorithm in order to improve performance in case of computational time. Pure translational movement is assumed and two direction histograms for pan and tilt angles of a PTZ camera are constructed. Then a Gaussian filter is applied on the histograms and only the dominant angles of the histograms remain. These angles indicate the egomotion of the camera. Similar to other algorithms, reference image is inversely mapped with the egomotion and background subtraction is applied. It is stated that [5] frame differencing is not adequate to obtain a resulting image of pure motion without noise. Morphological operations are necessary to eliminate noise and some connected component labeling operations are applied to obtain the complete silhouette of the moving object. Cucchiara [5] claims that his algorithm can work real time such that average 5 or 6 frames per second can be processed.

2.1.2.3 Motion Detection by Mosaic Imaging

Bevilacqua and Azzari [3] define that "A mosaic is a compound image built through properly composing (aligning) a high number of frames and warping them into a common reference coordinate system, both spatial and tonal." In order to apply the classical background subtraction method, [3] and [10] construct mosaic image of the scene. Bevilacqua and Azzari [3] extract corner points on successive frames and match them. They try to eliminate inconsistent matches. Enough number of matches are evaluated and a model for camera egomotion is constructed. The model contains scaling, rotation, translation and perspective changes so as to construct a better model for camera egomotion. Finally successive frames are aligned with respect to

the camera egomotion and the mosaic image of the scene is obtained. Motion detection is performed by applying frame differencing between the related part of the mosaic image and the reference image. Figure 2-5 [3], is an example of mosaic image which is constructed by combining a number of consecutive frames.

Figure 2-5 An example mosaic image [3]

2.1.2.4 Motion Detection by Feature Matching

Feature based algorithms are one of the most promising type of solutions to active camera motion detection problem. Matching operation is carried out only on limited number of feature points. Thus these kinds of algorithms are generally superior to optical flow based algorithms in case of computational performance. However the choice of feature type is critical. The selected feature type should have a good performance in case of repeatability, robustness and computational efficiency. Also rotation, scale and affine invariant features probably achieve a better performance.

Harris and Hessian based detectors are former examples of feature detectors and they achieve lower performance with respect to the criterions which is mentioned above. SIFT [11], SUSAN [12] and SURF [13] are more recent algorithms and these algorithms generally achieve better performance. Tuytelaars and Mikolajczyk [14] compare algorithms for some performance criterions and create Table 2-1. They [14] claim that SURF is one of the best candidate feature trackers for real time applications. Juan and Gwun [15] also compare three feature trackers and claim that SURF is a good feature type in robustness and it is one of the best in the literature in case of computational performance.

Table 2-1 Performance comparison of features [14]

Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	Repeatability	Localization accuracy	Robustness	Efficiency
Harris	√			√			+++	+++	+++	++
Hessian		√		√			++	++	++	+
SUSAN	√			√			++	++	++	+++
Harris-Laplace	√	(√)		√	√		+++	+++	++	+
Hessian-Laplace	(√)	√		√	√		+++	+++	+++	+
DoG	(√)	√		√	√		++	++	++	++
SURF	(√)	√		√	√		++	++	++	+++
Harris-Affine	√	(√)		√	√	√	+++	+++	++	++
Hessian-Affine	(√)	√		√	√	√	+++	+++	+++	++
Salient Regions	(√)	√		√	√	(√)	+	+	++	+
Edge-based	√			√	√	√	+++	+++	+	+
MSER			√	√	√	√	+++	+++	++	+++
Intensity-based			√	√	√	√	++	++	++	++
Superpixels			√	√	(√)	(√)	+	+	+	+

Foresti and Micheloni [16] select features based on the eigenvalues of a 2 by 2 matrix which consists of partial derivatives on a window W on the image. The features on successive frames are matched and inconsistent matches are eliminated. Consistent matches are examined and translational camera egomotion is estimated. Reference frame is inversely mapped and frame differencing is applied. As an innovation, Shi and Tomasi [17] employ dissimilarity to eliminate wrongly matched features. The cross correlation in pixel intensities is calculated and dissimilar pairs are eliminated.

Zhou [18] employs SIFT features for object detection. The suitable matches between consecutive frames are determined and a validation process based on RANSAC [19] is applied on these matches in order to eliminate inconvenient pairs.

Suitable matches are evaluated with Affine Transformation Model. This model is able to identify background motions including scaling, rotation and translation. Affine Transformation model is formulated in (2-1) by [20].

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} dx \\ dy \end{bmatrix} + \begin{bmatrix} S \\ S \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2-1}$$

d_x and d_y represent the translational movement in the corresponding direction. θ represents rotation and S represents scaling on the image. Having the egomotion, Zhou [18] inverts the movement of the camera on the current frame and applies classical frame differencing. Finally some morphological operations are carried out so as to filter out the remaining noise and locate moving objects on the image.

2.2 Tracking

Tracking is one of the most popular subjects on image processing. Due to widespread usage on commercial and military applications, there are a wide variety of solutions for this task in the literature. On the other hand, for this thesis, a simple implementation of tracking will be sufficient because most of the challenges of this thesis are on the motion detection part. Thus the subject of tracking will not be explained deep in detail instead a brief summary of the subject will be mentioned.

Kim [2] defines the purpose of tracking as “The goal of object tracking is to find a moving object detected in motion detection stage from one frame to another in an image sequence” According to object representation method, Kim [2] classifies tracking into three categories which are point tracking, kernel tracking, and contour tracking. In Figure 2-6, taken from Yılmaz [4], some examples of object representations are demonstrated.

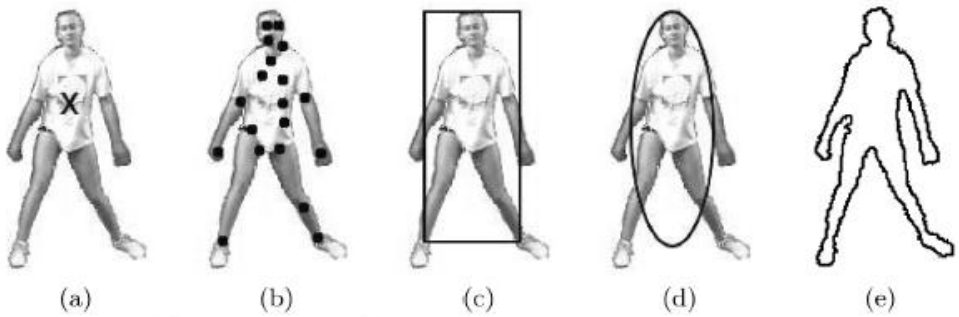


Figure 2-6 Different target object representations [4]

2.2.1 Point Tracking

In point tracking, target is represented by a point or a set of points detected on the image. A few examples are given on Figure 2-6 (a) and (b) [4]. Points on the target can be determined by one of the point detectors formerly mentioned in this thesis. Moreover the success of the tracker is mainly based on the chosen point detector and the detectors robustness performance on possible variations in an image [14].

2.2.2 Kernel Tracking

In kernel tracking, target is represented by a primitive geometrical shape like rectangular or ellipse such as given in Figure 2-6 (c) and (d) [4]. The motion of kernel which represents the target is generally modeled by translational or affine and route of the target can be calculated by this model. Kim [2] asserts that kernel based tracking is one of the most widely used method due to its performance and robustness.

2.2.3 Contour Tracking

In Contour Tracking, target is represented by an outline contour. Figure 2-6 (e) [4] is an example for Contour tracking. An initial contour is constructed from the first

image, and this initial contour is evolved between consecutive frames. Contour Tracking generally outperforms other methods for targets with complex shape changes. However, the success of the tracker is very bounded to the initial contour. Moreover according to [2], contour based trackers may fail on noisy, blurred and low contrast images.

CHAPTER 3

IMPLEMENTATION OF REAL TIME OBJECT DETECTION AND TRACKING ON ACTIVE CAMERA

As mentioned in chapter 2, there are four different kinds of motion detection algorithms for active cameras. Methods by camera geometrical properties are not chosen in order to avoid a hardware dependent solution. In this thesis it was aimed to use only image data to perform motion detection and tracking. Thus the proposed algorithm will be easily adaptable to all kinds of PTZ cameras. Optical flow based methods are not also implemented because real time performance is desired. Generally, Optical flow based methods are computationally more loaded thus they are unsuitable for real time applications.

In this thesis, two variants of feature based detection algorithms are implemented. The first variant can be defined as “Feature based egomotion estimation and frame differencing”. The second variant is a quite different approach to the current problem. On egomotion estimation step, a common approach is such that the outlier features are eliminated and motion is estimated from the remaining (inlier) features. Afterwards outlier features are not evaluated and simply disregarded [18], [16] and [17]. Contrarily, as well as inlier features, outlier features might contain valuable information. Thus the second algorithm tries to detect moving objects based on the outlier features on the image. Outlier based object detection is a known technique and has some examples like [21]. Pejcic [21] uses outlier blocks on the stable video to detect motion. The second variant algorithm can be defined as “Motion detection

and tracking based on outlier features". In this thesis, these two algorithms will be compared in case of detection and tracking performance.

Egomotion estimation step is identical for both variations of algorithms. In order to clarify the egomotion estimation step, the feature selection criteria and the selected feature SURF should be explained further.

3.1 Feature Type Selection

Feature type selection is one of the most important decisions for a feature based image processing algorithm. Surveys from Juan and Gwun [15] and Tuytelaars and Mikolajczyk [14] were examined for that purpose. Both surveys agree that SURF [13] is one of the most efficient and yet robust feature detectors in the literature. According to [15], SURF outperforms another well-known feature detector SIFT up to 1000 times with respect to computational time. Yet this is enough to choose SURF for a real time implementation. SURF features are scale and rotation invariant. Moreover they have a remarkable performance in case of repeatability and robustness. SURF is a relatively new method since it was proposed in 2006 yet SURF detectors were employed in lots of papers in the literature like [22], [23] and [24].

3.2 SURF: Speeded Up Robust Features

Speeded up robust features (SURF) is proposed by Bay [13] in 2006. SURF detects blob like structures and SURF features are translation, scale and rotation invariant. Fundamentally it relies on determinant of the Hessian Matrix. SURF feature detector is specially designed for computational performance. Thus some approximations and shortcuts are employed.

SURF interest points are found by calculating an interest point criteria $R(x, y)$ which is the blobness value of a pixel on the image. f is the blobness function

which takes intensity value of image pixels as input parameter. R can be formulated as follows

$$R(x,y) = f(I(x,y)) \tag{3.1}$$

For the sake of robustness to scale changes, any input image is considered as an image stack which is a collection of the input image in different scales. According to that approach, $I(x,y)$ becomes a 3D data which is $I(x,y,\sigma)$, which can be referred to as "image pyramid" in some cases. σ refers to scale parameter. Thus interest point criteria R becomes

$$R(x,y,\sigma) = f(I(x,y,\sigma)) \tag{3.2}$$

As mentioned earlier, interest point criteria R of SURF features are the determinant of the Hessian Matrix. Given image I, Hessian Matrix is defined as equation (3.3).

$$\mathrm{H}(x,y,\sigma) = \begin{bmatrix} L_{xx}(x,y,\sigma) & L_{xy}(x,y,\sigma) \\ L_{xy}(x,y,\sigma) & L_{yy}(x,y,\sigma) \end{bmatrix} \tag{3.3}$$

Here $L_{xx}(x,y,\sigma)$ refers to the convolution of second order derivative Gaussian $\frac{\partial^2}{\partial x^2} g(\sigma)$ with the image I in point (x,y). $L_{yy}(x,y,\sigma)$ and $L_{xy}(x,y,\sigma)$ have similar meanings such that only the direction of the second order derivation differs.

Interest point criteria R finally becomes as follows

$$R(x,y,\sigma) = \det(H(x,y,\sigma)) \tag{3.4}$$

Calculating second order derivatives for all pixels on an image is a time consuming process. Thus Bay [13] suggests an approximation for the second order Gaussian derivative kernels. Instead of a discretised Gaussian kernel, Bay [13] suggests proper box filter kernels. It is claimed that such an approximation for the kernels does not dramatically affect the performance of the algorithm but results a boost in the speed of the algorithm together with the implementation of integral image which is explained in the further chapters. Discretised Gaussian Kernels and related box

filters are given in Figure 3-1 [13]. On the left are the Gaussian kernels and on the right are the corresponding box filter kernels.

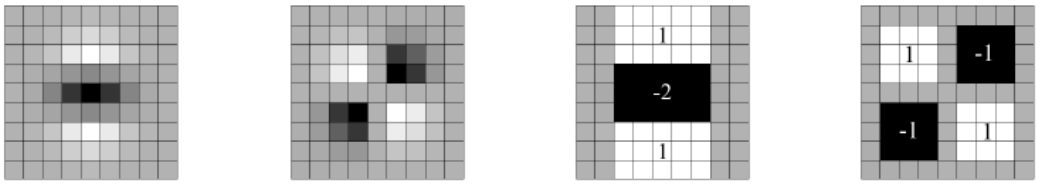


Figure 3-1 Exact and approximated Gaussian kernels [13]

In order to clarify the advantage of box filters more clearly, the subject of integral image should be explained further.

3.2.1 Integral Image Concept

Integral image $I_{\Sigma}(x,y)$ of an image $I(x,y)$ is defined as follows [13]:

$$I_{\Sigma}(x,y)=\sum_{i=0}^{i\le x}\sum_{j=0}^{j\le y} I(i,j) \tag{3.5}$$

In other words, the intensity value at any location (x,y) in the integral image $I_{\Sigma}(x,y)$, is the sum of all intensity values of all pixels inside the rectangular region with the top left corner $(0,0)$ and bottom right corner (x,y) on the original image I. Integral image (on the left) of the famous Lena image (on the right) is given in Figure 3-2.

Figure 3-2 Lena image and the corresponding (normalized) integral image

The most important property of integral image concept is the easiness of calculating the summation of the pixel intensities in a rectangular area on the image. Figure 3-3 is an example. Consider image $I(x,y)$ in the Figure. For normal operation, we should make $(B-D)x(A-B)$ number of summations in order to calculate the summation of the pixels in the region Σ. Assume the integral image $I_{\Sigma}(x,y)$ corresponding to the image $I(x,y)$;

$$I_{\Sigma}(A) = A1 + A2 + A3 + \Sigma \tag{3.6}$$

$$I_{\Sigma}(B) = A1 + A2 \tag{3.7}$$

$$I_{\Sigma}(C) = A1 + A3 \tag{3.8}$$

$$I_{\Sigma}(D) = A1 \tag{3.9}$$

(Desired formula) $$\Sigma = I_{\Sigma}(A) + I_{\Sigma}(D) - I_{\Sigma}(B) - I_{\Sigma}(C) \tag{3.10}$$

(Validation) $$\Sigma = (A1 + A2 + A3 + \Sigma) + A1 - (A1 + A2) - (A1 + A3) \tag{3.11}$$

$$\Sigma = \Sigma \tag{3.12}$$

Note that **Σ** can be calculated with only 3 summations (or subtractions) for integral image case by using the equation (3.10).

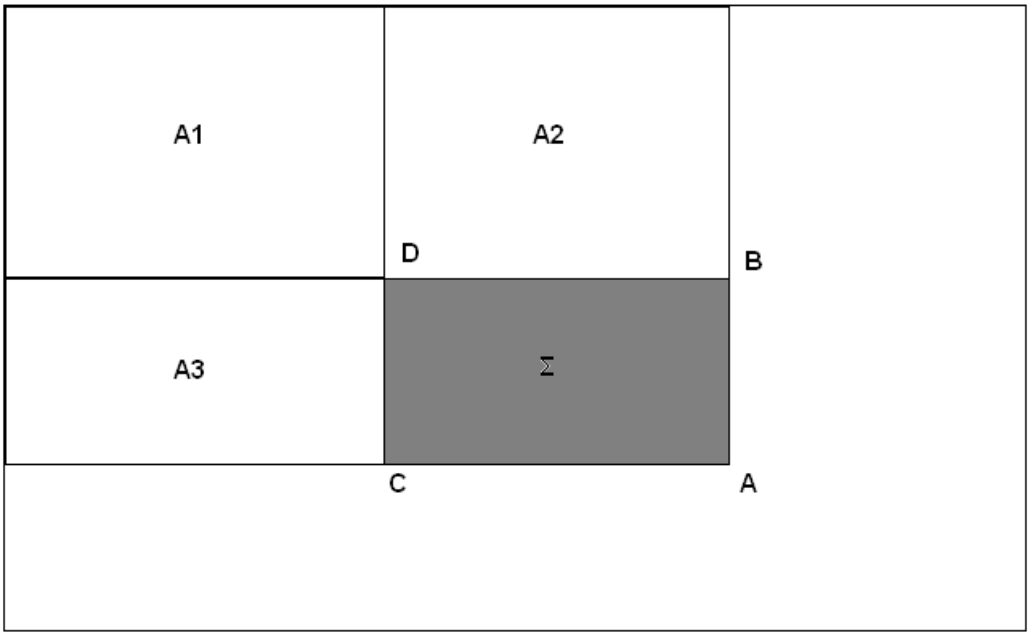


Figure 3-3 Box filtering example

3.2.2 Approximated Hessian Matrix Determinant Calculation

Let's reconsider to the smallest kernel (9x9) of box filter in Figure 3-4. For normal convolution operation, 81 multiplication and 80 addition operations are needed. For convolution operation with integral image concept, just 9 addition operations are enough.

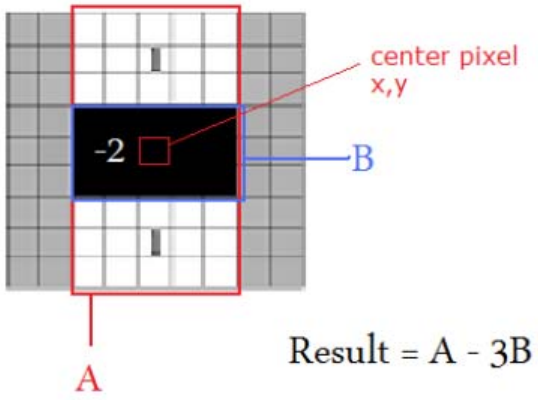


Figure 3-4 Smallest kernel for box filtering

The determinant of the approximated Hessian matrix is defined in (3.13). Note the constant multiplier 0.9. It is necessary to normalize the error caused by the approximation. L_{xx}, L_{yy} and L_{xy} are the approximations on the related direction.

$$\det(H_{approx}) = L_{xx}L_{yy} - (0.9L_{xy})^2 \tag{3.13}$$

Another advantage of box filtering is the fact that computation time is identical for all kernel sizes. On the other hand, for normal convolution, computation time increases proportional to the square of the filter dimension. Thus, any kernel size filters can easily be applied on Integral images. While calculating SURF features, Hessian determinant is applied with different size kernels (9x9, 15x15, 21x21, 27x27). Each kernel size represents a layer or a scale on the image pyramid.

3.2.3 Non-maxima Suppression and Interpolation

Approximated hessian determinant values are thresholded through the image in all scales and candidate interest features are found. Final step to obtain SURF features is “Non-maxima Suppression”. A blob on image may give blobness response on more than one scale or more than one point on the coordinate plane. It is obvious that an elimination step is necessary. A candidate point is chosen as SURF feature if

its blobness response is greater than its entire 3x3x3 neighborhood in x, y, σ dimensions. Figure 3-5 [11] consists the visualization of this phenomenon.

SURF features can be localized in sub pixel resolutions in x, y and σ domain. By interpolating neighboring points of a feature, a continuous interest point criteria plane is constructed. Local maxima on these plane corresponds to exact sub pixel resolution coordinates of selected SURF feature.

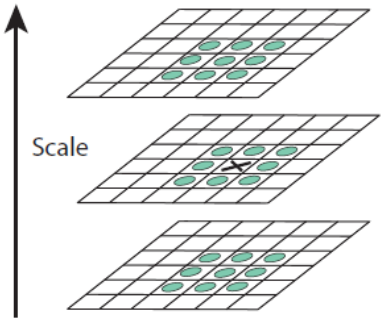


Figure 3-5 Non-maxima suppression for candidate SURF features [11]

3.2.4 SURF Descriptors

So far, SURF interest points on an image are founded. A descriptor calculation step for each features are necessary. Descriptors are used for the inter frame feature matching step. They are like IDs for SURF features and any two SURF features can be matched if their descriptors are similar in case of some measuring criterion. Haar wavelets are employed during descriptor calculation step. Haar wavelets in Figure 3-6 [13] are simple filters for gradient calculations but they are computationally very efficient due to the integral image concept. SURF descriptor calculation consists of two steps which are orientation assignment and calculation of descriptor components.

Figure 3-6 Haar Wavelets [13]

In the first part a repeatable and robust orientation is assigned for each SURF feature. This concept also ensures the rotational invariance of SURF features. The descriptors are calculated based on this orientation. In an area of radius 6σ, Haar wavelets of size 4σ are calculated. Here σ refers to the scale at which the current interest point was detected. Since the SURF features can be extracted in any allowed size, the descriptors should also be calculated in that specific size. Then, the calculated wavelet responses are weighted with a Gaussian which is centered at the location of interest point and of size 2σ. The Haar Wavelet responses are positioned in the X-Y plane and a dominant direction is chosen on the plane. The dominant direction forms the orientation of the related feature. Figure 3-7 which is taken from the original SURF [13] paper, demonstrates the orientation assignment step.

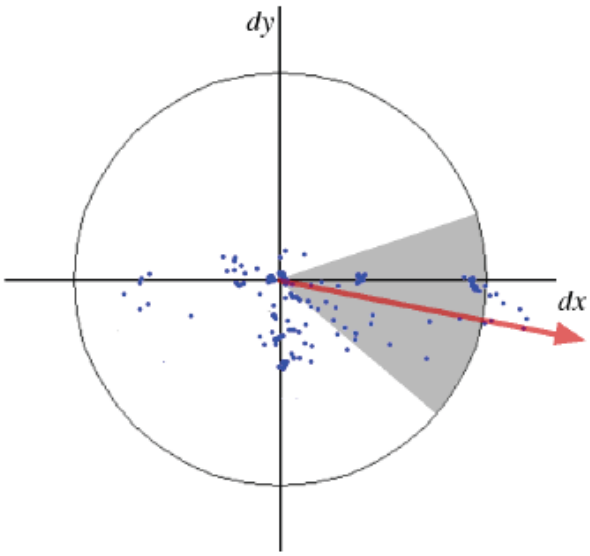


Figure 3-7 Orientation assignment for SURF features [13]

The second step consists of the calculation of the descriptor components. Based on the orientation which is calculated before, a square region of size 20σ is allocated. The square area is divided into equal sized 16 sub regions. Then these sub regions are also divided into 5x5 sub regions and Haar wavelet responses are calculated. For each 16 sub regions, 4 descriptor values are calculated. Two of the descriptor values are the summation of Haar wavelet responses in the direction of x and y, and the remaining two descriptors are the summation of the absolute value of Haar wavelet responses in the direction of x and y.

Finally by applying the above procedure, for each SURF feature, a descriptor array of size 64 (16 x 4) is constructed. Similarity of any two features can be determined by calculating the Euclidean distance between their descriptors. Figure 3-8 [25] visualizes the descriptor concept as well as the descriptor formulation.

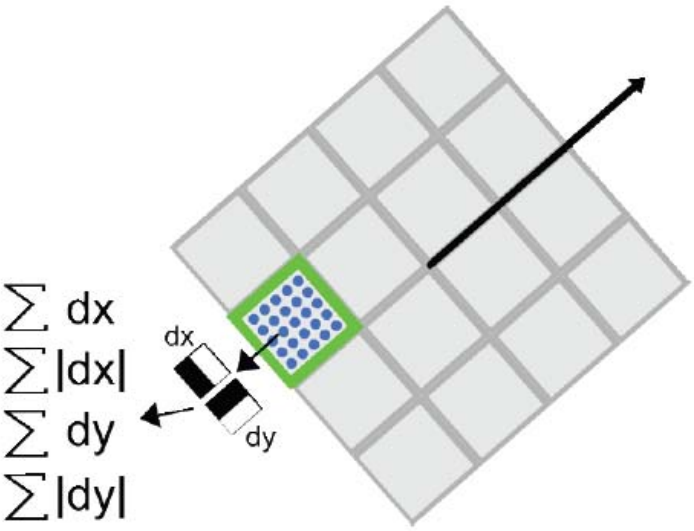


Figure 3-8 SURF descriptor calculation [25]

3.3 Egomotion Estimation

Egomotion estimation is a common step for most of the active camera motion detection algorithms. [3], [10], [5], [16], [7], [8], [9] and [18], all initially estimates the camera egomotion before attempting to object detection. Background is not stable though moving objects cannot be recognized by simple methods like frame differencing. Objects can be recognized by moving pixels whose direction is distinct from the remaining portions of the video frame. That is the main reason why egomotion estimation step is necessary.

One basic assumption should be taken such that the moving object consists only a small portion of the video and the background covers most of the portions of the video frame. This assumption is essential if there is not any priory information about the existence and the position of the objects. In case of feature based

algorithms, this assumption evolves such that most of the features are on the background and only a small portion of the features are on moving objects.

In this thesis, egomotion is found by using only the input video frames and using internal camera parameters like pan and tilt information is avoided. SURF features are employed in order to estimate the camera motion. In most cases feature based egomotion estimation is computationally more efficient compared to Optical flow based methods. The main reason is Optical flow based methods works on pixel domain and some loaded calculations are applied to all pixels on the image. On the other hand, egomotion can be calculated by examining a few hundred features. In the proposed algorithm 100 SURF features are usually enough to estimate the egomotion. The critical point here is that the feature extraction step itself should not consume too much processing time. That is the main reason why SURF features are chosen.

In this thesis egomotion is modeled with three different ways which are translational RANSAC, translational K-MEANS and linear RANSAC. Affine transformations are able to handle translation, rotation and zooming actions of the image however due to the hardware used in this thesis rotation movement is not expected so affine model is not chosen. Thus only translational and linear models are employed in this thesis. Estimated egomotion is accepted only if more than 30 percent of the features strictly agree on the same motion model. Based on taken experiments, this ratio is often sufficient for robust motion estimation. Flow diagram of the motion estimation algorithm is given in Figure 3-9.

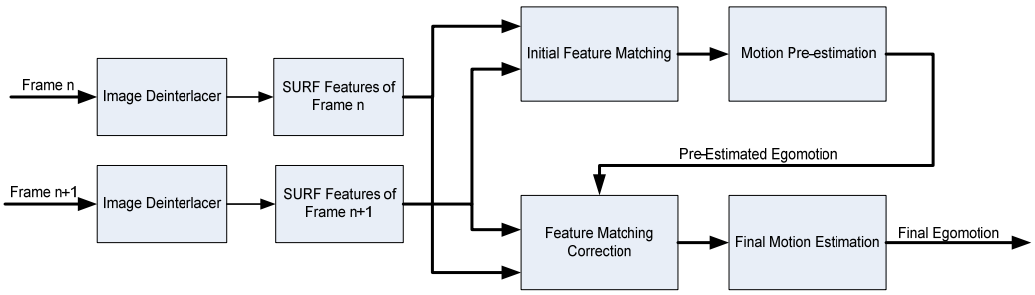


Figure 3-9 Flow diagram of motion estimation

3.3.1 Feature Extraction and Initial Feature Matching

Initially, input images should be deinterlaced. The camera used in this thesis produces PAL video. Videos in PAL standard are interlaced such that sequentially at each frame, only the odd lines or even lines are sent to the receiver. Human eye usually cannot catch that event and interlacing effect is sensed as doubling the frame rate. However interlacing has an unwanted effect for active cameras. When the camera is moving, odd and even fields of a frame are snapped at different time instants. Especially at lower frame rates like 5 FPS, this causes deviation and blurring on the image. The simplest solution for this problem is deleting the even lines and copying the odd lines on to the even lines. Although resolution of the image is reduced, this does not have an important effect on the SURF feature extractor. Additionally this solution is computationally efficient. Figure 3-10 demonstrates the effect of the deinterlacer algorithm. Frame (a) is taken from interlaced video while the camera is moving and Frame (b) is the corresponding deinterlaced video part. Note the general blurring on the left image. The effect of interlacing is obvious by observing the pole on the left.

(a) (b)

Figure 3-10 The effect of deinterlacing

SURF features of frame n and frame n+1 are found as shown in

Figure 3-9. In order to gain processing time, SURF descriptors may not be calculated as applied by Nguyen [22]. Only the feature orientations are calculated in the proposed algorithm. Then between frame n and frame n+1 feature matching is performed for the first time. It is trivial to assume that blob type (white or black blob) cannot be changed between consecutive video frames. Moreover, by the fundamental assumption of tracking, features can only make small motions between successive frames. Thus a pair of SURF features is matched only if their type, size, location and orientation values are close to each other up to some predefined thresholds. Pseudo code for feature matching is given in Figure 3-11. The order of condition checks are specially designed such that computationally more loaded checks, location and orientation are handled at the end of the nested conditions block.

- For all the features of frame n+1
  - Search through all the features of frame n

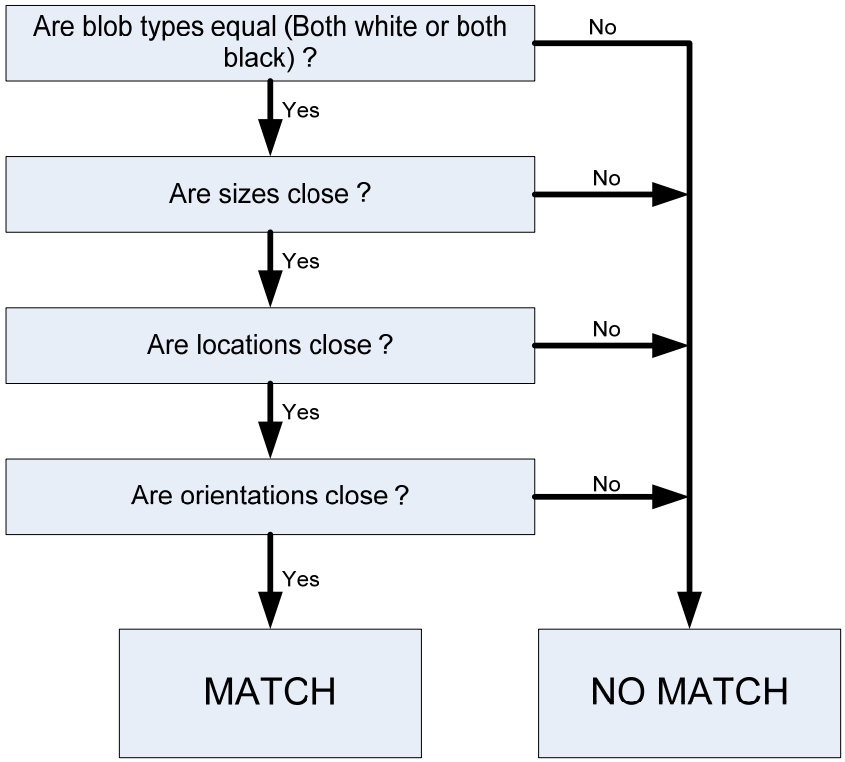


Figure 3-11 Pseudo code for feature matching procedure

3.3.2 Egomotion Pre-estimation

After the matching step, an array of motion vectors is obtained. It is assumed that most of these motion vectors belong to the background so they each contain the egomotion information. However there are some outlier features inside the motion vector array. Outlier features can be on a moving object or they might be erroneously matched pairs. A process is necessary to filter out these outlier features

and to estimate the global motion of the camera. Egomotion estimation is implemented with three different ways in this thesis work.

3.3.2.1 Translational RANSAC Based Modeling

Ransac [19] is the abbreviation of "Random Sample Concencus". It is a probabilistic and iterative method to estimate parameters of a mathematical model from a set of observed data which contains some misleading and erroneous data samples. It is widely used in computer vision for motion estimation applications [22], [23], [18]. Assuming that inliers in the data set are larger in number, RANSAC chooses a small number of samples randomly and assume that they are inliers (correct). Then a model is constructed with these samples. Constructed model is simply the average vector of the chosen inlier features. The whole set is reexamined with this model and inlier set is updated. Then estimated model is recalculated with the updated inlier set. The algorithm iteratively continues until a large number of samples fit to the constructed model. Otherwise if sufficient number of inliers cannot be obtained, the whole process is repeated until a valid model is found or an iteration limit is reached.

In the developed algorithm, initially 5 features are selected as inliers and an average translational movement is calculated based on the selected inlier set. Then the algorithm is iterated to enlarge the inlier set as described before. A model is accepted whenever 30 percent of the features fit to the estimated model. The same method is repeatedly performed until a model is obtained or the iteration count reaches to 20.

3.3.2.2 Translational K-means Based Modeling

K-means is an iterative clustering method frequently used in computer vision for segmentation purposes. The algorithm aims to divide the input data set to K distinct clusters [26]. For egomotion estimation, a single, intense cluster consists of at least a predefined number of samples are searched. After matching step, each motion

vectors of the matched features are located in the X-Y plane. Initial guess “P” is the average of the motion vectors. Motion vectors inside a circle of radius R and center P are selected as potentially inlier features and a new average is calculated with these features. The algorithm iteratively continues until convergence. R value is decreased in each iteration thus at the end of the iterations, an intense point in the vector plane might be reached.

Figure 3-12 demonstrates the procedure. Red dots are the input samples. Blue dot 1 is the initial center of mass (average) of the samples. A circle or radius R is positioned and a new center is obtained by using the points inside the initial circle. The algorithm iteratively runs and finally it converges at point 5.

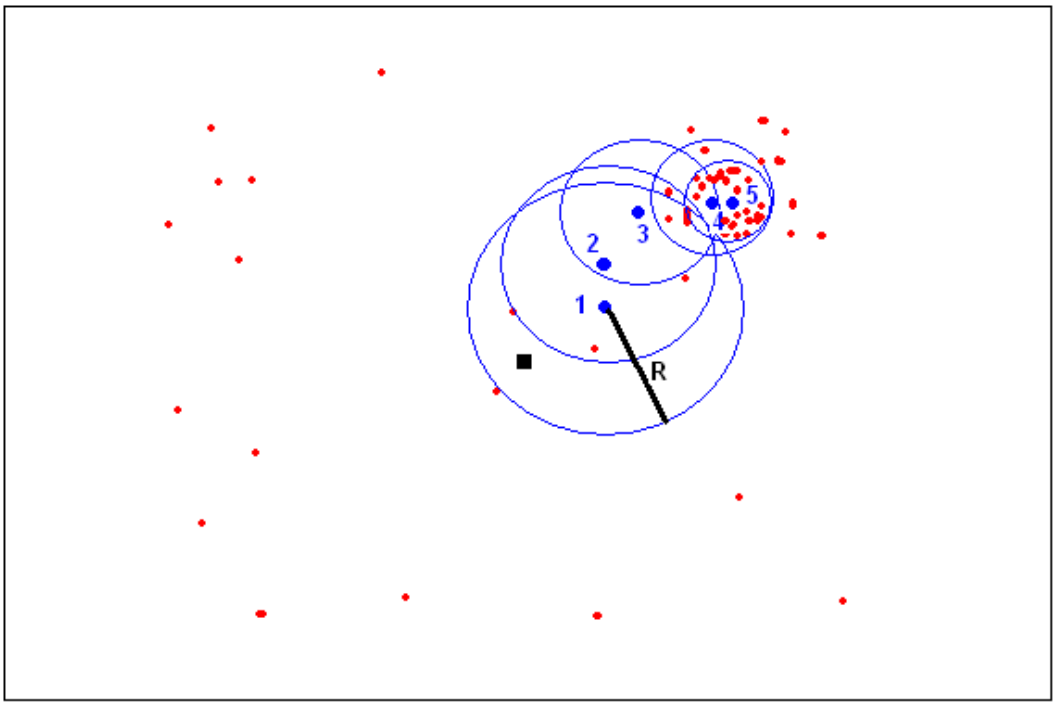


Figure 3-12 K-means based egomotion estimation

K-means based egomotion estimation has a considerable advantage such that it is not probabilistic. Ransac relies on an initial subset which is assumed as inlier. If

initial set contains corrupted samples then Ransac fails. However K-means based egomotion estimation does not need an initial randomly chosen subset instead it needs an initial starting point for search. This initial starting point is given as the average value of all the sample vectors which is also a deterministic value. Thus the algorithm may fail only if the ratio of outlier features rises to a very high level in which Ransac has already failed. Such high outlier ratios are not frequently encountered if the initial assumption is valid. High ratios might be encountered only if a moving object of very large size appears on the screen. However this is a contradiction to the main egomotion assumption. Nevertheless, a precaution is taken for that kind of situations in the proposed algorithm. While calculating egomotion on frame n+1, the features on an area which belongs to a formerly detected object, is not counted thus some of known outlier features are eliminated before the egomotion modeling algorithm. It is observed that this precaution causes a distinctive improvement on the robustness of egomotion estimation algorithms.

3.3.2.3 Linear RANSAC Based Modeling

Former two models assume only translational movement. However, the camera used in this thesis has zoom capability and this function can be beneficial when the target object is too far or too close. A linear transformation model is implemented thus the egomotion of the camera can be modeled when the camera is zooming as well as translating. Deterministic K-means based modeling can not be used with linear model because movement vectors cannot be located on a 2D plane when scale changes are also possible. Thus only RANSAC is used for this case.

Assume x and y are the initial coordinates and x’ and y’ are the corresponding points after the transformation. dx and dy are the translation and S is the scale parameters of the linear mapping. Transformation formula is given in (3.14);

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} dx \\ dy \end{bmatrix} + S \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.14}$$

Note that there are three unknowns and so two points (thus four equations) are enough to model the egomotion. Initially two matched points are chosen as inliers and four equations are obtained. Three of the equations are used to calculate the model and the fourth equation is used to verify the estimated model. Then the initially verified model is applied to all the features. The model is accepted if 30 percent of all the features obey the model. The whole process is repeated if the model is not verified or an iteration limit is reached. Figure 3-13 demonstrates an example instant, where egomotion is estimated based on the linear model. The camera is translating left down and zooming out at that instant.

Figure 3-13 Linear RANSAC based egomotion estimation.

3.3.3 Feature Matching Correction and Final Motion Estimation

In order to improve the performance of the egomotion estimation step, feature matching and motion estimation parts are repeated once more but with some slight changes. In the first matching step, candidate features are searched inside a larger circular area without any priory knowledge about the motion of the entire video. However after the initial motion estimation step, an initial guess for camera egomotion is obtained. Thus in the matching correction step, candidate feature search process is repeated such that, new candidates are searched inside a smaller area based on the estimated motion of the camera. Formerly matched pairs are not updated unless a new match is found inside the new smaller area. This algorithm is demonstrated in Figure 3-14.

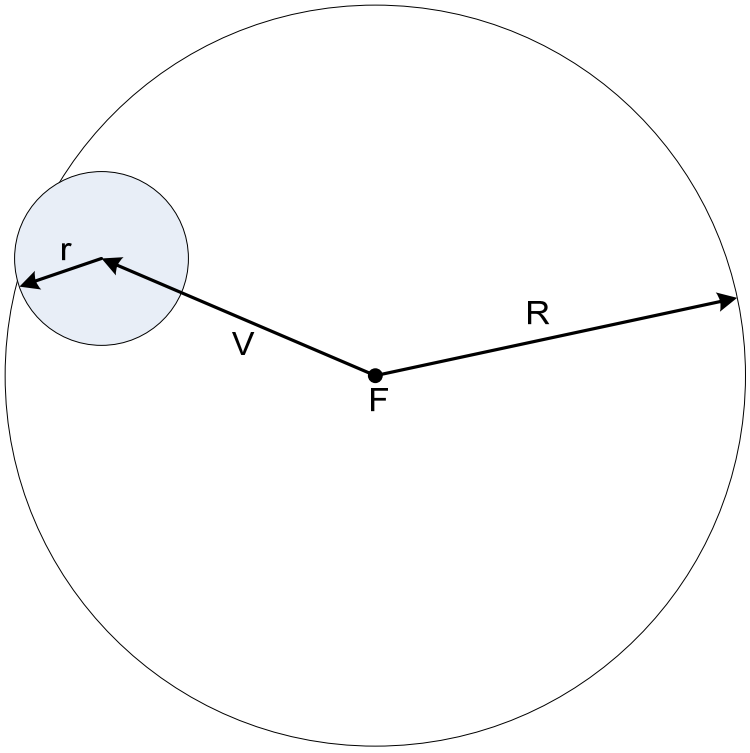


Figure 3-14 Feature matching correction

For feature F, a candidate SURF feature is searched through a circular area of radius R in the first matching step. Assumed that egomotion is estimated as vector V in the first motion estimation step. Thus a better candidate is searched through the smaller circular area of radius r in the match correction step. If a candidate is found, the match for feature F is updated and F becomes an inlier.

Feature matching correction step, eliminates some of the wrongly matched pairs. Thus egomotion estimation algorithm is repeated and the final egomotion is calculated. Repetition of the algorithm is useful in cases where the initial estimation algorithm fails to converge due to high ratio of wrong matches.

3.4 Moving Object Detection

Moving object detection is the next step of the main algorithm. As mentioned earlier, two variants of object detection algorithm are implemented and compared in this thesis.

3.4.1 Feature Based Egomotion Estimation and Frame Differencing

This algorithm mainly inspires from the classical detection approach used with static cameras. Background modeling and frame differencing is a promising way for static camera case. Similarly the same method is applied with active cameras however with an important modification. Camera motion is calculated first and the current frame is inversely shifted. Then background subtraction is applied. Reference papers [3], [10], [5], [16], [7], [8], [9] all rely on the same principle with some variations on other parts of the algorithms. The same idea is implemented in this thesis. The flow diagram of the algorithm is given in Figure 3-15.

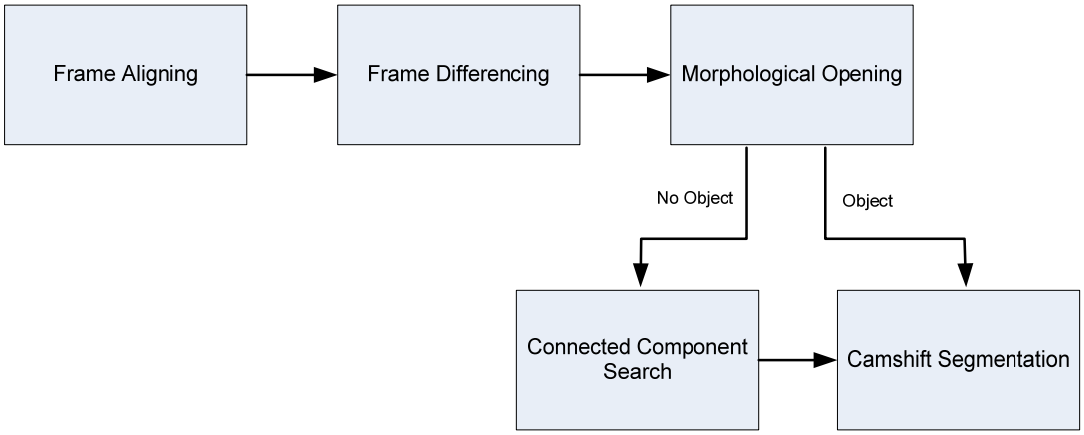


Figure 3-15 Flow diagram for moving object detection by frame differencing

3.4.1.1 Frame Aligning and Frame Differencing

Relative shift due to the camera motion is known thus the current frame n+1 is inversely shifted according to the estimated egomotion. Consecutive frames are aligned now and frame differencing can be applied. Indeed an exact image shifting operation is not implemented. Instead a function called “subtractImage” is designed such that it takes egomotion as one of its inputs and applies pixel by pixel alignment and subtraction. After subtraction, pixels lower then a threshold are discarded and other nonzero pixels are kept thus a binary difference image is obtained.

3.4.1.2 Morphological Opening

Even with a perfect egomotion model, frame differencing may result some corruptions on the difference image. Especially on the edges, thin lines may occur as seen in Figure 3-17 (d). Morphological operations are well suited in order to overcome such kind of malfunctions. A square kernel of size 2 is used and morphological opening operation is applied in the developed algorithm.

3.4.1.3 Connected Component Search

Assuming that the binary difference image consists only the moving objects, a group of nonzero pixels should correspond to a moving object. Figure 3-17 (e) is a good example. A connected component labeling operation is necessary to separate and label the location which consists of nonzero pixels. In order to determine the location, a segmentation algorithm called Camshift [27] "Continuously Adaptive Mean Shift" is used. Camshift algorithm can locate the nonzero pixels and determine the size of the segment. However the algorithm needs an initial search location. Connected Component Search procedure is necessary to find this initial search location which is necessary for Camshift.

Connected Component Search procedure is demonstrated in Figure 3-16. The image is divided into squares of size 80x80 pixels. The segments on the edges which are shown with gray on Figure 3-15 are omitted. Four lines and totally 24 segments are obtained inside the image. By considering real time working requirement, at each frame only one line which consists of six segments are analyzed. Nonzero pixels on each segment are counted and if there are more than a defined threshold, then a moving object is assumed at that location. The segment area is used as the initial search location for Camshift tracker. Note that this process is necessary only at first detection of the object. After the first detection, search location can be obtained from the track result of Camshift and the preceding speed of the object.

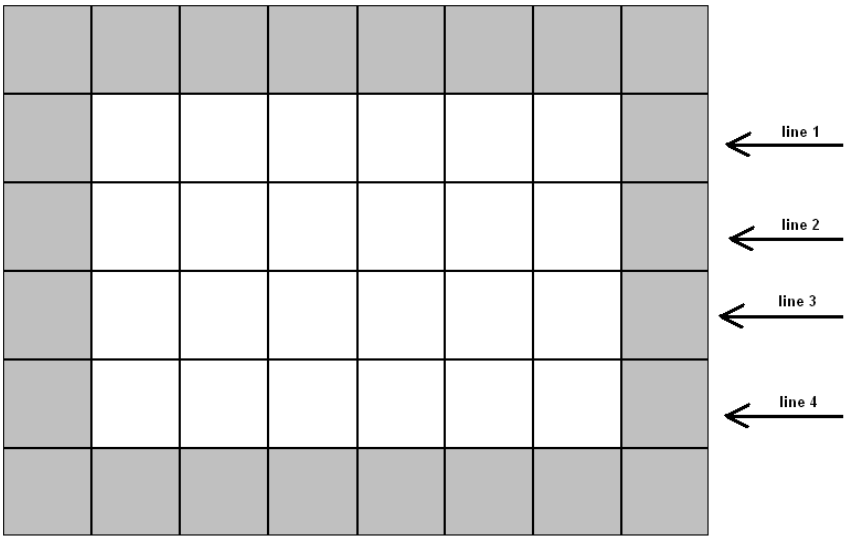


Figure 3-16 Subdivisions of the screen during moving object detection.

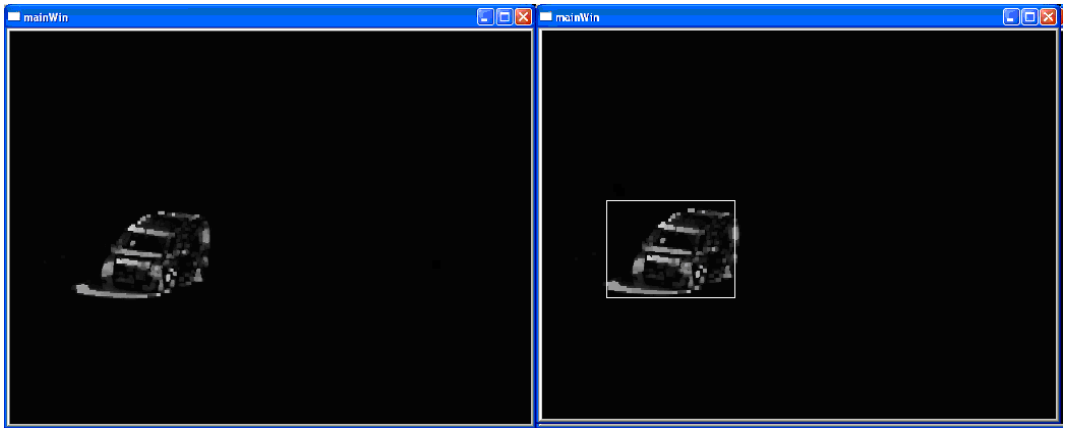
3.4.1.4 Camshift Tracker

Camshift [27] algorithm is based on Meanshift [28] algorithm followed by target size and orientation estimation. Meanshift algorithm is mainly used for segmentation purposes. It is an iterative color based procedure aims to locate the maxima of a density function by using the discrete data sampled from that function [29]. Camshift is a well known algorithm which is used for tracking purposes as well as image segmentation. In the literature there are a lot of successful examples in which Camshift is used for object tracking [30], [31] and [32]. Moreover a proper implementation of the algorithm is available in OpenCV library.

In the developed algorithm, Camshift works on binary image. As demonstrated in Figure 3-17 (e), the moving object forms a white segment surrounded by black pixels. In such a frame, Camshift successfully estimates the location and size of the object as seen on Figure 3-17 (f). At each frame, the object location, size and speed is updated and the initial search location for the next frame is estimated.

(a) (b)

(c) (d)



(e) (f)

Figure 3-17 Implementation steps of moving object detection

Intermediate steps of the algorithm are demonstrated in Figure 3-17. Frame (a) and frame (b) are consecutive frames. Camera makes a small motion through left and down. Egomotion is estimated by K-means. In order to clarify the idea, the current frame is shifted through the opposite direction and frame (c) is constructed. Frame differencing is applied and frame (d) is obtained. Note the thin lines on the edges. This is due to motion estimation errors and digitization errors of video input devices. These unwanted thin lines and other small sized errors are filtered out by morphological opening operation. A kernel of block size 2 is used thus lines thinner than 4 pixels are deleted. Final moving object is demonstrated in frame (e) and it is located in frame (f).

3.4.2 Motion Detection and Tracking Based on Outlier Features

The first implemented method estimates the egomotion of the camera based on SURF features. Then moving object detection step is carried out based on image pixels. Instead of working on image pixels, already calculated SURF features can be employed to detect moving objects. This idea seems to be superior in terms of computational time for two reasons. The first reason is that operations on image pixels are generally time consuming because any process should be repeated for approximately 300000 times (640 x 480 = 307200). The second reason is that necessary SURF features are already calculated for egomotion estimation phase and there is no need to consume effort for recalculation. The flow diagram of the proposed algorithm is given in Figure 3-18.

Figure 3-18 The flow diagram of outliers based object detection

The main motivation of the algorithm is as follows. It is assumed that a group of inconsistent features correspond to a moving object if their positions on the image are close and their motion vectors have approximately the same directions. Once an object is detected, object location and speed are recorded and the same object is searched through an approximate location based on the former location and speed of the object.

3.4.2.1 Outlier Feature Detection

Algorithm starts with detection of the outlier features. All matched SURF features are compared with the camera egomotion regardless of the location of the features. Features whose motion vectors deviate from the egomotion, are counted as outlier and they are added to a vector which consists of all the outlier SURF features of the current frame. The subsequent parts of the algorithm works on this outlier vector array.

Figure 3-19 demonstrates the procedure. The image is the 147. frame of the input video. 367 SURF features are found and 317 of them are matched with features of the former frame. White line or dots demonstrates the motion of an inlier feature at the exact location. It is seen that there are no white lines but there exists white dots and that means that inlier features are steady. Thus the camera is steady at the moment.

Figure 3-19 Inlier and outlier features on frame 147 of video1

There are 18 outlier features. Blue circles correspond to black blobs and red circles correspond to white blobs. Black lines inside the circle show the motion of the feature. The center of the circle shows the current location of the feature and the other end of the black line shows the old location of the feature. Note that the motion of SURF features on the moving car is generally parallel. There are a few wrongly matched outliers on the fences and on some other locations of the image but they do not ensure the object detection principle thus they are simply ignored. However the features on the car are close to each other and their motion is parallel thus the car can be detected as a moving object by the main object detection principle.

3.4.2.2 New Object Search

Chronologically new object search is handled after pre-defined object update. However in order to clarify the subject, firstly new object search procedure will be explained. A group of adjacent outliers might refer to an object on the image thus special attention is taken on adjacent outlier groups. The flow diagram of the procedure is given in Figure 3-20.

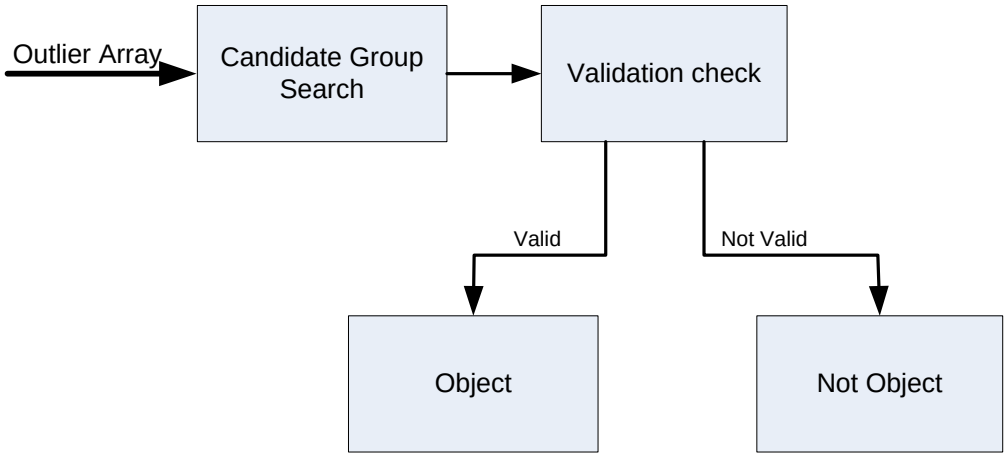


Figure 3-20 Flow diagram of new object search

The algorithm starts with the candidate group search. For all outlier features, adjacent neighbor features are determined and close features are gathered in a group. A group should contain at least a predefined number of features. This limit is set to three in this thesis.

A validation operation is needed to classify real moving objects and false detections. This validation operation decides whether a group of outlier feature corresponds to an object or not. Validation mechanism consists of two rules. In the first rule, existence of at least one feature is demanded. In the second rule, parallel motion is desired. Object is validated if there exists at least three outlier features whose last motions are consistent with each other. It is assumed that if both rules

are ensured, then features correspond to a moving object and the object motion vector is determined as the consistent motion of the three outliers. Candidate groups which are not validated are simply ignored.

Two example images are given in order to clarify the subject. In Figure 3-21, some incorrect matches are found on the building and a candidate group is constructed with these outlier features. However this candidate group violates rule 2 such that there is not a common motion vector for at least three features. All features have distinct motion directions thus it is concluded that this candidate group does not correspond to a moving object. In Figure 3-22, a group of outlier features are detected on the moving van. Note that most of the features indicate the same direction and both two rules are ensured. Thus the candidate group of outliers is validated as an object.

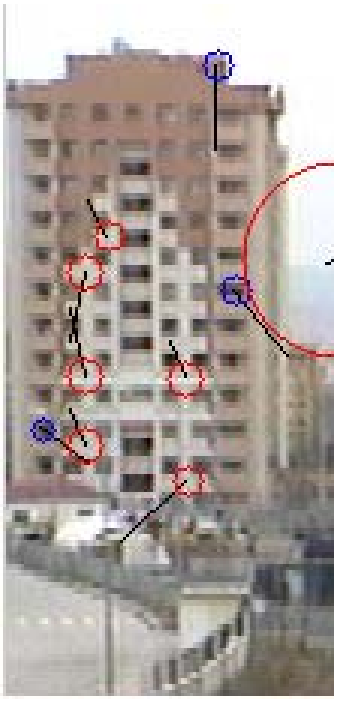


Figure 3-21 Outlier features due to wrong matches

Figure 3-22 Outlier features on a moving vehicle

3.4.2.3 Pre-Defined Object Update

Once a moving object is detected in the image, its location and motion information is updated at each frame of the video. Kalman filter is employed for group object update process in order not to loose the object with momentary variations. Group update procedure consists of the steps demonstrated in Figure 3-23.

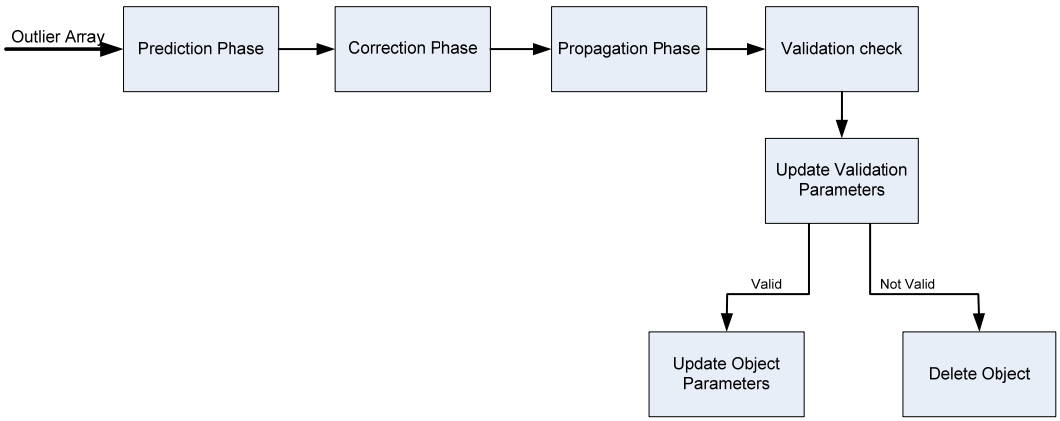


Figure 3-23 Object update procedure

The algorithm starts with Kalman Prediction phase. Object location in the current frame are predicted based on the last validated location and the motion of the object. Next, outlier SURF features are searched inside the predicted object location and found features are added to vector array for further observation. Based on the determined features, a new object location is obtained thus the predicted object location is corrected. This step corresponds to correction phase of Kalman filter

Note that if the size of the object decreases, it can be realized in the correction phase. However an increment on the size of the object cannot be noticed with the above procedure. In order to detect object size increment, a similar propagation phase is added in group update step. Close features are added to the group and object size is updated.

Some constraints are added to the propagation phase to increase the robustness of the algorithm. Object size cannot change rapidly. Similarly object cannot jump directly to a new location far away from the object initial location. Final constraint is that the object speed can not change rapidly but expected to change with a limited acceleration. These constraints limit the search area and increase the robustness of the algorithm.

Validation step is exactly the same. At the end of this step two validation parameters are found based on rule 1 and rule 2. However, when an object is not validated, the object is not directly deleted. Instead consecutive none valid cases are counted. Different limits are determined for the two rules. As a result of trial and errors, limits are arranged such that an object is deleted if rule 1 is not validated for consecutive 3 frames or rule 2 is not validated for consecutive 6 frames. Whenever a group of outliers is validated, the object location and object speed parameters are updated and non-valid counters are set to zero.

As mentioned, while using a translational egomotion estimation technique, scale changes can not be modeled. Thus the algorithm cannot estimate egomotion while the camera is zooming. When egomotion cannot be estimated, the outlier features are not reliable thus new objects cannot be detected. However a formerly detected

object can still be followed because features on the object remain even at the zoom instants due to the fact that SURF features are tolerant to scale changes. Figure 3-24 demonstrates the incident. Figure 3-24 (a), (b), (c) and (d) are taken at different time instances while the camera is zooming out. The car is detected formerly and the detector does not loose it during the zoom out period. Besides, note that any new object is not detected even though lots of outliers exist in the images.

(a) (b)

(c) (d)

Figure 3-24 Target detection while the camera is zooming out. Note the unreliable outlier features

For comparison, the same frames are evaluated with linear egomotion model in Figure 3-25. The camera motion is estimated as pure zooming out. Note that outlier features are very limited in this case compared to the translational model. Outlier features are still reliable in this model so new object search process is applicable at the moment.

(a) (b)

(c) (d)

Figure 3-25 Target detection while the camera is zooming out. Note that outlier features are still reliable at the moment.

CHAPTER 4

IMPLEMENTATION RESULTS AND COMPARISONS

The main aim of this thesis is to construct real time object detection and tracking system with an active camera. For that purpose, EVI D100P [33] which is a PTZ camera produced by SONY, is used. The camera produces PAL analog video output. It has a serial port thus pan, tilt and zoom parameters can be adjusted. The camera is operated at 640*480 resolutions. Analog PAL video is taken to the computer by DIGITUS DA-VC211 video grabber. The computer that is used for this thesis is a standard laptop which has Core2 Duo 2.4 GHz CPU and 3 GB RAM.

Microsoft Visual Studio 2005 is selected as the development environment and C++ language is used. OpenCV library (OPENCV 2.0) is frequently employed. There is a SURF implementation in OpenCV however an open source library called OpenSURF [25] is used instead, due to its superior performance.

Figure 4-1 SONY EVID100P PTZ camera

In order to develop the algorithms and make comparison, a small dataset containing outdoor surveillance videos is constructed. The surveillance videos include pan, tilt and zoom action of the camera. Day and night videos are included and single or multiple moving objects are occurring in the videos.

Before starting to discuss the results, the symbology on the videos will be explained. Figure 4-2 is a snapshot including the symbology. On the top left, FPS (frame per second) information of the software is written. On the top line of the frame, the number 95 is the frame number of the input video. 312 and 195 are respectively the total number of SURF features on frame 95 and the total number of matched features. In the second line, the first number is the ratio of the frames in which the egomotion can be successfully estimated. The character “D” indicates that the object detection algorithm is active and the character “T” indicates that the tracking algorithm is active so that the camera is automatically directed to the target object by the software. The white square over the car refers to a detected moving object. Final symbology is the egomotion indicator located in the middle of the image. The small filled green circle indicates that a motion model is successfully found by egomotion estimator. If this circle is red, that means camera motion cannot be modeled in this frame. The black circle indicates that the camera is zooming out. The radius of the black circle is proportional to zooming speed of the camera. The circle is white if the camera is zooming in and simply there is not any such circle if the camera does not zoom. Finally any translational movement is represented by a line starting from the center of the image. If there is a small black dot instead of a line then that means the camera is stable and simply it is not making any translational movement.

Figure 4-2 Symbology on images

The implementation results can be examined under three chapters which are egomotion estimation results, motion detection results and tracking results. During the experiments 6 videos from the dataset which are video1, video5, video11, video19, video24 and video25 are used. In the videos, the camera is sometimes stable, sometimes rotating and sometimes zooming.

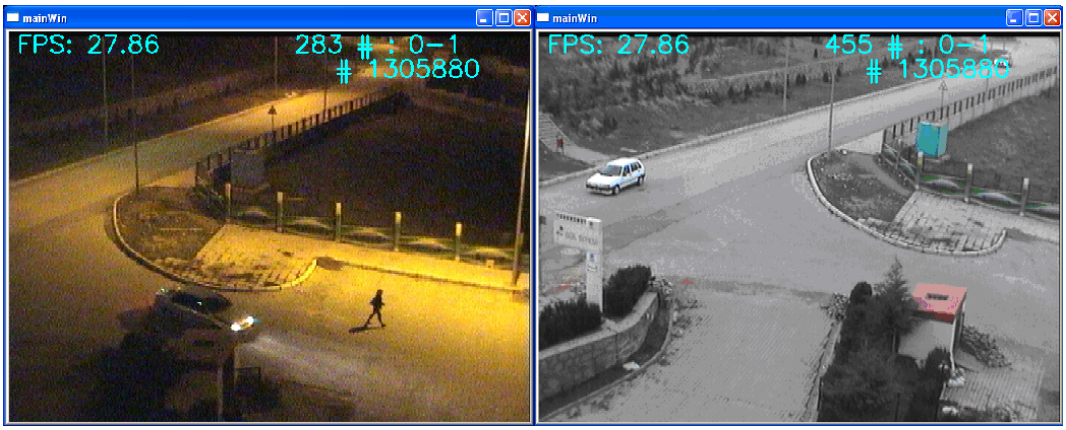
Video1 is a day video and consists of both translational and zooming variations of the camera. A mobile car is seen on the video. This video is used to test the egomotion estimation algorithms. Video5 is a day video and consist of a walking pedestrian. Pedestrian is a harder target for a tracker since its size is smaller compared to a car and its shape is continuously changing unlike a car. Moreover, feature number decreases at some instants in this video. Video9 is a night video. At

first a car appears and then a walking pedestrian passes. Video19 is a day video and consists of only translational variations of the camera. Cars are moving on the way. It is a relatively easy video for target detection. Video24 and video25 are evaluated for measuring the performance of the egomotion estimation algorithms. There is not any significant moving object inside these videos. One example frame from each video is given in Figure 4-3. Table 4-1 includes the total frame numbers of the video, the number of frames while the camera is stable, the number of frames while camera is moving translational only and the number of frames while the camera is zooming in or out.

Table 4-1 Test videos table

	Day/ night	**Target Objects**	**Total Frame Number**	**Stable Frames**	**Translational Frames**	**Zooming Frames**
Video1	Day	Gray car	294	102	121	71
Video5	Day	Walking woman	750	406	284	59
Video9	Night	Car and pedestrian	332	73	219	40
Video19	Day	Two cars	468	284	184	0
Video24	Day	No object	622	95	404	123
Video25	Day	No object	490	82	408	0

(a) (b)



(c) (d)

(e) (f)

Figure 4-3 The videos which are used for performance evaluation. (a) is video1, (b) is video5, (c) is video9, (d) is video19, (e) is video 24 and (f) is video 25

4.1 Egomotion Estimation Results

In this thesis camera egomotion is estimated by using only the image data. Three different methods are implemented and their performances are examined. Video frames are categorized as stable, shifting, zooming and three different methods are evaluated at each of these categories. Obviously Translational RANSAC and Translational K-means methods are not evaluated at instances when the camera is zooming because scale changes cannot be modeled with these two methods.

In order to evaluate the results of the algorithms, determining the exact egomotion of the camera for all the frames in the test videos is an inconvenient way. Instead, another procedure is followed based on the initial assumption and remaining nonzero pixels in the difference image.

The initial assumption asserts that the majority of the features appear on the background and thus a model which is obeyed by the majority should be the egomotion of the camera. Based on taken experiments, it is concluded that a model which is validated by 30 percent of all the features, mostly represents the egomotion of the camera. Due to the initial assumption, a group of consistent features on a moving object cannot reach up to this ratio. All the three egomotion estimation methods applied in this thesis, obey this ratio. Performance comparison is applied by checking whether a motion model which is validated by 30 percent of the features, is found or not.

A second method is used to validate the estimated egomotion, together with the initial assumption. A function called “countNonzero” is designed for verifying the estimated egomotion. The test videos “video24” and “video25” are specially recorded such that there are no apparent moving objects on these videos while the camera is moving at the same time. Frames are aligned based on the estimated motion. Then frame differencing is applied between the two images. Since there are no moving objects on the videos, if estimated egomotion is correct, then frame differencing on these videos should result images whose pixel values are zero often.

Considering the momentary variations on the pixels intensities, a threshold is applied such that pixels whose value is smaller than 10 is deleted. Remaining nonzero pixels are counted and the estimated egomotion is validated only if the ratio of nonzero pixels is smaller than 5 percent.

Experiments are taken with video24, video25 and video1. Details of these videos are given in Table 4-1. “video24” and “video25” do not contain moving object so that both validation methods are employed. “video1” contains a mobile car and it is specially selected to test egomotion algorithms while outlier feature ratio is higher. Since there is a moving object on the video, the second validation criteria cannot be directly applied. Instead, nonzero pixels due to the moving car are discarded and the remaining nonzero pixels are counted in order to check the estimated egomotion. RANSAC is a probabilistic algorithm so RANSAC based methods are repeated for three times. Results for these algorithms are calculated based on the three repetitions.

Results for video24 are given in Table 4-2. In order to emphasize the motion, only the odd frames of the video are used. Totally 622 frames are processed. The camera is stable in 95 frames, it is panning or tilting in 404 frames and in 123 frames, the camera motion involves zooming. In the table, the “Test” lines give the number of successful egomotion estimations for corresponding algorithm and the “Result” line contains the success ratios. In the “Total” column, success ratio of algorithms through the entire video is given. The values inside the parenthesis are the success ratio at frames where the camera is not zooming. For K-means and Translational RANSAC, this value is more acceptable since these two methods did not operate at zooming frames as mentioned before. The performance of K-means and Translational RANSAC is almost the same and Linear RANSAC has a slightly fewer performance. However it works well at zoom instances unlike the other two methods.

Table 4-2 Test results for video24

Video24		**Static**	**Shifting**	**Zooming**	**Total**
		95	404	123	622
Translational RANSAC	Test 1	95	402	0	497
	Test 2	95	402	0	497
	Test 3	95	401	0	496
	Result	%100	%99,4	%0	%80(%99,5)
K-means	Test	95	402	0	497
	Result	%100	%99,5	%0	%80(%99,6)
Linear RANSAC	Test 1	95	397	122	614
	Test 2	93	398	123	614
	Test 3	95	400	122	617
	Result	%99,2	%98,6	%99,4	%98,8(%98,7)

Table 4-3 Test results for video25

Video25		**Static**	**Shifting**	**Zooming**	**Total**
		82	408	0	490
Translational RANSAC	Test 1	82	408	-	490
	Test 2	82	408	-	490
	Test 3	82	408	-	490
	Result	%100	%100	-	%100
K-means	Test	82	408	-	490
	Result	%100	%100	-	%100
Linear RANSAC	Test 1	80	397	-	477
	Test 2	81	391	-	472
	Test 3	80	394	-	474
	Result	%97,9	%96,5	-	%96,8

Table 4-4 Test results for video1

Video1		Static	Shifting	Zooming	Total
		103	120	71	294
Translational RANSAC	Test 1	103	76	0	179
	Test 2	103	78	0	181
	Test 3	103	80	0	183
	Result	%100	%65	%0	%61,5(%81,2)
K-means	Test	103	86	0	189
	Result	%100	%71,6	%0	%64,2(84,7)
Linear RANSAC	Test 1	103	72	41	216
	Test 2	103	72	37	212
	Test 3	103	72	43	218
	Result	%100	%60	%56,8	%73,2(%78,4)

Table 4-3 includes the performance evaluations on video25 and Table 4-4 includes the performance evaluations on video1. It is observed that success ratio is higher while the camera is stable. While camera is rotating or zooming, blurring occurs on the video frames. Blurring due to interlaced video standard is partially solved by simple deinterlacing implementation however blurring due to camera internal mechanism still remains. This is less effective compared to the first one nevertheless, this blurring decreases the number of SURF features thus the performance of the algorithm decreases. The second reason is that with a slight

movement of the camera, pixel intensities on a SURF feature might change significantly. Thus, a feature might not be matched on consecutive frames.

It is also observed that the performance of Linear RANSAC is very similar while the camera is rotating or zooming. Zooming action of the camera does not cause an extra reduction on the performance of the algorithm.

The performances of the three algorithms are compared with static and shifting frames. In “video24” and “video25” the performance of the translational models are very close to 100 percent. The performance of Linear RANSAC method is approximately %97. These two videos are relatively easy for egomotion estimation task since there are no moving objects. This means that erroneous features are very limited. However “video1” is a harder case since there is a moving car on the video and at some instants, outlier feature ratio ascends to 50 percent. Therefore performances are decreased at “video1”.

In “video1”, best success ratio is belongs to K-means (%84.7), Translational RANSAC (%81.2) and Linear RANSAC (%78.4) are following it. Translational RANSAC and linear RANSAC have the same principal mechanism however, their performances differ. There is one main reason for this consequence. Linear RANSAC tries to fit a three parameters model on the other hand; Translational RANSAC has a two parameters model. Model fitting is easier if a less parameter model is employed. There is a trade off here such that two parameters models can possess a higher performance for translational frames however they can not operate while the camera is zooming.

The performances of K-means and Translational RANSAC are almost equivalent whenever the ratio of outlier features is small as seen in experiments with “video24” and “video25”. However, K-means has a superior performance whenever outlier feature ratio increases. RANSAC is a probabilistic method and works based on an initial assumption set. If this initial set is not correct, then RANSAC cannot fit a model. In order to increase the performance, the algorithm is repeated with new initial sets. However if the ratio of outlier features are significantly high, RANSAC

may fail to estimate the motion. K-means has a similar mechanism but the algorithm does not start with a random feature set instead, it starts with the mean of the entire set. Moreover at each iterations, inliers are searched at a circle of lower radius thus outlier features are eliminated at each iteration. As a result, K-means based egomotion estimation outperforms RANSAC.

Whenever the initial assumption is ensured, all the three methods have sufficient performance. Moreover some precautions are taken to improve the performance of the final algorithm. At frame n, SURF features whose location contains a moving object at frame n-1, are not counted while estimating the egomotion. This eliminates some of the outlier features. As another improvement, assuming that the camera motion does not change between frame n-1 and frame n, if egomotion estimation algorithm fails at frame n, the motion model at frame n-1 is used. Experiments show that this results an explicit improvement in the final algorithm.

Finally, since Linear RANSAC has a sufficient performance and can operate while camera is zooming unlike the other algorithms, this method is chosen for the final tracking system.

4.2 Motion Detection Results

Motion detection experiments are taken on video5, video9 and video19. Target objects at each video are given in Table 4-1. Two detection methods which are outlier features based motion detection and frame differencing based motion detection are compared. The two algorithms are compared based on one criterion which is the ratio of number of frames where the target object can be detected. The algorithms are executed and the frames in which the target is detected are counted manually. Results are given in Table 4-5. The first column contains the total number of frames in which the target appears. The further columns contain the corresponding detection results for the algorithms. It is difficult to determine an objective success value for the detector algorithms since different object sizes, speeds, shapes and appearances can change the performance of the algorithms

dramatically. In test videos, the target objects are sometimes unsuitable for detection even for human eye. Thus the results in Table 4-5 are suitable only for comparing the two detection algorithms.

Table 4-5 Test results for motion detection

	# of frames target seen	Frame dif. based detection	Outlier based detection
Video5	1272	1064 (%83)	332 (%26)
Video9	332	243 (%73)	184 (%55)
Video19	381	196 (%51)	256 (%67)
Totally	1985	1503 (%76)	772 (%39)

Egomotion estimation ratios are sufficient through all the test videos thus, these videos are suitable for the detection tests. Frame differencing based method clearly outperforms the outlier features based method. Frame differencing based method achieves a detection performance up to 76 percent while outlier based method has a performance of 39 percent.

It is observed that outlier based method is good at detecting a car while it has a poor performance on pedestrians. This method relies on close outlier features whose motion is parallel. Cars are more suitable for this method. Shape of a car is fixed thus all SURF features on a car move parallel unless the car changes its angle with respect to the camera. On the other hand, shape of a walking person is not stable and its shape continuously changes during his motion. SURF features on a pedestrian do not make parallel motion thus most often they can not be recognized as a moving object. Moreover, since human has relatively small size compared to a car, less SURF features exist on a human and this causes an important performance decrease for outlier based method. Frame differencing based method has an

adequate performance on human. Since it relies on frame aligning and frame differencing, if the object size is sufficiently large, this method can successfully detect any moving object with any complex shape. Experiments also demonstrate that frame differencing based method considerably outperforms the other method at video5 since the target object is a walking pedestrian in this video.

Frame differencing based method has a lower performance on video19. The main reason is the delay in the first detection. As explained earlier, a kind of connected component search mechanism is employed at first detection. In order to relieve the computational load, a sequential scan procedure is followed. At each frame only a single line is evaluated. Thus this procedure might cause latency at first detection of the object.

By examining the results of the experiments, one advantage of outlier features based method can be realized. This algorithm can work faster compared to the other algorithm. For instance, at video9, outlier based algorithm works at an average speed of 7.2 FPS. Frame difference based algorithm works at an average speed of 5.8 FPS at the same video. This is an expected result since frame difference based method works on image pixels after egomotion estimation step and thus it is computationally more loaded.

Outlier based method sometimes makes false detections such that a static section of the video is marked as a moving object. This failure is very rare for frame difference based method. For instance, at frame 651 of video5, a false object is detected on the sidewalk. Pieces of the sidewalk have very similar shapes with each other thus wrong matches are very possible at that location. These wrong matches sometimes mislead the detector algorithm. In general, similar shaped objects at background are a common problem for both algorithms. Egomotion estimation process might also be affected by these wrong matches. However, egomotion estimation algorithms are specially designed to filter out these unwanted matches. For instance at Figure 4-4, egomotion is estimated correctly at both algorithms. Outlier based method makes a false detection. Frame difference based method is

less vulnerable to this effect. As long as egomotion is estimated correctly, this algorithm is expected to make accurate detections. This method is able to detect the real object at the same time instant.

(a) (b)

Figure 4-4 False detection of outlier features based algorithm (a) and the response of frame difference based algorithm at the same instant (b)

Another consequence obtained from the experiments is that frame difference based method is more robust. A detected object is not lost as long as the target object appears in the video and is sufficiently large. Outlier based method frequently looses and relocates target objects. Furthermore, frame difference based method, locates object boundary better while outlier based method sometimes partially locates the object. Figure 4-5 demonstrates this phenomenon. This situation can be observed through all the test videos.

(a) (b)

Figure 4-5 Outlier based detector, partially locates the object (a) while the object is located better with frame difference based detector (b)

Experiments prove that even though frame difference based detection is computationally more loaded, it has a superior detection performance compared to its competitor. False detection rate is lower and a detected object is not lost most often. Moreover object boundaries are better located. Finally it was concluded to use frame difference based detection technique for the final detector-tracker algorithm.

4.3 Detector-Tracker System

A detector-tracker system is developed depends on the main goal of this thesis work. When the detector is initiated, moving objects are searched. Detected objects are marked and the camera is directed to the target if the tracker is initiated by the user. A simple GUI is designed to handle user commands. Different buttons are available to initialize the camera, the detector and the tracker. Moreover, vibration and sweep modes are added to the GUI. While the camera is not tracking, user is able to vibrate the camera or a large area can be scanned. In vibration mode, the camera makes small circular movements. In sweep mode, the camera makes long

pan movements through right and left thus moving objects can be searched through a large area. These modes are valuable for testing the performance of the detector while the camera is moving.

At egomotion estimation step, Linear RANSAC method is chosen since it can model the camera egomotion while the camera is zooming. Then detection is performed by frame difference based detector. If track command is active, the Euclidean distance between the object center and the image center is calculated. Pan and tilt speed of the camera is arranged proportional to this distance and camera is directed through the target object. Moreover object size is stabilized by using the zoom capability of the camera. The algorithm checks the size of object marker rectangle. If the length of the diagonals is smaller than a lower limit, then the camera zooms in to the object. If the length of the diagonals is larger than an upper limit, then the camera zooms out. Flow diagram of the final tracking system is given in Figure 4-6.

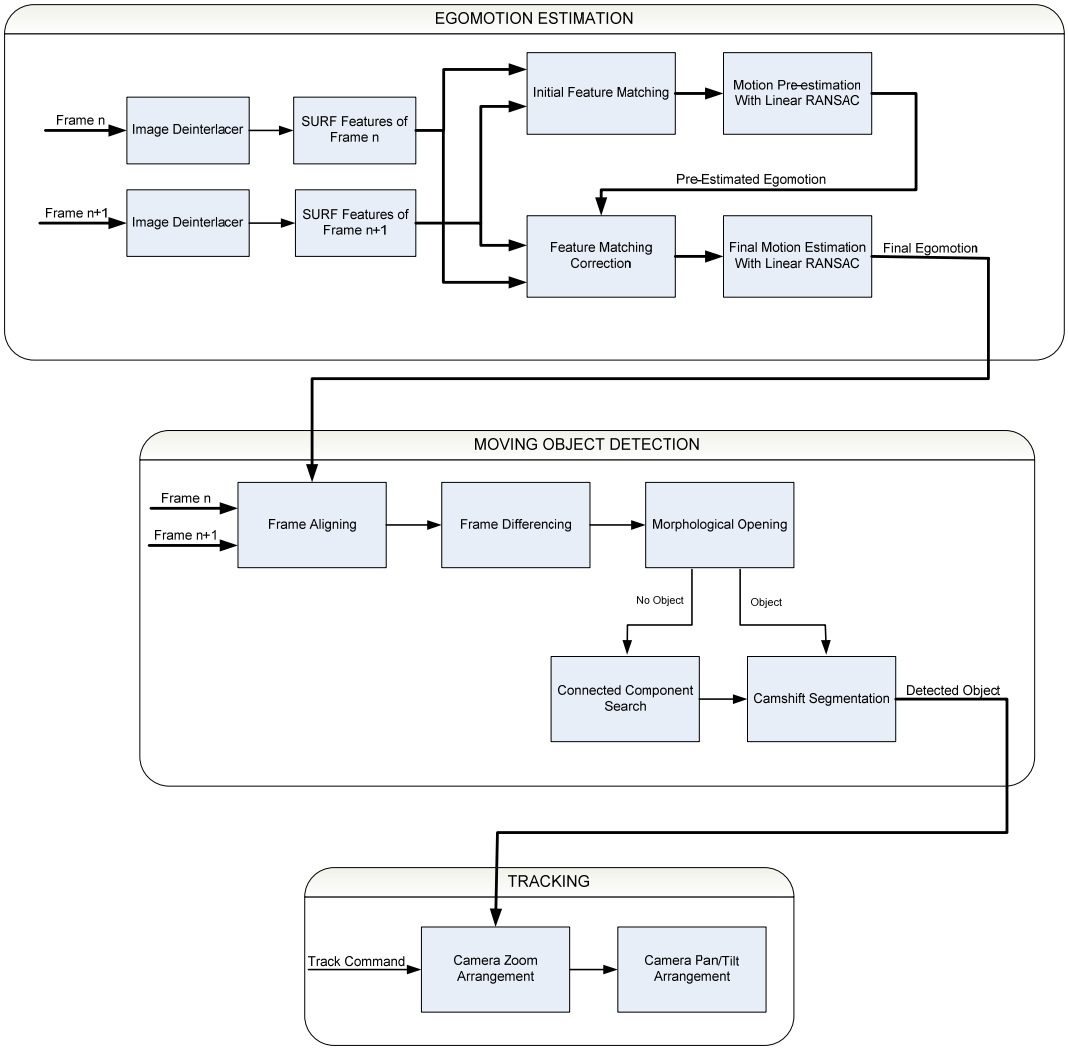
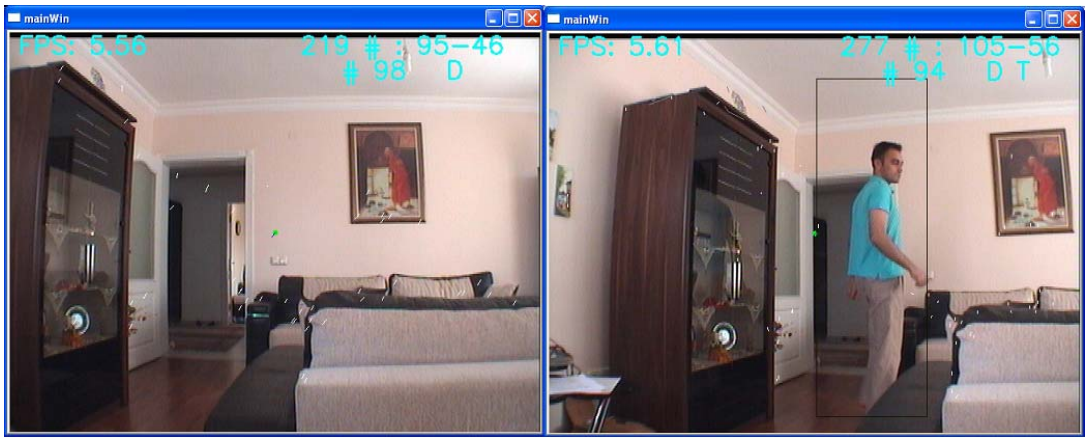
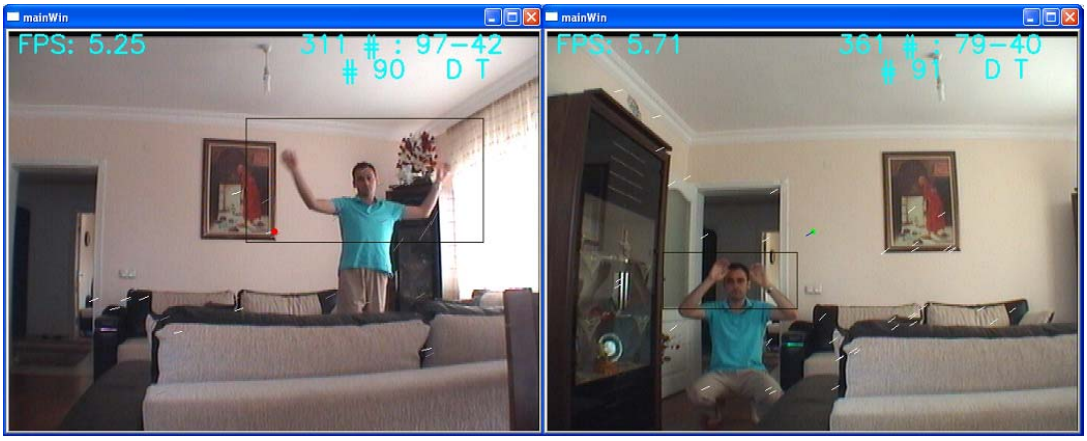


Figure 4-6 Flow diagram of the tracker system

Example videos are recorded with the final detector-tracker algorithm. Figure 4-7 is an indoor demonstration. In this video, vibration function is active until the tracker is enabled. The target person is successfully detected and tracked through the video.



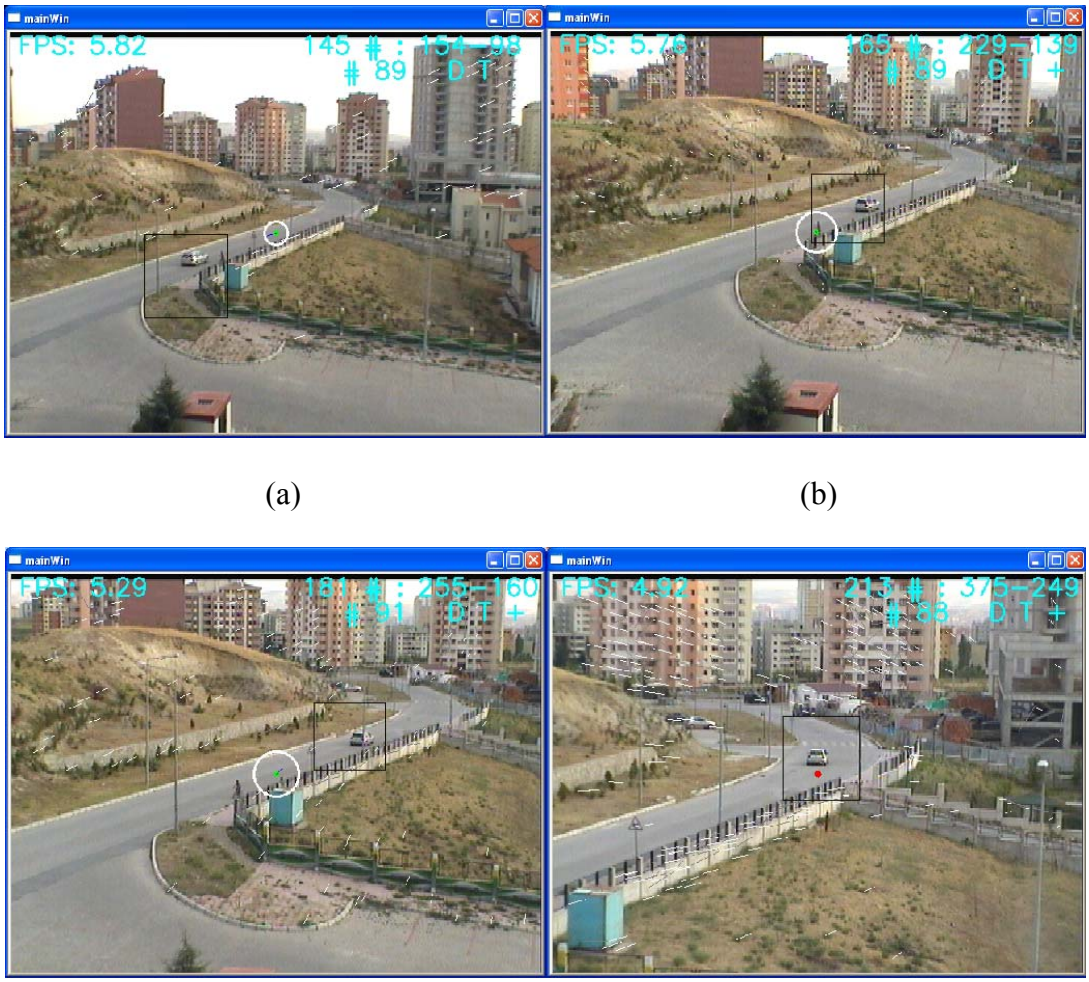
(a) (b)



(c) (d)

Figure 4-7 Indoor tracking experiment

Figure 4-8 is an outdoor demonstration of the algorithm. Target car moves away from the camera thus the software is continuously zooming the camera to the car. Zoom in symbology is seen on the center of the video. Note that at frame (d) egomotion cannot be estimated but the detector is still operational.



(a) (b)

(c) (d)

Figure 4-8 Outdoor tracking experiment. The camera zooms in to the target car since its size is smaller than expected.

Figure 4-9 demonstrates another outdoor experiment. Target object are walking people this time. Pedestrians are detected at frame (a) and software zooms to them for an amount as seen in frame (b). Frame (c) and frame (d) are examples of partially occlusion. The algorithm successfully continues to detect the target. The tracking task continued more than 750 frames without loosing the target at any time instant.

Figure 4-9 Outdoor tracking experiment. Target is pedestrians

CHAPTER 5

CONCLUSION

5.1 Summary and Conclusions

Due to the increasing needs to surveillance applications and due to the technological improvements, the number of active surveillance systems is continuously growing through out the world. It is not an efficient way to operate all these systems only with human operators. This is an expensive solution. Instead, today's technology has a tendency to reduce the number of human operators by using automated surveillance software. Automated detection and tracking is one of the most common research areas.

Stable cameras have a limited line of sight and generally a single static camera is not sufficient to monitor a wide terrain. A few of them are necessary to completely cover the entire terrain. However, a clearly located PTZ camera can alone monitor a wide area. For example a PTZ camera and automated tracker software, can detect any violation to a terrain. On the other hand, active camera tracker systems have a main disadvantage such that the algorithms for active cameras are generally more complex compared to the algorithms for static cameras.

In this thesis, an automated tracker system for active cameras is developed. It was aimed to develop a computer based solution which is capable of real time working. In order to ensure real time performance, C++ language and OpenCV library are used for algorithm development. Similar applications in the literature are analyzed. It is observed that due to computational efficiency, feature based detection methods

are frequently used. This idea is adapted to this thesis work and SURF features are chosen as the main building block of the proposed algorithm.

SURF features are matched between consecutive frames and motion vectors are obtained on the image. Assuming that majority of these motion vectors belong to the background, wrong matches which are not compatible with the majority, are eliminated and a camera motion model is constructed. Three different methods are implemented and evaluated with some test videos. All the three methods are convenient if outlier features are limited however K-means based method outperforms the other two when ratio of outlier features is higher. Translational RANSAC and K-means can model only pan and tilt movements of the camera while Linear RANSAC can also model zoom movements. For this reason, this method is employed in the final tracking system.

Two different motion detection methods are implemented. The first one is a very popular method used in the literature. Consecutive frames are aligned with respect to the estimated egomotion and then frame differencing is performed as like the camera is stable. The second method relies on the outlier features on the image. Parallel and close outlier features are recognized as moving objects. This method is computationally more efficient however, it is very vulnerable to wrong matched features. If there are relatively more wrong matches, this algorithm might detect false objects. Moreover, this method is not successful to detect objects whose shape is variable. Thus this method is not suitable for tracking pedestrians. On the other hand, frame difference based method does not suffer from the same drawbacks. Features are only used for egomotion estimation. Experiments verify that this method is more suitable to detect walking people.

According to the initial goal of the thesis, a detector-tracker system was developed. The developed algorithm can work approximately at 5 FPS and this speed is acceptable for this thesis work. The algorithm is tested with different targets and appropriate results are obtained as long as the initial assumption is valid in the test scene.

5.2 Future Work

The proposed algorithm in this thesis is capable of working at a speed of 5 FPS. By increasing the speed of the algorithm, smoother camera motion can be obtained. Moreover faster objects can be tracked since the response time of the camera is increased. In order to achieve this, efficiency of the algorithm should be enhanced. Before feature matching step, features can be indexed based on their locations thus a pair for a feature is not searched through the entire feature set. Or a better feature type might be found instead of SURF to increase the speed. Instead of computer, the algorithm might be implemented on an FPGA. The speed of the algorithm can be increased with parallel processing.

The proposed algorithm has a zooming limitation through a target object since the principal assumption should be assured for proper egomotion estimation. Moreover lack of blob regions on the background might result to poor egomotion estimation. In order to overcome this situation, supporting hardware can be used for egomotion estimation task.

REFERENCES

[1] H. Dee and S. Velastin. How close are we to solving the problem of automated visual surveillance? Machine Vision and Applications, 19(5-6):329-343, 2008.

[2] I. S. Kim, H. S. Choi, K. M. Yi, J. Y. Choi, and S. G. Kong, “Intelligent Visual Surveillance - A Survey”, International Journal of Control, Automation, and Systems (2010), pp. 926-939.

[3] Alessandro Bevilacqua , Pietro Azzari, High-Quality Real Time Motion Detection Using PTZ Cameras, Proceedings of the IEEE International Conference on Video and Signal Based Surveillance, p.23, November 22-24, 2006

[4] A. Yilmaz , O. Javed and M. Shah "Object tracking: A survey", ACM Comput. Surv., vol. 38, No 4 , 2006.

[5] R. Cucchiara, A. Prati, and R. Vezzani , “Advanced video surveillance with pan tilt zoom cameras,” Proc. of the 6th IEEE International Workshop on Visual Surveillance, 2006.

[6] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision", Proc. Int. Joint Conf. on Artificial Intelligence, pp.674 - 679 , 1981.

[7] D. Murray and A. Basu, "Motion tracking with an active camera", IEEE Trans. Pattern Anal. Machine Intell., vol. 19, no. 5, pp.449 - 454 , 1994.

[8] S. Kang, J. Paik, A. Koschan, B. Abidi, M. A. Abidi. Real-time video tracking using ptz cameras. In Proc. of SPIE 6th International Conference on Quality Control by Artificial Vision, pages 103–111, Gatlinburg, TN, May 2003.

[9] Y. Sugaya, K. Kanatani, Extracting moving objects from a moving camera video sequence, in: 10th Symposium on Sensing and Image Information, Yokohama, Japan, June 9–11, 2004, pp. 279–284.

[10] C.M. Wang, Y.J. Chang, Y.C. Chen, Realtime object extraction and tracking with an active camera using image mosaic, Proceedings of the IEEE International Workshop on Multimedia Signal Processing, Virgin Islands, USA, December 2002.

[11] D. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” Int'l J. Computer Vision, vol. 2, no. 60, pp. 91-110, 2004.

[12] S.M. Smith and J.M. Brady, “SUSAN—A New Approach to Low Level Image Processing,” Int"l J. Computer Vision, vol. 23, no. 1, pp. 45-78, 1997.

[13] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In CVIU (1), pages 346–359, 2008.

[14] T. Tuytelaars and K. Mikolajczyk, "Local Invariant Feature Detectors: A Survey," *Foundations and Trends in Computer Graphics and Vision,* vol. 3, no. 3, pp. 177-280, 2008.

[15] L. Juan and O. Gwun, A comparison of SIFT, PCA-SIFT and SURF, International Journal of Image Processing 3 (2009), pp. 143–152.

[16] G. L. Foresti and C. Micheloni, “A robust feature tracker for active surveillance of outdoor scenes”, Electronic Letters on Computer Vision and Image Analysis 1(1):21-34, 2003

[17] J. Shi and C. Tomasi, "Good features to track", Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, pp.593 - 600 , 1994.

[18] D. Zhou, L. Wang, X. Cai, and Y. Liu, “Detection of Moving Targets with a Moving Camera”, International Conference on Robotics and Biomimetics, 2009

[19] R.C. Bolles and M.A. Fischler, A RANSAC-based approach to model fitting and its application to finding cylinders in range data, Proc. Int. Joint Conf. Artif. Intell. Vancouver, Canada (1981), pp. 637–643.

[20] S. Bayrak “Video stabilization: digital and mechanical approaches” M.S. Thesis, METU, 2008

[21] N. Pejčić, N. Reljin, S. McDaniel, D. Pokrajac, A. Lazarević, “Detection of moving objects using incremental connectivity outlier factor algorithm”, ACMSE, 2009

[22] Ta, D.N., Chen, W.C., Gelfand, N., Pulli, K.: SURFTrac: Efficient tracking and continuous object recognition using local feature descriptors. In: CVPR (2009)

[23] A. Torii, M. Havlena, T. Pajdla, Omnidirectional image stabilization for visual object recognition, Int Journal Comput Vis (2011) 91, pp. 157–174

[24] W. He, T. Yamashita, H. Lu, and S. Lao. Surf tracking. In ICCV, 2009.

[25] C.Evans, “Notes on opensurf library”, University of Bristol Tech. Rep. CSTR-09-001, 2009

[26] www.wikipedia.org, “K-means clustering”, last accessed date: 05.09.2011.

[27] Bradski , G. R. (1998): Computer vision face tracking for use in a perceptual user interface. Intel Technology Journal, 2nd Quarter, 1998.

[28] K. Fukunaga and L.D. Hostetler,“The estimation of the gradient of a density function, with applications in pattern recognition,” IEEE Trans. Information Theory, vol. 21, pp. 32-40, 1975.

[29] www.wikipedia.org, “Mean-shift”, last accessed date: 05.09.2011.

[30] Z.Wang, X.Yang, Y.Xu, S.Yu, ”Camshift guided particle filter for visual tracking”, Proc.IEEE workshop Signal Proc. Systems, pp.301- 306,2007.

[31] J. G. Allen, R. Xu, J. S. Jin, "Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces", Proceeding of Pan-Sydney Area Workshop on Visual Information Processing, Sydney, 2003.

[32] Hongxia Chu et al. 2007, Object Tracking Algorithm Based on Camshift Algorithm Combinating with Difference in Frame, IEEE Automation and Logistics,18-21 Aug. 2007, page: 51-55

[33] SONY Corporation,"EVID100 Technical Manual", November 2001.