BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Optical Flow Based Object Motion Tracking With Cascaded Outlier Rejection

By

Md. Hasibul Hoque (084401)

Md. Nahid Hossain (084446)

Supervised By

Dr. Md. Hasanul Kabir

Co-supervised By

Faisal Ahmed

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT)
Organization of the Islamic Cooperation (OIC)
Dhaka, Bangladesh

October, 2012

Optical Flow Based Object Motion Tracking With Cascaded Outlier Rejection

By

Md. Hasibul Hoque (Student No: 084401)

Md. Nahid Hossain (Student No: 084446)

Supervised By

Dr. Md. Hasanul Kabir

Co-supervised By

Faisal Ahmed

A Thesis Submitted to the Department of Computer Science and Engineering (CSE) in Partial Fulfilment of the Requirements for the Award of the Degree of Bachelor of Science in Computer Science and Engineering (B.Sc. in CSE).

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT)

Dhaka, Bangladesh

October, 2012

Abstract

Object tracking is one of the main areas in computer vision. Tracking is basically a process of locating a moving object over time using any kind of sensing device. Here the image sensing device works as the input device and the whole tracking is done by processing the image information. To track an object is not an easy task because of various limitations of the input device and the processing ability of the processors. The underlying scientific phenomenon of the dynamics and physics of this mysterious world make it worse. Our tracker was developed to track an object regardless of its any feature, for example, the shape, color or intensity, illumination change. It was also developed to work in fast camera motion movements. To have an idea of the motions of every pixel in two consecutive frames we have to determine the motion vectors at first. Using Lucas-Kanade method we managed to find the motion vectors. Now it is the problem of determining the actual motion vector which has caused the tracked object move. We used the Global motion estimation from the coarsely sampled motion vector field. We incorporated the cascaded outliers rejection method where outliers indicates the noises. It was incorporated after getting the motion vectors at first stage and before calculating the global motion estimation. This system was experimented on three different videos which had distinctive characteristics. Comparing with the state of the art trackers our tracker showed a very good performance and sometimes better than those.

Acknowledgment

First of all, we would like to thank the Almighty Allah for all the blessings he has bestowed upon us throughout our entire life, without the mercy of Allah, we couldn't make everything. "All thanks and praises be to Allah".

I would like to thank my thesis supervisor, Dr. Md. Hasanul Kabir, for his support and guidance on this thesis. Dr. Kabir has been an instrumental to this work and my career. He taught me how to do research, think critically and find out useful solutions. He provided me with a strong vision for what this should accomplish, without his insights the "Optical Flow Based Object Motion Tracking With Cascaded Outlier Rejection" would not have reached completion. Sir may the Al-Mighty Allah reward you abundantly.

Again, it was our pleasure to get the cooperation and coordination from Head of The Department, Prof. Dr. M.A. Mottalib during various phases of the work. We are grateful to him for his constant and energetic supervision, constructive criticism and valuable advice.

Special thanks goes to the Faisal Ahmed, Lecturer, CSE Dept., Islamic University of Technology, for his kind suggestion and cooperation regarding implementation of our algorithm.

Finally, we like to offer thanks to all who contributed us in any way during the project.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Information retrieval and data analysis is the most necessary field in today's world. To survive in this competitive world this field is becoming popular day by day .In every aspect of life people now think about how to involve the technology and make their life easier .An easier life wants to reduce their burden by giving their task to others. From this thought the automated system or the process that can be done by the computer itself is being developed .One way to do this is to retrieve enough information about the task and analyze in a way so that the task can be done without any direct help of the human .Moreover to increase the accuracy and efficiency of the task. Computer vision now-a-days is a buzz word in this world .People now wants to see the world in more precise way with the computer. They want to leave many of their tasks related to the eye monitoring to the automated computer vision .Visual tracking is one of the fascinating area of this field. Our topics are about the object tracking.

Tracking can have many classifications, branches according to their nature, uses, and procedures. Object tracking is one of a kind where we want to track an object. The object can have a special shape or sizes or color or any other feature. In another kind we can track any kind of shapes, sizes or color or any other feature. We have just to select the object regardless of any feature. But many of us still think about what is actually meant by tracking. If we present an example we think it will be clear for many of our readers. For example we

are watching a football game with our own eyes. But one of us doesn't know the best player in the team. So others are helping him to make him known to him. Let's say we helped him to recognize that the jersey no 9 is the best player and he should follow him to every step so that he can enjoy the game best even the player doesn't possess the ball all time . He then watched him throughout the game like a spy and didn't watch any of the other players. This is called tracking because tracking is all about to follow in every step .Whatever happens in rest of the world is not concerned here. The motion or the environment or the other situation should not hamper here.

Our tracking concentrated on any kind of shaped object tracking. We are not narrowing down our focus in any particular environment. Whatever fast movements happen or the motions of the object or the other backgrounds are it should work.

There are many challenges related with this system because of its nature. The most difficult problem is that we actually don't know the mystery of the motion of the objects that we see are moving. There are many underlying things happening in front of our eyes that we cannot perceive. There are many complex motions happening all the time with incredible synchronization. But if we want to describe the technical difficulties we have to first mention about the processing power. Our human eyes capture the images and send it to the brains which have the immense calculation power and the storage. The brain not only depends on calculating the motion trajectory, it also does the synchronization with the previous stored images to solve the reference problem. We have processors that cannot meet the calculation of any complex motion. The capture device also does not have the sufficient input quality. We have to work with very little resources. There is not enough optimized algorithm also. It creates the problem more difficult.

Although we have many challenges in this field we have a very wide and interesting application fields for this. It is being implemented in numerous applications,

for example, security, surveillance [17], human-computer interaction [18], traffic pattern analysis [19], recognition, medical image processing [20]. In these all fields tracking is there but the object may vary according to the subject of purpose. For example in surveillance one may choose any particular person to track wherever he goes. In a densely populated area among all the people it should track only that people.

1.2 Problem Domain

We have talked little about tracking. Now we want to dig into a little deeper. There are many problems and challenges in tracking an object. What we see from other tracker is that still no tracker can solve all the problems. If any tracker wants to do these it makes the program very difficult to handle all the aspects and it cannot encompass all the calculation in it. As a result considering all the technical limitation it is very slow and show very poor performance. So when any tracker is being developed there focus of interest has to be narrowed down to a small set of problems. Our area of interest is to develop a robust and cost effective system that allows user to track any kind of selected object. Challenges often encountered in object tracking:

- Loss of information caused by the projection of the 3D world on a 2D image
- Illumination change
- Object deformation
- Cluttered-background
- Complex object motion
- Full occlusion
- Partial occlusion
- In-plane rotation

- Out-of-plane rotation
- Fast object motion or moving background
- Real-time processing requirements

1.3 Research Challenges

We have some challenges in our research what we want to solve. These are discussed below:

- **Illumination change**: People don't want to bother about the illumination problem when they track. There any time may happen a low illumination in an environment .And a particular range of illumination shouldn't be chosen either so we try to not to focus any kind of region of illumination to run our program.
- **Accuracy in detection**: This is the main focus of research problem. Many trackers don't track the object correctly. There error rate is not negligible, so they are useless in most of the applications. We hope that whatever motion complexity is our program should track it with the minimalist error.
- **Any kind of shape of Objects**: The object shape can vary largely. We should not limit it with only rectangle shape, triangle shape or circular shape. Our user will choose any kind of shape they want to track. A particular shape may make the algorithm easier but it cannot be usable for the general use.
- **Rotation invariant**: Rotation invariant tracker means that if we rotate any tracked object they should not lose the object in the proceedings. As we know the object is still there, so to provide the accuracy in the tracking we have to ensure the rotation invariant feature. Rotation can make an object appear completely different; so many trackers can differentiate it from its original appearance.

- **Fast object motion**: Due to the camera configuration and the computation complexity the fast object motion tracking is avoided by most of the trackers. But if we look at the basic purpose of tacking we have to concentrate on fast object motion tracking. Though we have technical limitation we have to overcome it with our algorithm.

1.4 Motivation

Our main motivation to do this research is to ensure tracking accuracy. There are many researches going on to develop a good tracker but no tracker is providing a good accuracy. The tracking of fast movement of objects is another motivation for us. Fast motion of object is a very normal phenomenon in our daily life. If we cannot incorporate in our system then its application range will fall down to very small range. These kind of things a make tracker more usable to a user and we want that.

1.5 Research contribution

Our tracker is cautious about the computation complexity. To minimize the computation complexity it doesn't calculate the same frame twice. The consecutive frames are necessary to track a single transition. Most of the trackers at first calculate the present frame, then the next frame and then to compute the motion they have to go back to the previous frame again.

In this way every frame is being computed at least twice. Our tracker is a single pass tracker where a frame is computed only once and thus has less computation. It doesn't store the frame information also. So it has minimum space complexity also. For example the state of the art TLD tracker [8], [9], [10] uses the forward-backward procedure where they have to compute every single frame twice. They also store different appearance of the object, so they have space complexity too.

We have incorporated the outlier rejection on the motion vector so that we find the actual motion that actually happened among all the noises from the background. The noisy motion always corrupts the actual data; as a result the accuracy decreases in a great manner. An outlier rejection should be considered as an intelligent way to handle the situation.

1.6 Thesis outline

In Chapter 1 we have talked about the introduction of our study in a precise manner. Chapter 2 deals with some definition of tracker, how motion vectors are generated, some outlier rejection methods, how Global motion estimation works and how some state-of-the-art existing tracker works. Chapter 3 will be discussed about our proposed algorithm and some elaborate discussion. Chapter 4 will consist of the experimental analysis and result comparisons.

Chapter 2

Literature review

2.1 Object Tracking

Object tracking can be defined as tracking a single or multiple objects over a sequence of images [1]. So by just processing the current frame the tracker tracks the object on the next image and then the next one. The object or objects are at first selected on the first frame. There are lots of challenges that are associated with tracking an object. They are - illumination, pose, scale, deformation, motion blur, noise, and occlusion. By considering these challenges different trackers are developed. But there is not a single tracker that can overcome all these challenges. So according to the need specific tracking algorithms are used for specific environments.

2.2 Key components of a tracker

Although different online object tracker has different components and features according to the specific environment, there are some common components all the trackers have. According to [1] there are three key components of a tracker -

- Object representation
- Dynamic model
- Search mechanism

We are going to discuss these three key components plus some other components of tracking.

2.2.1 Object Representation

An object in an image can be represented as holistic descriptors or local descriptors. Some common holistic descriptors are color histograms and raw pixel values. The advantage of histogram-based representation that they can effectively handle object deformation as well as occlusion. However they cannot handle scale changes and do not exploit the structural information of the object. Local descriptors are used mainly because of their robustness to pose and illumination change.

2.2.2 Adaptive Appearance model

For ensuring robust tracking performance it is crucial to update the appearance model. The most straightforward and easiest way is to replace the current appearance model with the information from the most recent tracking result. However this type of tracking causes drifting in the tracking which leads to a very noise output. There are many other update algorithms. In SemiT [5] the update algorithm formulates the update problem as a semi-supervised task where the drawn samples are treated as unlabeled data. The task is then to update a classifier with both labeled and unlabeled data. Specific prior is used in BeSemiT [6] approach to reduce drifts. In MILT [7] the tracking problem within the multiple instances learning (MIL) framework to handle ambiguously labeled positive and negative data obtained online for reducing visual drifts. TLD [8] pose the tracking problem as a semi-supervised learning task and exploit the underlying structure of the unlabeled data to select positive and negative samples for update.

2.2.3 Motion Model

The dimensionality of state vector x_t, at time t depends on the motion model that a tracking method is equipped with. The most commonly adopted models are translational motion (2 parameters), similarity transform (4 parameters), and affine transform (6 parameters).

2.2.4 Dynamic Model

A dynamic model, either predefined or learned from certain training data, is often used to predict the possible target states (e.g., motion parameters) in order to reduce the search space and computational load. A dynamic model is usually utilized to reduce computational complexity in object tracking as it describes the likely state transition, i.e.,$p(x_t|x_{t-1})$, between two consecutive frames where x_t is the state vector at time t.

2.2.5 Search Mechanism

Based on the search mechanism the tracking algorithms are categorized as deterministic or stochastic. The searching mechanism is done using an objective function which maximizing or minimizes the results based on distance, similarity or classification measures, such as the Lucas-Kanade [2] algorithm which is a deterministic method uses the sum of squared distance to measure the error. Kalman filter [3] is also another deterministic method. TLD uses the Lucas-Kanade method for searching. By considering observations over multiple frames within a Bayesian formulation stochastic methods usually optimize the objective function. It improves robustness over deterministic methods by its capability of escaping from local minimum with much lower computational complexity than sampling-based methods that operate on each frame independently.

2.3 Types of tracker

According to the tracking of an object and how can it track to different situation we can categorize trackers of mainly two types:

- Short-term tracker
- Long-term tracker

The definition of these trackers are given below-

- **Short-term tracker**: Short term trackers track an object for short period of time. Short term trackers cannot track an object if it gets occluded by another object or disappear from the view of the camera and reappear again because it doesn't store any kind information regarding the object. Rather it only calculates the motion of the object. Although it might seem that short term trackers has no use in real time applications but to build the long-term tracker a short-term tracker is essential. The better the short-term tracker is the better the tracker would be able to track the object perfectly and find its trajectory. Lucas-kanade [2] is a widely used optical flow estimation which is used to determine the motion vectors of a short-term tracker. Another short-term tracker could be the template matching algorithm which is a really primitive and brute force type of tracker which is not very effective.

- **Long-term tracker**: Long-term trackers can track object even if it gets occluded or disappears from the view of the camera and reappears again. A good long-term tracker can track an object for a very long amount of time. Long-term trackers use short term trackers to find the motion trajectory of the object. It also stores information about the object. So even if the object does disappear or get partially occluded by another object and then reappears again a good long term tracker can locate that object and then using the short term tracker to track it again. Some examples of good long-term tracker are - TLD [8] [9] [10], BoostT [4], SemiBoostT [5] etc. TLD uses the Lucas-kanade method as the short-term trackers.

2.4 Optical Flow

Optical flow or optic flow calculates the motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer (an eye or a camera) and the scene [11]. Optical flow specifies how much the pixel has moved from its adjacent images. The moving patterns cause temporal varieties of the

image brightness. It is assumed that all temporal intensity changes are due to motion only.

But in case of optical flow there are some constraints. There can be no occlusion (one object moving in front of/or behind another object), unless it is modeled in such a. All objects in the scene are rigid, no shape changes allowed. This assumption assures that optical flow actually captures real motions in a scene rather than expansions, contractions, deformations or shears of various scene objects.

Let's assume in Fig 2.1 $I(x, y, t)$ is the center pixel in a nn neighborhood and moves by δx, δy in time δt to $I(x + \delta x, y + \delta y, t + \delta t)$. Since $I(x, y, t)$ and $I(x+\delta x, y+\delta y, t+\delta t)$ are the images of the same point (and therefore the same) we have:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{2.1}$$

The assumption is true to a first approximation (small local translations) provided δx, δy, δt are not too big. We can perform a 1st order Taylor series expansion about $I(x, y, t)$ in equation to obtain:

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\delta I}{\delta x}\delta x + \frac{\delta I}{\delta y}\delta y + \frac{\delta I}{\delta t}\delta t + \text{H.O.T.} \tag{2.2}$$

Where H.O.T. are the Higher Order Terms, which we assume are small and can safely be ignored. Using the above two equations we obtain:

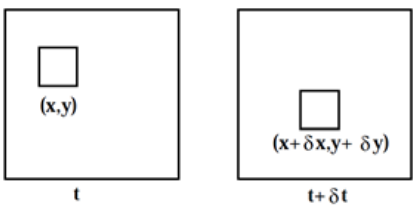


Figure 2.1: The image at position (x, y, t) is the same as at $(x+\delta x, y+\delta y, t+\delta t)$.

$$\begin{aligned}&\tfrac{\delta I}{\delta x}.\delta x + \tfrac{\delta I}{\delta y}.\delta y + \tfrac{\delta I}{\delta t}.\delta t = 0 \text{ or}\\ &\tfrac{\delta I}{\delta x}.\tfrac{\delta x}{\delta t} + \tfrac{\delta I}{\delta y}.\tfrac{\delta y}{\delta t} + \tfrac{\delta I}{\delta t}.\tfrac{\delta I}{\delta I} = 0 \text{ and finally:}\\ &\tfrac{\delta I}{\delta x}.v_x + \tfrac{\delta I}{\delta y}.v_y + \tfrac{\delta I}{\delta t} = 0.\end{aligned} \tag{2.3}$$

Here $v_x = \frac{\delta x}{\delta t}$ and $v_y = \frac{\delta y}{\delta t}$ are the x and y components of image velocity and $\frac{\delta I}{\delta x}$, $\frac{\delta I}{\delta y}$, $\frac{\delta I}{\delta t}$ are image intensity derivatives at (x, y, t). We normally write these partial derivatives as:

$$I_x = \frac{\delta I}{\delta x}, I_y = \frac{\delta I}{\delta y} \text{ and } I_t = \frac{\delta I}{\delta t}. \tag{2.4}$$

The difference between (v_x, v_y) which are the x and y components of optical flow and (I_x, I_y, I_t) which are intensity derivatives. This equation can be rewritten more compactly as:

$$(I_x, I_y).(v_x, v_y) = -I_t \tag{2.5}$$

or as:

$$\nabla I.\vec{v} = -I_t \tag{2.6}$$

where $\nabla I = (I_x, I_y)$ is the spatial intensity gradient and $\vec{v} = (v_x, v_y)$ is the image velocity or optical flow at pixel (x, y) at time t. The equation here has two unknowns and thus cannot be solved by using the equation alone. To find the optical flow a set of equations are needed given some constraints.

2.4.1 Lucas Kanade Method

In computer vision, the Lucas-Kanade method is a widely used differential method for optical flow estimation developed by Bruce D. Lucas and Takeo Kanade [2]. It assumes that the flow is essentially constant in a local neighborhood of the pixel under consideration, and solves the basic optical flow equations for all the pixels in that neighborhood, by the least squares criterion.

The Lucas-Kanade method assumes that the displacement of the image contents between two nearby instants (frames) is small and approximately constant within a neighborhood of the point under consideration. So it takes into account the information of the neighborhood for the considered point. It devises an equation which has two unknown variables. The equation is $Av = b$ where A is the partial derivative of image with respect to x and y , v is the motion vector(V_x, V_y) and the b is the derivative matrix of image t with respect to time t.

From the above equation we can see in the motion vector (V_x, V_y) the two variables are unknown . If we need to evaluate it we need more equations and that is provided by the neighborhood pixels. Thus this method calculates other pixels and tries to find the velocity or the motion vector of one pixel.

2.5 Outlier Rejection

After generating a MV (motion vector) we try to estimate the Global motion of that particular object. But before estimating the global motion of the selected object we need to reject the outliers. Outliers are those noisy outputs in the MV that might cause to give bad output while estimating the global motion. These outliers are rejected using different outlier rejection schemes. After rejecting the outliers we are left with the inliers from which we can get almost accurate global motion. We are now going to discuss three outlier rejection methods.

2.5.1 MV Outlier Rejection Cascade

MV outlier rejection cascade [12] is used for removing the outliers from the MV and also to speed up the GME process.

Here three cascaded filter is used. These filters have their own way to filter out the outliers. The first filter computes the individual MVs from the neighborhood MVs, the second filter computes the averages of diagonally opposite MVs from

the neighborhood MVs and the last filter computes the averages of triangularly opposite MVs from the neighborhood MVs.Thus in every steps it filter out the outliers depending on some threshold value.The last inliers are ready to go for the global motion estimation then. Using this technique we can remove the outliers

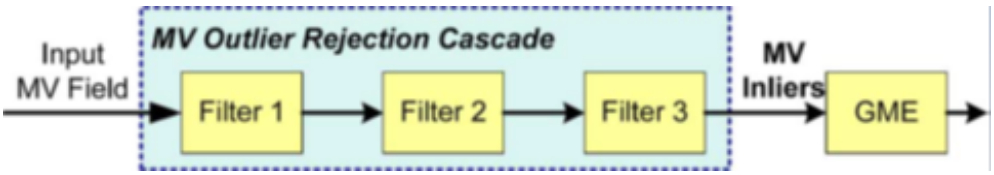


Figure 2.2: Motion vector outlier rejection cascade work flow

according to the percentage we want.

2.5.2 Forward Backward error

Forward-backward consistency assumes that correct tracking should be independent of the direction of time-flow [10]. Given a point location in time t, the goal is to estimate its location in time $t + 1$.

Let $S = (I_t, I_{t+1}, \ldots, I_{t+k})$ be an image sequence and x_t be a point location in time t. Using an arbitrary tracker, the point x_t is tracked forward for k steps. The resulting trajectory is $T_f^k = (x_t, x_{t+1}, \ldots, x_{t+k})$, where f stands for forward and k indicates the length. Our goal is to estimate the error (reliability) of trajectory T_f^k given the image sequence S. For this purpose, the validation trajectory is first constructed. Point x_{t+k} is tracked backward up to the first frame and produces $T_b^k = (\hat{x}_t, \hat{x}_{t+1}, \ldots, \hat{x}_{t+k})$, where $\hat{x}_{t+k} = x_{t+k}$. The Forward-Backward error is defined as the distance between these two trajectories:

$FB(T_f^k|S) = \text{distance}(T_f^k, T_b^k)$. Various distances can be defined for the trajectory comparison. For the sake of simplicity, we use the Euclidean distance between the initial point and the end point of the validation trajectory, distance $(T_f^k, T_b^k) = ||x_t - \hat{x}_t||$.

After finding the distance for all the pixels we set a threshold and eliminate all those pixels whose distance value exceeds the threshold. We can than keep the inliers and run the GME on the inliers.

2.5.3 Random Sample Consensus (RANSAC)

RANSAC is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers [13]. It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability, with this probability increasing as more iteration are allowed. A basic assumption is that the data consists of "inliers", i.e., data whose distribution can be explained by some set of model parameters, and "outliers" which are data that do not fit the model. In addition to this, the data can be subject to noise. The outliers can come, e.g., from extreme values of the noise or from erroneous measurements or incorrect hypotheses about the interpretation of data. RANSAC also assumes that, given a (usually small) set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data. The input to the RANSAC algorithm is a set of observed data values, a parameterized model which can explain or be fitted to the observations, and some confidence parameters. RANSAC achieves its goal by iteratively selecting a random subset of the original data. These data are hypothetical inliers and this hypothesis is then tested as follows:

a) A model is fitted to the hypothetical inliers, i.e. all free parameters of the model are reconstructed from the inliers.

b) All other data are then tested against the fitted model and, if a point fits well to the estimated model, also considered as a hypothetical inlier.

c) The estimated model is reasonably good if sufficiently many points have been classified as hypothetical inliers.

d) The model is re-estimated from all hypothetical inliers, because it has only been estimated from the initial set of hypothetical inliers.

e) Finally, the model is evaluated by estimating the error of the inliers relative

to the model.

This procedure is repeated a fixed number of times, each time producing either a model which is rejected because too few points are classified as inliers or a refined model together with a corresponding error measure. In the latter case, we keep the refined model if its error is lower than the last saved model.

2.6 Global Motion Estimation (GME)

When we calculate the motion vectors of particular frames it means every pixel of that frame has that motion in between the consecutive frames. By looking at the direction of the vectors we can understand the trajectory of the object we were tracking. But there are noises which can make many unnecessary vectors and it can lead us to a confusing motion trajectory. We said outlier rejection can remove some of the noises but it cannot remove all. Moreover besides noise there are background motions. So to get the actual motion we had to think of another way. In a video sequence there works many parametric transformation. To get the motion about the object we should take into account those parameters. The process of estimating the transform parameters is called global motion estimation (GME) [14].This a powerful tool .There are many GME algorithms but we are interested in the algorithms where parametric transformation are being considered. Since the transformation are not linear and the equations are so complex to calculate and time consuming so a simple parametric equation is required.

A number of featureless GME algorithms have been proposed in the past, and generally they can be categorized into direct and indirect methods. Direct GME methods are pixel-based and try to minimize the prediction errors in the pixel domain. Indirect GME methods contain two stages, and GME is performed at the second stage based on the motion vectors resulted from the first motion estimation stage [14]. For the estimation of the perspective model parameters, direct GME methods in the pixel domain [22], [23] are computationally expensive due to the

iterative processes in the nonlinear estimation and the number of pixels involved, which limit their applicability in real applications. Two-step methods consisting of a local motion estimation followed by a GME were proposed [24], [25], but they were not general enough to handle perspective motions.

The global motion estimation from the coarsely sampled motion vector field uses eight parameters to compute the transformation of a pixel's position. The parameters mainly came from the concept of four transformation models. They are the translational model, geometric, affine and the perspective model. The parameters are being tuned by the neighborhood pixel's information. The equations [14] are given below:

$$\begin{aligned} x' &= f_x(x, y|m) = \frac{m_0 x + m_1 y + m_2}{m_6 x + m_6 y + 1} \\ y' &= f_y(x, y|m) = \frac{m_3 x + m_4 y + m_5}{m_6 x + m_6 y + 1} \end{aligned} \tag{2.7}$$

Here m_i are the parameters and the new position of any pixel is (x', y') which was at (x, y) position previously. So from the motion vectors it gives only one motion which represent the final motion of the object.

2.7 Examples of some trackers

We are now going to discuss some state-of-the-art trackers such as - TLD, BoostT and SemiT. Their short description is given below.

2.7.1 TLD (Tracking Learning Detection)

TLD [8] [9] [10] is a long time online tracker. It estimates the object location, scale changes, how the object looks and what is not that object by scanning every frame. TLD uses the adaptive short term tracker on Lucas-Kanade [2] method. Then it models the appearance in an unsupervised manner based on two events called growing and pruning events. These events correct each other. It uses Forward-backward error detection as outlier rejection method. The online model

is represented by a set of 15x15 intensity normalized patches. Distance between two patches is defined using normalized cross-correlation. Object Detector is based on new features that we call 2bit Binary Patterns (2bitBP). These features measure gradient orientation within a certain area, quantize it and output four possible codes.

2.7.2 Online Boosting Tracker

Online boosting tracker [4] uses the boosting algorithm, discrete AdaBoost for classification. Boosting is strongly related to SVM. Online boosting tracker uses boosting for feature selection. Boosting is a general method for improving the accuracy of any given learning algorithm. This is done by combining N hypotheses which have been generated by repeating training with different subsets of training data. Boosting transforms a weak learning algorithm into a strong one. The on-line algorithm requires that the number of weak classifiers is fixed at the beginning. In the off-line case all samples are used to update (and select) one weak classifier, whereas in the on-line case one sample is used to update all weak classifiers and the corresponding voting weight.

2.7.3 Semi-supervised Boosting Tracker

In semi-supervised tracking [5] the basic idea is to formulate tracking as binary classification problem between the foreground object, which has to be tracked, and the local background. Assuming the object has been detected in the first frame; an initial classifier is built by taking positive samples from the object and randomly chosen negative ones from the background. From time t to $t+1$ the classifier is evaluated exhaustively pixel by pixel in the local neighborhood. The confidence distribution is analyzed and in the simplest case the local maximum is considered to be the new object position. In order to robustly find the object in the next frame and thus adapt to appearance changes of the object, different lightning conditions or background changes, the classifier gets updated. A positive

update is taken for the patch where the object is most likely to be and negative updates are drawn from the local neighborhood.

Chapter 3

Proposed Method

3.1 Overall concept

Object tracking has been developed by several algorithms by several developers. No algorithm has made its position on the top of others. Ours tracking algorithm has very logical reason at its every steps. The thing first comes anyone's mind when he wants to track someone is that the motion vector. So our first step is to build the motion vector of every pixel between two consecutive frames. We have used the Optical motion flow technique to derive the motion vector. The Lucas-Kanade [2] method was followed to get the motion vectors. These motion vectors now indicate the trajectory of every pixel but they do not mean the exact motion of the object. There are noises that make the object motion confusing. So what we have to do now? The logical answer is, we should try to remove the outliers that are the noise. In this situation we introduce an outlier rejection to our program and that is the cascaded motion vector outlier rejection. It works with the three cascaded filters on the motion vectors to remove the outliers. Here it takes into account the neighborhood of every pixel. After this we have some refined motion vectors. But problem of getting the final motion vector is not finished here because there may have background motion or any kind of other motion that does not coincide with our actual object motion. Here we have used the global motion estimation tool that is very useful. We have used the global motion estimation from coarsely sampled motion technique. It also considers the neighborhood of the pixel. It uses some motion model like translational, geometric, affine, perspective. Then we get the final motion vector that actually

represents our tracked object motion.

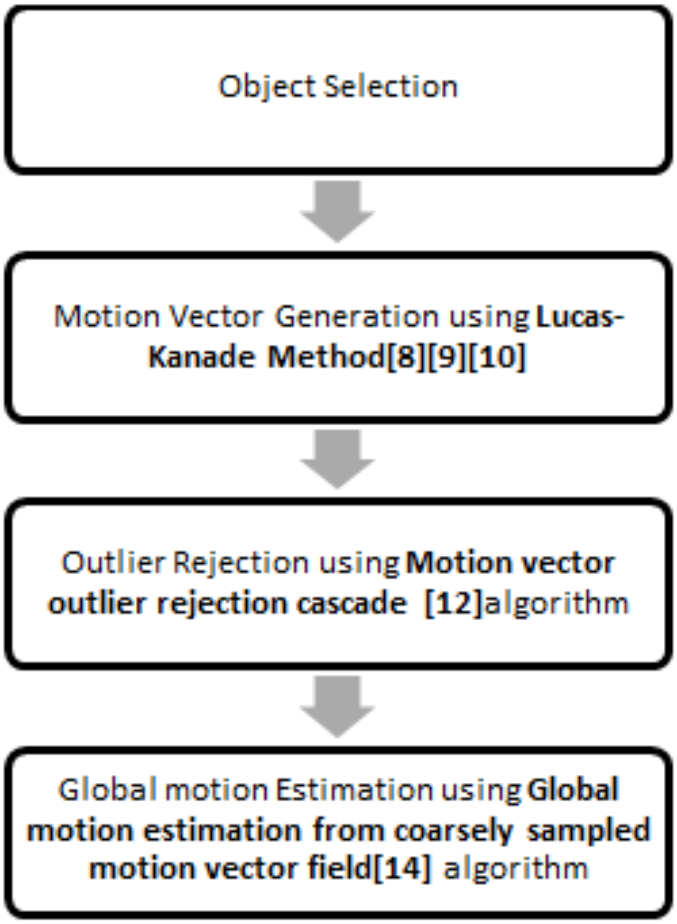


Figure 3.1: Our proposed method

3.2 Object Selection

At first the object selection happens in the system. We encompass the object with a rectangle shape boundary window. It cannot be less than 8x8 dimensions. The bigger the dimension size, the less the frame rate processing will be. The medium dimension size window brings good result for example, 50x50 or 30x30.

Figure 3.2: Object Selection

3.3 Motion Vector Generation using Lucas-Kanade Method

As Lucas-Kanade [2] assumes that the displacement of the image contents between two nearby instants (frames) is small and approximately constant within a neighbourhood of the point p under consideration. From the optical flow equation can be assumed to hold for all pixels within a window cantered at p. Namely, the local image flow (velocity) vector (V_x, V_y) must satisfy

$$\begin{aligned} &I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1) \\ &I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2) \\ &\vdots \\ &I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n) \end{aligned} \tag{3.1}$$

Where $q_1, q_2, \ldots, q_n$ are the pixels inside the window, and $I_x(q_i), I_y(q_i), I_t(q_i)$ are the partial derivatives of the image I with respect to position x, y and time t, evaluated at the point q_i and at the current time. These equations can be written in matrix form $Av = b$, where

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \\ I_x(q_n) & I_y(q_n) \end{bmatrix}, \; v = \begin{bmatrix} V_x \\ V_y \end{bmatrix} \text{ and } b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix} \tag{3.2}$$

From the Lucas-Kanade Method we find a system. This system has more equations than unknowns and thus it is usually over-determined. The Lucas-Kanade method obtains a compromise solution by the least squares principle. From $Av = b$ equation we can write,

$$\begin{aligned} A^T Av &= A^T b \text{ or} \\ v &= (A^T A)^{-1} A^T b \end{aligned} \tag{3.3}$$

Now we can write it in vector form as:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i) I_y(q_i) \\ \sum_i I_x(q_i) I_y(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i) I_t(q_i) \\ -\sum_i I_y(q_i) I_t(q_i) \end{bmatrix} \tag{3.4}$$

With the sums running from $i = 1$ to n. To generate the motion vector we have to deliver two consecutive frames, the present one and the previous one. Then the corresponding pixels in the boundary window of two frames will be computed. Here for every pixel we take a 5x5 neighborhood window and pass this to the Lucas-kanade method iteratively. Lucas-kanade method then gives us the motion vectors for every pixel.

3.4 Outlier Rejection using Motion vector outlier rejection cascade algorithm

Now we have the motion vectors at our hand. At first, we initialize outlier rejection rate at 0.5. Then we pass the motion vector to cascaded outlier rejection method. The boundary window is just sent here not the whole image.

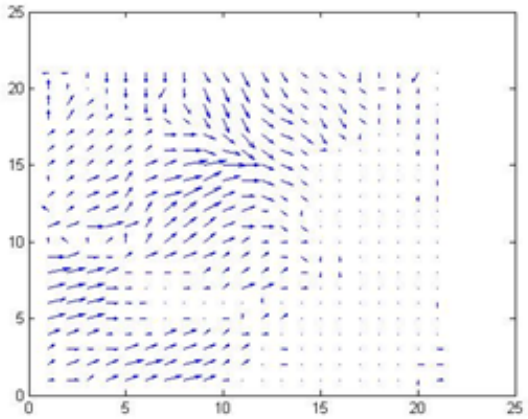


Figure 3.3: The motion vector generated after the selected object is passed to the Lucas-kanade algorithm

The cascade consists of three filters [12]. Input MV field is subject to testing in the first filter, and then the MVs declared as inliers are further tested in the second filter, and so on.

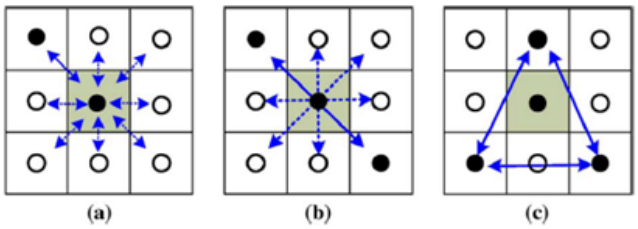


Figure 3.4: Construction of MVs in S_i^j; (a) S_i^1 (b) S_i^2 (c) S_i^3

To test each input MV, the filters in the cascade employ the following strategy. Let MV_i be the input MV to be tested in j filter, where $j \epsilon \{1, 2, 3\}$. Associated with MV_i is the set S_i^j of MVs computed from the 8-neighborhood of MV_i as shown in Fig. 3.4, where the location of is shown in gray. For filter 1, S_i^1 consists of individual MVs from the neighborhood of MV_i, as shown in Fig. 3.4(a). For filter 2, S_i^2 consists of the averages of diagonally opposite MVs from the neighborhood of MV_i, as shown in Fig. 3.4(b). Finally, for filter 3, S_i^3 consists of the averages of triangularly opposite MVs from the neighborhood of MV_i, as

shown in Fig. 3.4(c). There are at most eight MVs in S_i^1, and at most four MVs in each of S_i^2 and S_i^3.

Once S_i^j is constructed, the following conditions for each $MV_k \epsilon S_i^j$ and count how many times the following conditions are satisfied:

$$||MV_i - MV_k||/||MV_i|| < T_{mag}^j \phi(MV_i) - \phi(MV_k) < T_{ph}^j \tag{3.5}$$

where T_{mag}^j and T_{ph}^j are the thresholds for maximum relative magnitude difference, and maximum phase difference, respectively. To avoid the computation of phase $\phi(.)$,the equation above can be rewritten as $(MV_i, MV_k) > ||MV_i||.||MV_k||.cos(T_{ph}^j)$. Let N_i^j be the number of times the above conditions are satisfied. Note that, $N_i^1 < 16$ and $N_i^j < 8$ for $j \epsilon \{2, 3\}$. The weighted count is given by $WN_i^j = W_i^{j-1}.N_i^j$, where $W_i^j = exp(-(WN_{max}^j - WN_i^j))$, $WN_{max}^j = max_i WN_i^j$ and $W_i^0 = 1$ for all i . The weight W_i^j is a measure of how similar is MV_i to vectors in S_i^j.

The MV goes through three filters and at the end the top 50% motion vectors are kept. This rejects any noisy motion vector. The algorithm for the Motion Vector Outlier Rejection Cascade is given below:

Algorithm 1: Motion Vector Outlier Rejection Cascade

MV is the motion vector generated from the Lucas-Kanade method; p is the Outlier Rejection rate

1)**Initialize** all MV s as inliers and **Set** $p := 0.5$

2)**Set** $j := 1$

3)**For** $j \leq 3$ **do**

a)For each inlier MV_i , find the weighted count WN_j^i .

b)Sort MVs in descending order of their WN_j^i.

c)**Set** p percentage of MVs at the bottom of the sorted list as outliers.

4)**End for**

5)**Return** MV

After the MV has gone through the outlier rejection method the MV without the noisy output looks like Fig 3.5.

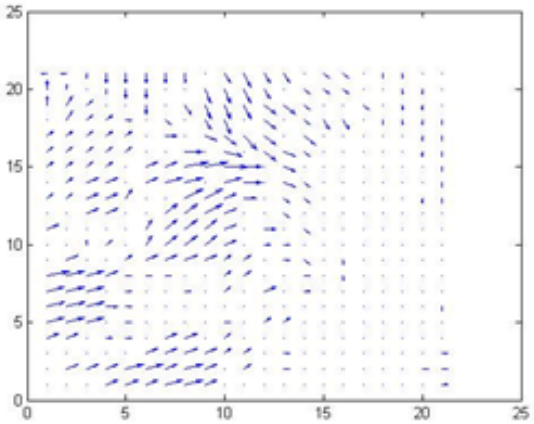


Figure 3.5: MV after the outlier rejection method

3.5 Global motion Estimation

The refined motion vector is passed to this stage. We use the algorithm of global motion estimation from coarsely sampled motion vector field. From the boundary window we get the motion vector that represents the motion of the object. This final motion vector is then set for the whole boundary window and it moves toward that direction. The GME we are going to use have considered the models given in Fig 3.6 and Fig 3.7.

Here x' and y' are the new cordinates of the previous position x and y of the tracked object . We see from the above below that if we can compute the eight parameters then we can cover all the transformations . So the equations goes like :

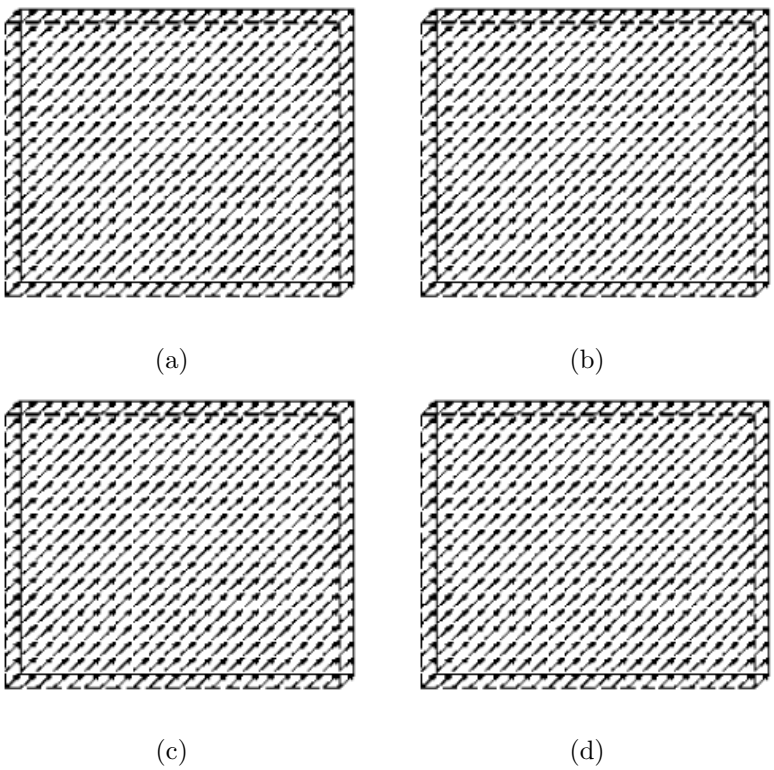
$$\begin{aligned} x' &= f_x(x, y|m) = \frac{m_0 x + m_1 y + m_2}{m_6 x + m_6 y + 1} \\ y' &= f_y(x, y|m) = \frac{m_3 x + m_4 y + m_5}{m_6 x + m_6 y + 1} \end{aligned} \tag{3.6}$$

Here m_0 to m_7 are the eight parameters we were talking about. These equations are then computed for the differentiations. Thus the parameters are tuned

DIFFERENT PARAMETRIC GLOBAL MOTION MODELS

Motion Model	Number of Parameters/Reference Points	Transform
Translational M_2	2/1	$x' = x + c$ $y' = y + f$
Geometric M_4	4/2	$x' = ax + by + c$ $y' = -by + ay + f$
Affine M_6	6/3	$x' = ax + by + c$ $y' = dy + ey + f$
Perspective M_8	8/4	$x' = \frac{ax + by + c}{px + qy + 1}$ $y' = \frac{dx + ey + f}{px + qy + 1}$

Figure 3.6: Global Motion Models



(a) (b)

(c) (d)

Figure 3.7: Typical motion vector fields for (a) translational, (b) geometric, (c) affine, and (d) perspective models.

and we get the desired.

The algorithm for the GME is given below:

Algorithm 2: Global Motion Estimation From Coarsely Sampled Motion Vector Field

m is the motion parameter ; A is the Hessian Matrix ; b is the gradient vector.

1)**Set** $T := 99999$ and initialize m_k

2)**For** $i \leq 32$ **OR** $m \geq 0.001$ **do**

a)**For** each pixel (x_i, y_i) **do**,

i)compute (x_i', y_i') ;

ii)compute e_i ;

iii)**If** $i = 1$ **Then** compute histogram of e_i

iv)**End If**

v)**If** $e_i < T$ **Then**

A)compute derivative of e_i with respect to m_k

B)add the pixel's contribution to A and b.

vi)**End If**

b)**End for**

c)Solve $A\Delta m = b$ and update $m(t+1) = m(t) + \Delta m$

d)**If** $i = 1$ **Then** compute T to exclude the top 10% of the histogram of e_i.

e)**End If**

3)**End for**

4)**Return** m

The GME gives a single motion from the motion vectors and according to that the bounding box moves to that position. This steps are repeated in every frame sequence and the object is tracked in that way.

Chapter 4

Experimental Analysis and Result Discussion

4.1 Used software and tools

The software we have used are Matlab R2010a, Microsoft Visual Studio 2010, Open CV. The experiments were done in the Intel Pentium Dual CPU E2200 @2.20 GHz, Ram 2.00 GB, 32-bit Operating System and with a HD Webcam.

4.2 Datasets

The datasets Hallway , CarPhone , Salesman are the video sequences taken from http:// media.xiph.org/video/derf . The video sequences are in QCIF format. The dataset named Face was taken by a HD webcam . Table 4.1 contains the details about the datasets.

Table 4.1: Summary of Datasets

Video Sequence name	Dimension	Number of frames	Dimension of bounding box
Hallway	177×144	280	30×30
CarPhone	177×144	380	52×52
Salesman	177×144	449	24×24
Faces	320×240	160	52×52

4.3 Tracking Performance

In our tracker we tracked an object by generating the motion vector and then rejecting the outliers using cascaded outlier rejection method. By using the outlier rejection method our tracker has been able to track the object more accurately. We have discarded 50

In the Hallway video sequence we took a 30x30 window to track the man walking in the hallway. In the video sequence we tried tracking the man by discarding the outliers and also keeping them.Fig. 4.1 shows the error in both cases.

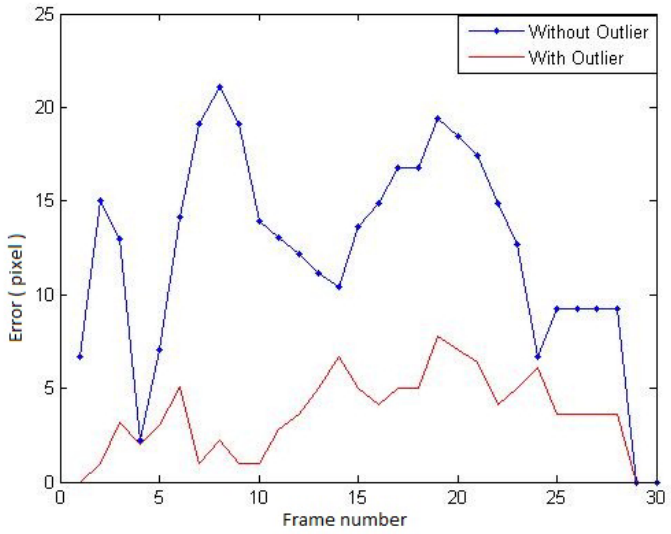


Figure 4.1: Error rate comparison using and not using a outlier method in Hallway video sequence

As we can see from the Figure the error rate is higher when an outlier rejection method is not use.

In the CarPhone video sequence we used a 52x52 window to track the face of the man in the car. Fig. 4.2 shows the error of using the cascaded method.

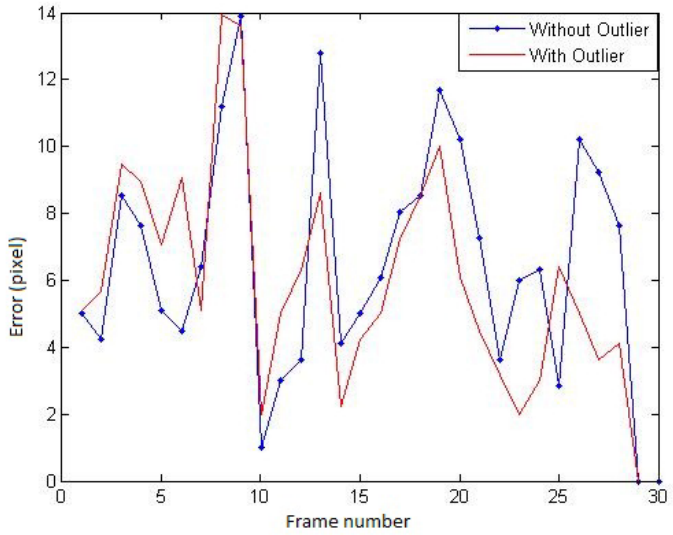


Figure 4.2: Error rate comparison using and not using a outlier method in Car-Phone video sequence

We can see from the figure that in this case the error is almost same because due to the slow changes in the motion. But still the tracker with the outlier rejection method gives a slightly better performance.

The Salesman video sequence has a fast object moving in it. We used a 24x24 window to track the object. Fig. 4.3 shows the error of not using the outlier rejection method:

We took the video sequence that we have created using a HD webcame and tried to track the face using our tracker and it giva us the squence in Fig 4.4.

From Fig. 4.4 we can see that our tracker tracks the selected place which is a 52x52 window sized almost accurately even when the images are not good qualities.

Some other images showing tracking of the selected object throughout the video sequences in Fig. 4.5, Fig. 4.6 and Fig. 4.7.

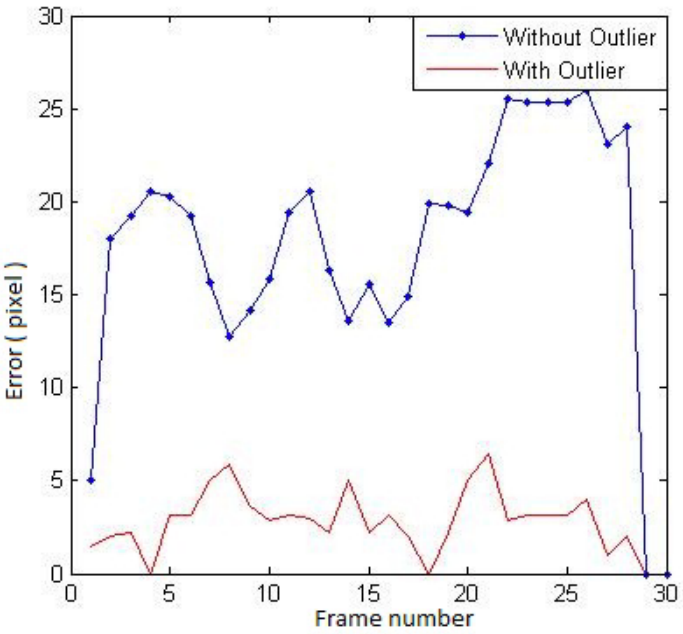
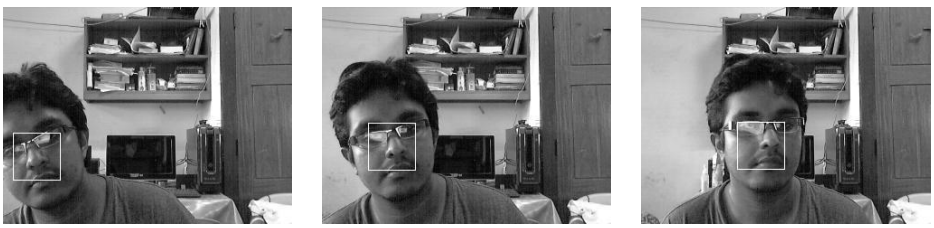


Figure 4.3: Error rate comparison using and not using a outlier method in Salesman video sequence



(a) (b) (c)

Figure 4.4: Trajectory on the video sequence Face

From the given video sequences we can see that our tracker tracks the selected object in most case. From the salesman video sequence we can tell that out tracker in rotation invariant and is able to track a fast moving object. Our tracker is also able to track a face very accurately throughout the video sequence.

4.4 Comparative Analysis

In this section we are going to compare our tracker with some state-of-the-art tracker TLD, Boost tracker and SemiBoost tracker. We have taken the same

(a) (b) (c)

Figure 4.5: Tracking the face in the CarPhone video sequence

(a) (b) (c)

Figure 4.6: Tracking the man in the Hallway video sequence

(a) (b) (c)

Figure 4.7: Tracking the object in the man's hand in Salesman video sequence

object with the same size of the bounding box and tried tracking the selected object. We than compared their error with our tracker. In the Salesman video sequence we get the graph in Fig. 4.8.

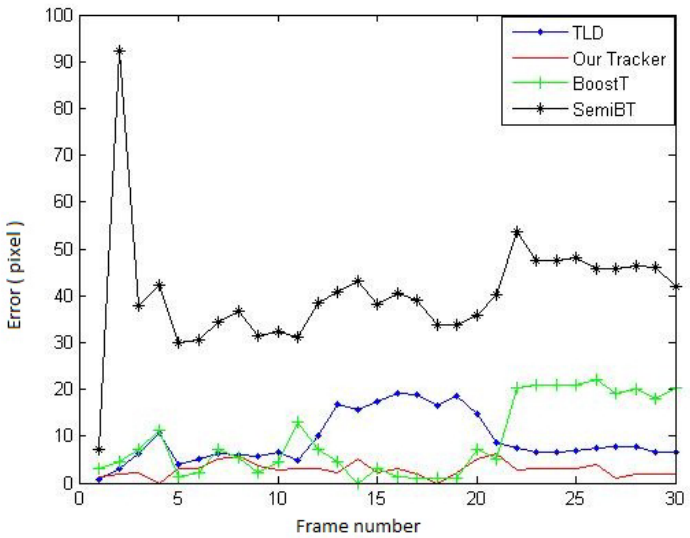


Figure 4.8: Comparison with other trackers in the Salesman video sequence

It is clear from the graph that our tracker gives the lowest error. As we can see in SemiBoost tracker gives a really high error. But TLD and Boost tracker gives a less error than the SemiBoost tracker. But our tracker is able to track the object when TLD and Boost tracker gives a high error. So in case of rotation invariant and fast motion our tracker gives the best output.

In the Hallway video sequence we get the graph in Fig. 4.9

Here we can see that Boost tracker and SemiBoost tracker gives high error. But our tracker and TLD gives the best performance. In some frames our tracker gives better output than TLD. But as TLD has better zooming capability it works better in other frames.

In the CarPhone video sequence we get the graph in Fig. 4.10.

In this video sequence although all the trackers performs really good at first but semiBoost tracker starts failing to track the object in the middle. As there is lot of translational motion of the face in this video sequence that's why all the trackers performs well. But still we can see our tracker gives the best output than

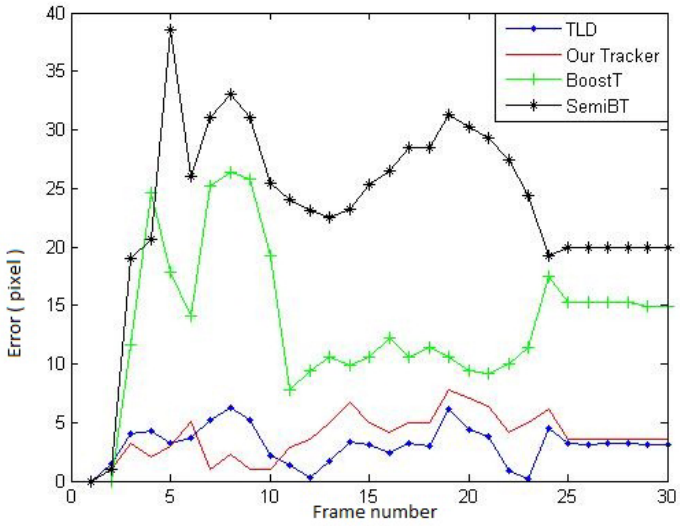


Figure 4.9: Comparison with other trackers in the Hallway video sequence

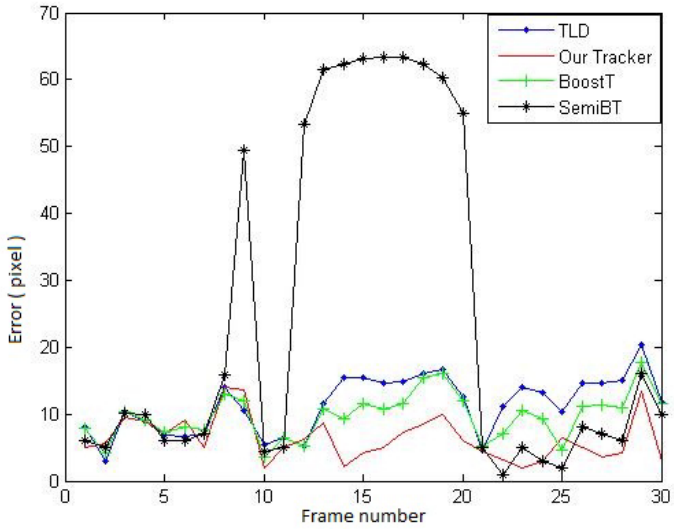


Figure 4.10: Comparison with other trackers in the CarPhone video sequence

all the other trackers.

Below we have given the comparison of the Frame rate of all the video sequence and also their average error rate in Table 4.2 and Table 4.3.

From the table we can see that our tracking is showing small error rate than the others. Though the frame rate is not more in all sequences but this doesn't

Table 4.2: Frame rate(FPS)

Video Sequence	Our Tracker	TLD	Boost Tracker	SemiBoost Tracker
Salesman	2.8997	9.2933	9.2100	40.4335
CarPhone	6.3802	11.3270	9.7082	24.4340
Hallway	3.8295	3.0956	13.5439	23.6292

Table 4.3: Average Error Rate(Pixel)

Video Sequence	Our Tracker	TLD	Boost Tracker	SemiBoost Tracker
Salesman	10.51	6.92	15.87	9.64
CarPhone	3.75	7.61	10.64	7.14
Hallway	7.9	13.5	15.87	9.64

hamper the accuracy of our tracker. In the salesman video the best was done by our tracker. In this video the object was having fast movement and rotation. In the carPhone video sequence the frame rate is low but it gives lowest average error rate than the others. In the Hallway video sequence TLD shows the best accuracy because it has zoom-in, zoom-out capabilities. But our tracker still shows the closer error rate like TLD which is second best in this case.

Chapter 5

Conclusion

In this paper, we developed an object tracking system. Optical flow based equation were used to generate the motion vector at the first stage. Then there were seen noises whose motion vectors were spreading errors to the next stages. The cascaded outlier rejection method was used then to find the noiseless set of motion vectors. Three cascaded filter is used to calculate this and they take into account the neighborhood pixels information. The final motion vector of the object between two consecutive frames was determined by the global motion estimation method. It uses the transformational model with parametric equations.

The motivation behind every stage of this system was taken by very simple logic but as we know, though the concept is simple, the implementation is not always easy generally. In our case we have succeeded to implement it with very good performance. Comparing with the other state of the art algorithms it is performing very well in accuracy and in fast movement also. Sometimes it is showing better performance than those trackers.

In terms of future work, we look forward to developing a long term tracker with this short term tracker as the basis method. Some new challenges will also be tried to solve with this system for example, the zoom invariance, partial occlusion, and full occlusion. Though these facilities are supposed to be time consuming we cannot afford to consume that much time because we have to track in fast camera motion and as a real time tracker it shouldn't.

Bibliography

[1] Wang, Q., Chen, F., Xu, W., Yang, M.H.: An experimental comparison of online object tracking algorithms. Proceedings of *SPIE: Image and Signal Processing Track (2011)*

[2] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in Proceeding of *International Joint Conference on Artificial Intelligence*, pp. 674-679, 1981.

[3] A. Azarbayejani and A. Pentland, "Recursive estimation of motion, structure, and focal length," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(6), pp. 562-575, 1995.

[4] H. Grabner and H. Bischof, "On-line boosting and vision," in Proceedings of emphIEEE Conference on Computer Vision and Pattern Recognition, pp. 260-267, 2006.

[5] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in Proceedings of *European Conference on Computer Vision*, pp. 234-247, 2008.

[6] S. Stalder, H. Grabner, and L. Van Gool, "Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition," in Proceedings of *IEEE Workshop on Online Learning for Computer Vision*, 2009.

[7] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in Proceedings of *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 983-990, 2009.

[8] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-n learning: Bootstrapping binary classifiers by structural constraints," in Proceedings of *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 49-56, 2010.

[9] Z. Kalal, J. Matas, and K. Mikolajczyk. "Online learning of robust object detectors during unstable tracking." *OLCV*, 2009.

[10] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-Backward Error: Automatic Detection of Tracking Failures," *International Conference on Pattern Recognition*, 2010.

[11] J. Barron and N.A.Thacker, "Tutorial: Computing 2D and 3D Optical Flow," Tina Memo 2004-012

[12] Y. Chen, and I. Bajic, "Motion vector outlier rejection cascade for global motion estimation," *IEEE Signal Processing Letters*, vol. 17, no. 2, 2010, pp. 197-200.

[13] Robert C. Bolles Martin A. Fischler, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *ACM Communications*, vol. 24(6), pp. 381-395, 1981.

[14] Y. Su, M.-T. Sun, and V. Hsu, "Global motion estimation from coarsely sampled motion vector field and the applications," *IEEE Trans. Circuits Syst. Video Technol*, vol. 15, no. 2, pp. 232-242, Feb. 2005.

[15] S. Tubaro and S. Rocca, "Motion field estimators and their application to image interpolation," in *Motion Analysis and Image Sequence Processing*,M. I. Sezan and R. L. Lagendijk, Eds. Norwell, MA: Kluwer, 1993, pp. 153-187.

[16] D. Farin, "Automatic Video Segmentation Employing Object/Camera Modeling Techniques," Ph.D. Thesis, Technische Univ. Eindhoven, Eindhoven, Netherlands, 2005.

[17] I. Haritaoglu, D. Harwood, and L. Davis, "W4s: A real-time system for detecting and tracking people," in Proceedingsof *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 962-968, 1998.

[18] M. de La Gorce, N. Paragios, and D. Fleet, "Model-based hand tracking with texture, shading and self-occlusions," in Proceedings of *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[19] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(5), pp. 694-711, 2006.

[20] M. Kim, S. Kumar, V. Pavlovic, and H. Rowley, "Face tracking and recognition with visual constraints in real-world videos," in Proceedings of *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, 2008.

[21] X. Zhou, D. Comaniciu, and A. Gupta, "An information fusion framework for robust shape tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(1), pp. 115-129, 2005.

[22] "MPEG-4 Video Verification Model version 18.0,", ISO/IEC JTC1/SC29/WG11, 2001.

[23] F. Dufaux and J. Konrad, "Efficient, robust, and fast global motion estimation for video coding," *IEEE Trans. Image Process*, vol. 9, no. 3, pp. 497-501, Mar. 2000.

[24] Y. T. Tse and R. L. Baker, "Global zoom/pan estimation and compensation for video compression," in Proc. *ICASSP'91*, Toronto, ON, Canada, May 1991, pp. 2725-2728.

[25] H. Jozawa, K. Kamikura, A. Sagata, H. Kotera, and H.Watanabe, "Twostagemotion compensation using adaptive global MC and local affine MC," *IEEE Trans. Circuits Syst.* Video Technol., vol. 7, no. 2, pp. 75-85, Feb. 1997.

Chapter 6

Appendix

6.1 Snapshot Of The Code

The core code for our proposed tracker is given below:

```
vid = videoinput('winvideo',1,'RGB24_320x240');
triggerconfig(vid,'manual');
set(vid,'FramesPerTrigger',1);
set(vid,'TriggerRepeat', Inf);
start(vid);
preview(vid);
%im = zeros(640,480,3);
trigger(vid);
im1= getdata(vid,1);
tempimg1 = rgb2gray(im1);
x = 160;
y = 120;
a = 0;
b = 0;
winsize = 52;
blkSiz = 16;
outlier = 0.5;
MAXITER_CAS = 1; % Maximum iterations for GD-GME with the cascade
MAXITER_ORI = 6; % Maximum iterations for plain GD-GME
MAXITER_LSS = 3; % Maximum iterations for LSS-ME
pC = 52;
pR = 52;
```

```
23 bC=pC/blkSiz; % number of column in blocks
24 bR=pR/blkSiz; % number of row in blocks
25 [coorX,coorY]=ndgrid(1:49,1:49);
26
27 % the parameters for the cascade
28 GM_TRAN = 1;  % translational model
29 GM_ISOT = 2;  % isotripic model
30 GM_AFFI = 3;  % affine model
31 GM_PERS = 4;  % perspective model
32 HALFPIX=2; % Half pixel
33 iniMM=[];
34 tic;
35 for i = 1:100
36     trigger(vid);
37     im2= getdata(vid,1);
38     tempimg2 = rgb2gray(im2);
39     [m n] = size(tempimg2);
40     temp1 = extract(tempimg1,x,y,winsize/2,160,120);
41     temp2 = extract(tempimg2,x,y,winsize/2,160,120);
42     [px py] = lk2(temp1,temp2); %Implementing Lucas Kanade
43     iMap = MVRemCas(px, py, outlier, blkSiz); %Removing the outliers
44     m = mvGME_NR_test(GM_TRAN, px(:), py(:), iMap(:),coorX(:) ...
           ,coorY(:), 6, outlier, iniMM); %Performing Global Motion ...
           Estimation
45     a = ( m(1,1)*x  +   m(1,3)*2 )  ;
46     b = (m(1,5)*y + m(1,6)*2);
47     x = round(a);
48     y = round(b);
49     temp3 = bb(tempimg2,winsize/2,x,y);
50     tempimg1 = tempimg2;
51     imshow(temp3);
52 end;
53 toc;
54 stop(vid);
55 delete(vid);
56 clear vid;
```