East West University

Mobile Object Tracking in a Video using Kalman Filter

Submitted By:

Samira Akter	ID: 2013-2-60-006
Sadia Jannath Zuthy	ID: 2013-2-60-011
Md. Mahamudun Hassan	ID: 2013-2-65-013

Supervised by:

Dr. Anisur Rahman

Assistant Professor

Department of Computer Science and Engineering

East West University

A Project Submitted in Partial Fulfillment of the requirements for the Degree of Bachelors of Science in Computer Science and Engineering to the Department of Computer Science and Engineering.

East West University

Dhaka, Bangladesh

August, 2017

Abstract

Object tracking in a video is the problem of estimating the positions and other related information regarding moving objects in video. Object tracking is a very important task in the field of security automation surveillance systems. For detecting and tracking the moving objects, surveillance system are used. First stage of the system is detecting the moving objects in the video. Second stage of the system is tracking the detected object. Here, detection of the moving object is done by using a simple background subtraction and tracking of moving objects is done by using Kalman filter. The algorithm is applied successfully on standard video datasets. The videos used here for testing have been taken at indoor as well as outdoor environment having moderate to complex environments. Kalman filter tracks an object by assuming the initial state and estimating noise covariance. It provides an efficient method for calculating the state estimation process. An experimental result which came from different moving object video samples shows a very good result. This filter is intended to be robust without being programmed with all environment specific rules.

Declaration

We hereby declare that, this project was done under CSE497 and has not been submitted elsewhere for requirement of any degree or diploma or for any purpose except for publication.

____

Samira Akter

ID: 2013-2-60-006

Department of Computer Science and Engineering

East West University

____

Sadia Jannath Zuthy

ID: 2013-2-60-011

Department of Computer Science and Engineering

East West University

____

Md. Mahamudun Hassan

ID: 2013-2-65-013

Department of Computer Science and Engineering

East West University

Letter of Acceptance

We hereby declare that thesis is from the student's own work and best effort of us, and all other sources of information used have been acknowledged. This thesis has been submitted with our approval.

Supervisor

____

Dr. Anisur Rahman
Assistant professor
Department of Computer Science and
Engineering
East West University, Dhaka

Chairperson

____

Dr. Md. Mozammel Huq Azad Khan
Professor and Chairperson
Department of Computer Science and
Engineering
East West University, Dhaka

Acknowledgement

Firstly our most heartfelt gratitude goes to our beloved parents for their endless support, continuous inspiration, great contribution and perfect guidance from the beginning to end.

We owe our thankfulness to our supervisor Dr. Anisur Rahaman for his skilled, almost direction, encouragement and care to prepare myself.

Our sincere gratefulness for the faculty of Computer Science and Engineering whose friendly attitude and enthusiastic support that has given me for four years.

We are very grateful for the motivation and stimulation from my good friends and seniors.

We also thank the researchers for their works that help me to learn and implement the Determination of average acoustic speed in underwater Environment.

Table of Content

List of Figures

Chapter 1

Introduction

Object detection and tracking in a video is an active research topic in computer vision that tries to detect, recognize and track objects in a sequence of images in video and also makes an attempt to understand and describe object behavior in video by replacing the old traditional methods of monitoring cameras by human operators. Object detection and tracking in a video is an important and challenging task in many practical applications such as surveillance, vehicle navigation and autonomous robot navigation. Object detection is nothing but locating objects in the frame of a video sequence. The availability of high speed computers, high quality video cameras, and the need for automated video analysis has led to a great interest in object tracking algorithms.

There are three key steps in video analysis: detection of interesting moving objects, tracking of such objects from frame to frame, and analysis of object tracks to recognize their behavior.

Video tracking can be defined as an action which can estimate the trajectory of an object in the image plane as it moves within a scene.

1.1 Objective

The aim of the project is to detect and track the moving object in a video using Kalman filter. In recent times, automatic security surveillance systems are an active research area due to an increasing demand for such systems in public places such as airports, underground stations and mass events [1]. The main objective of this algorithm is to assist human operators in analyzing the bulky video data so that work load becomes less for humans. The goal of the work in this thesis is:

1) To set up a system for automatic detection and tracking of moving objects in a video using stationary camera, which may serve as a foundation for higher level reasoning tasks and applications.

2) To make improvements in commonly used algorithms.

Therefore, the main objectives are:

- To analyze detection algorithm to detect the objects.
- To analyze some tracking method for tracking the objects.

1.2 Significance

Importance of object tracking from videos is increasing day by day and it's a very challenging task in many practical application like surveillance, vehicle navigation and automated robot navigation. We use tracking for

- To detect moving object
- Know the motion and speed of vehicles
- To count the objects
- To see the path of movement
- To know the future step of the object
- To see the path of movement

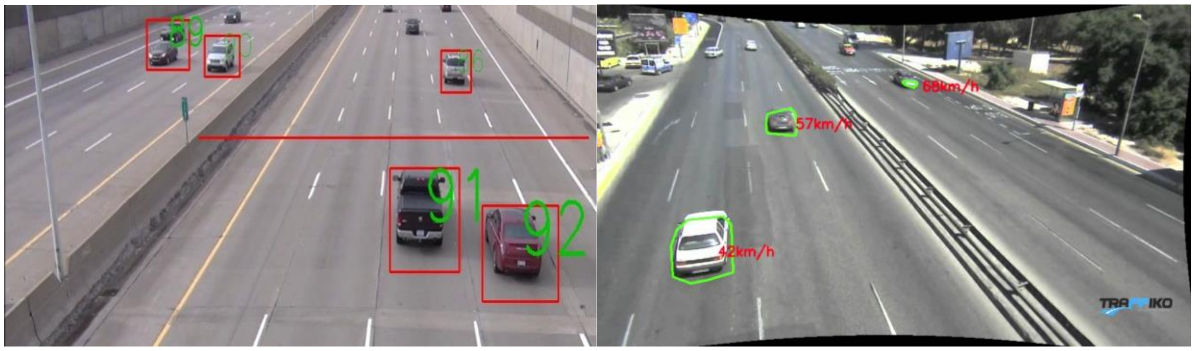

Figure1.1: significance of object tracking

For this reason automated video analysis has generated a great deal of interested in object tracking algorithms.

The Kalman filter has numerous applications in technology now a days. The Kalman filter is a widely applied concept in time series analysis used in fields such as signal processing and econometrics. Kalman filters also are one of the main topics in the field of robotic motion planning and control, and they are sometimes included in trajectory optimization. The Kalman filter also works for modeling the central nervous system's control of movement. Due to the time delay between issuing motor commands and receiving sensory feedback, usage of the Kalman filter supports the realistic model for making estimates of the current state of the motor system and issuing updated commands [2].

1.3 Methodology

To detect object from a video using Kalman filter there are mainly three simple steps which is necessary to follow step by step. First we use Background Subtraction (image processing) to detect the object from the video and for this we need to convert the video into frames. Secondly we use Gaussian filter to smooth the moving object to get the predicted values from the frames one after another. And thirdly we apply Kalman filter to find the accurate position of the moving object.

1.3.1 Background Subtraction

Background subtraction is a widely used approach for detecting moving objects in videos from static cameras. The rationale in the approach is that of detecting the moving objects from the difference between the current frame and a reference frame, often called the “background image”, or “background model”. As a baric, the background image must be a representation of the scene with no moving objects and must be kept regularly updated so as to adapt to the varying luminaire conditions and geometry settings. More complex models have extended the concept of “background subtraction” beyond its literal meaning [3].

In background subtraction we uses the difference of the current image and the background image to detect the motion region and it is generally able to provide data included information about object. The background image is subtracted from the current video frame. And if the pixel difference is larger than the set threshold value T, then it determines that the pixels of the moving object, or as a background pixels [4].

Here, for our project work, to start object detection, we select the video and convert the video into frames using Free Studio software which gives us frames of the video maintaining sequence and for this sequence images we can detect the moving one after another frames which is just like the video. When we get the frames, we take some frames for background substation. From the background, it will be easy to detect the moving object.

1.3.2 Gaussian Filter

By background subtraction we get the moving object and after that we apply Gaussian filter to get smoother image of the object which we actually want to track. On the other hand it can be said by using Gaussian filter we see the moving object more clearly in different colors which make more understanding to detect the object that we want to track. From this we also get a value of position of the object which is the accurate value of the tracking object.

1.3.3 Kalman Filter

Kalman filter technique is used to estimate the state of a linear system where state is assumed to be distributed by a Gaussian. In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem. Object tracking is performed by predicting the object's position from the previous information and verifying the existence of the object at the predicted position. Secondly, the observed likelihood function and motion model must be learnt by some sample of image sequences before tracking is performed [5].

Tracking is the process of locating moving objects over time in each frame of videos. A Kalman filter is not a filter. It is an optimal estimator i.e. infers parameters of interest from indirect, inaccurate and uncertain observations. It process the new measurements as they arrive regularly i.e. Kalman filter is recursive. The word 'filter' is used because it is the process of finding the 'best estimate' from noisy data for 'filtering out' the noise. It is an estimate obtained through combining both prediction and measurement. The Kalman filter consists of two stages: Time update (prediction) and the measurements update (correction). The time update equations projecting forward (in time) the current state and error co-variance estimates for obtaining the priori estimates for the next time step. The time Update equation is called predictor equation. The measurement update equations incorporates a new measurement into the a priori estimate to obtain an improved a posterior estimate. The measurement update is called corrector equation [1].

Chapter 2

Background and Literature Reviews

2.1 Related Work

Now a days tracking is very popular topic and many people work on it to track object more clearly from a distance and for this reason Kalman filter is also very well known to most of the people.

In April 2013, Hitesh A Patel, Darshak A thakure, published a paper on "Moving Object Tracking Using Kalman Filter". For detecting the moving object, they used background subtraction method and tracking of single moving object had been done using Kalman filter. The algorithm was applied successfully on standard surveillance video datasets of CAVIAR and PETS. But in that paper, for detecting the object they did not use background subtraction method with thresholding process. They only applied background subtraction method by averaging all the frames pixels. Also they did not use any filtering method like Gaussian Filter, Median Filter etc. for smoothing the frame image. The following figure shows how they work on their research [5].

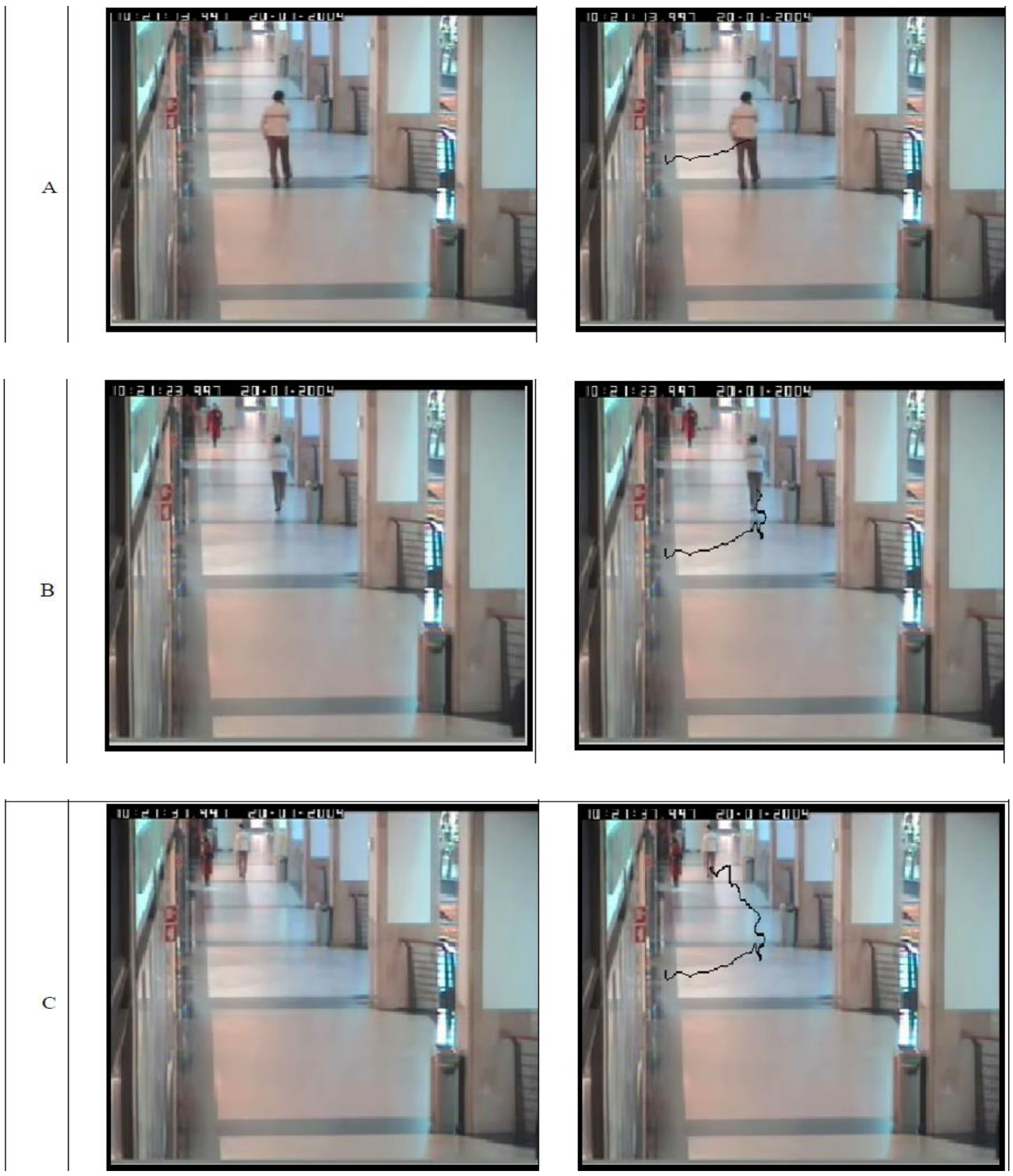

Figure 2.1: Object Tracking on CAVIAR dataset video using Kalman filter. Left column indicates original frame and right column indicates tracked output [A] 50th frame [B] 300th frame [C] 650th frame [5].

In 2015,Pravin A. Dhulekar, Vaishnavi D. Hire, Mandar S. Agnihotripublished another paper based on "Moving Object Tracking Using Kalman Filter". The paper presented the moving object tracking using Kalman filter and referenced of Background elimination. In this method they used fixed camera for video capturing and first frame of video was directly considered as Reference background frame and that frame is subtract from current frame to detect the moving object and set the threshold T value. As if the pixel difference is greater than the set threshold value T, then it determines that the pixels from travelling object, else the background image pixels. But this specified threshold suitable only for an ideal condition is not suitable for complex environmental condition with light effect changes. In that research, they also did not use any filtering method that could reduce the noise from image.

On 5th October 2016, THOTA VINOD RAJA, M. TIRUPATHAMMA, made a research paper on "Object Detection and Tracking in video using Kalman Filter where is showed to track an object using Kalman filter with median filter and threshold process. They used segmentation algorithm to detect the object.

The following figure showed who they worked with background subtraction method with thresholding process with median filter method and with Kalman filter algorithm [1].

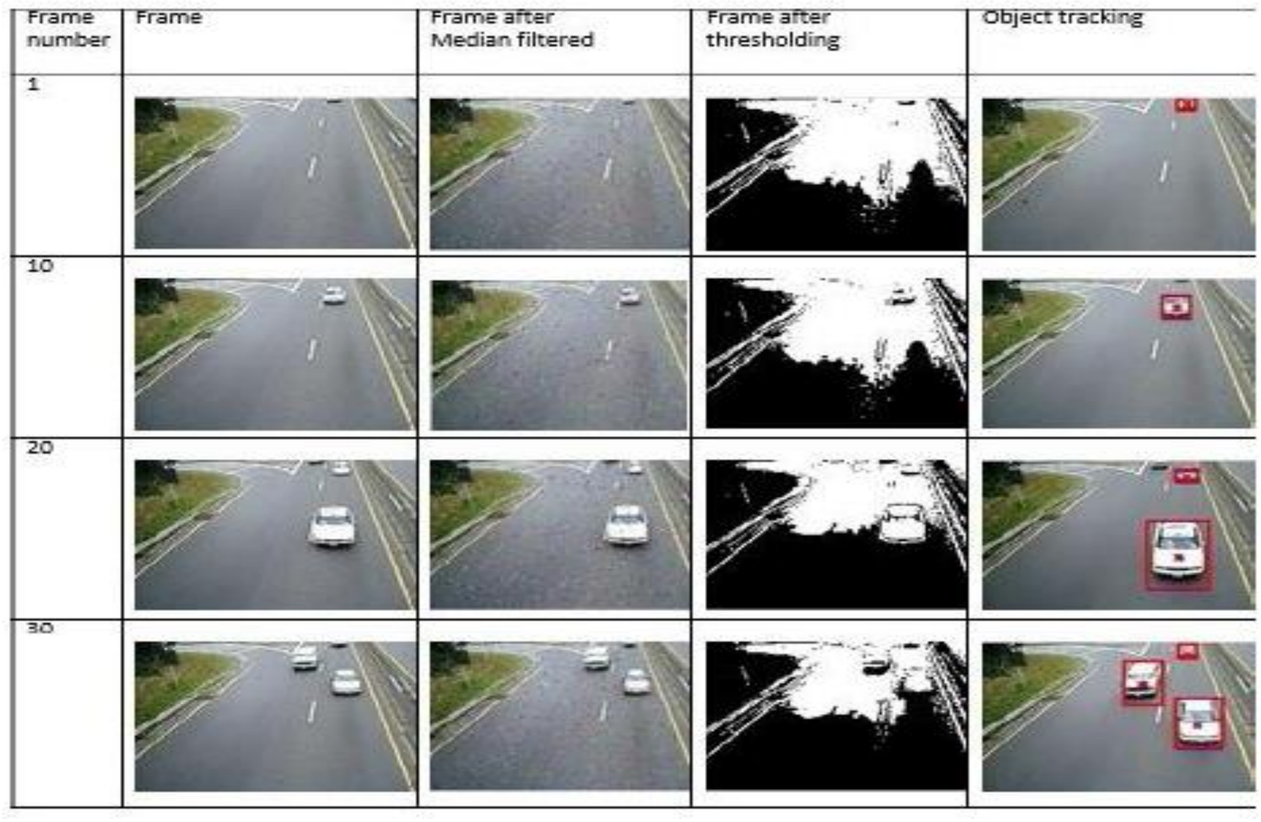

Figure 2.2: object detection and tracking in video using Kalman filter [1].

There are many other works in where only Median filter is used to detect the object with Kalman filter. But in our research, we use background subtraction method with thresholding process by using histogram idea. For this we able to know where the exact position of the object is. And also we use Gaussian Filtering process for smoothing the image so that we could see our interesting mobile object more precisely. And finally, we use the almighty Kalman Filter algorithm for tracking our mobile object.

2.2 Detection Algorithm

For complete a work we have to do many works. In this project we do something like this. To track a moving object by using Kalman filter we use video conversion to make a video frames, we use background subtraction to detect the object where actually it is situated, we use Gaussian filter to make the object more clear in the image and after that we apply Kalman filter to detect the object actual position. After that we also show the path of object where it used to move by using mean values of the object moving.

2.2.1 Video Conversion

At the 1st of the work we have to select the video from which we will track a moving object. Then we use 'Free Studio' software to convert the video into frames. We use this software because by using this we get the frames sequence number correctly.

First we open the software and select 'Free video to jpg '. Then we select add files for uploading the video in this software. We also can choose how many frames will be created after a limited time. Then we select convert button and after a few seconds the video will convert into frames and shown that the process is completed. We also can change the folder location where the frames saved or it chose it default.

2.2.2 Image Processing

Image is a collection of some points in 'X' and 'Y' co-ordinates so every points have relations with each other's like there is image of a pen on a table. Then the table's value is 1 in all place but where the pen there is value is greater than 1. So image processing goal is to find features of the different values of the image [6].

So in image processing we do three works

1. Average background subtraction.
2. Noise reduction via image smoothing using 2D Gaussian filter.

3. Threshold and point detection in binary image.

Background subtraction (BS) is a common and widely used technique for generating a foreground mask (namely, a binary image containing the pixels belonging to moving objects in the scene) by using static cameras [7].

As the name suggests, BS calculates the foreground mask performing a subtraction between the current frame and a background model, containing the static part of the scene or, more in general, everything that can be considered as background given the characteristics of the observed scene [7].

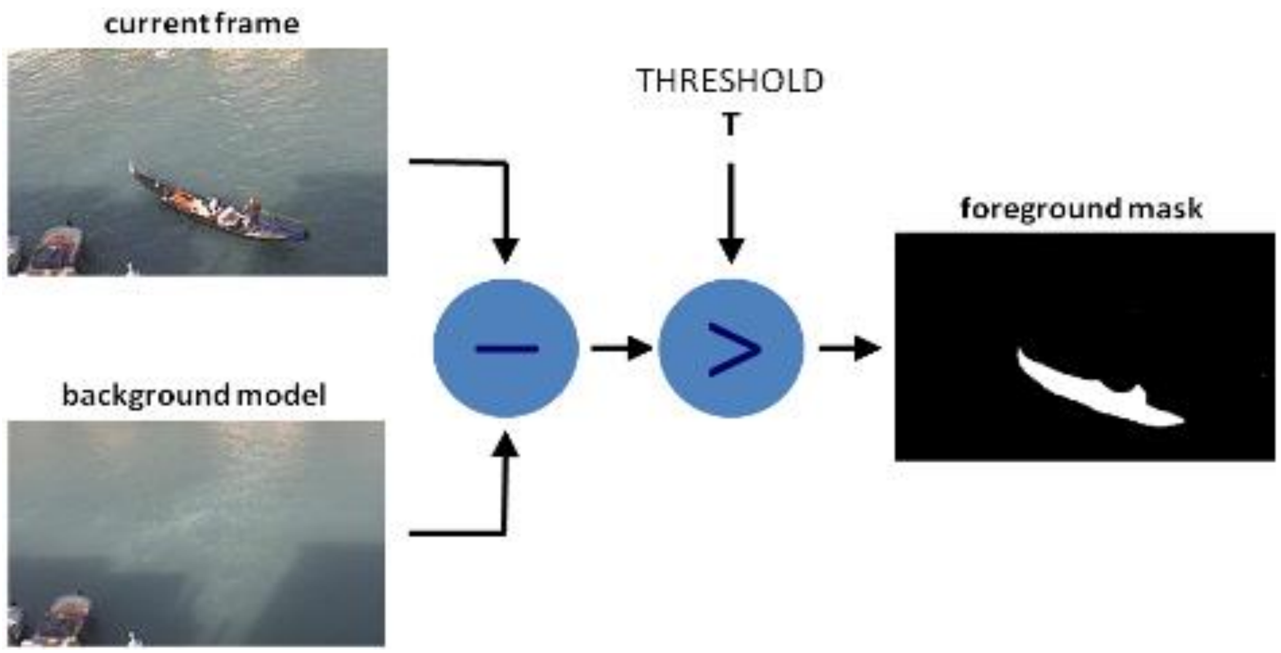

Figure2.3: Background Subtraction model [7].

Background modeling consists of two main steps:

1. Background Initialization.
2. Background Update.

In the first step, an initial model of the background is computed, while in the second step that model is updated in order to adapt to possible changes in the scene.

2.2.3 Gaussian Filter

The Gaussian smoothing operator is a 2-D convolution operator that is used to `blur' images and remove detail and noise. In this sense it is similar to the mean filter but it uses a different kernel that represents the shape of a Gaussian (`bell-shaped') hump [8]. This kernel has some special properties which are detailed below:

How It Works:

The Gaussian distribution in 1-D has the form:

$$G(x) = \left(\frac{1}{\sqrt{2\pi\sigma}}\right) e^{-\left(\frac{x^2}{2\sigma^2}\right)}$$

Here **σ** is the standard deviation of the distribution. We have also assumed that the distribution has a mean of zero (*i.e.* it is centered on the line *x*=0). The distribution is illustrated in Figure 2.4 [12].

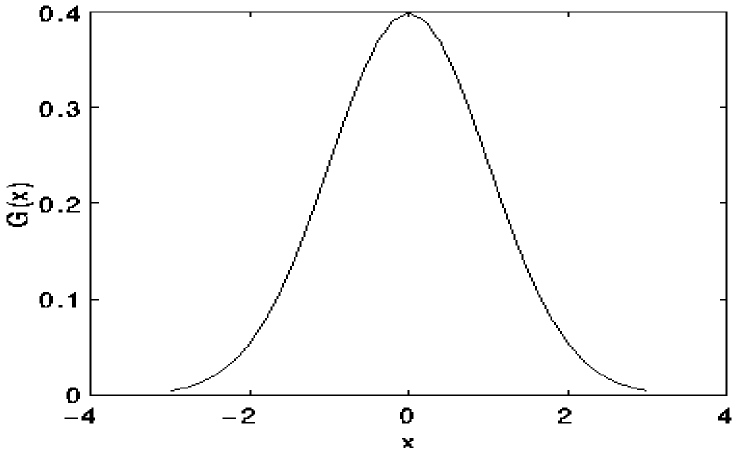

Figure 2.4: 1-D Gaussian distribution with mean 0 and $\boldsymbol{\sigma}$*=1*

In 2-D, an isotropic (*i.e.* circularly symmetric) Gaussian has the form:

$$G(x,y) = \left(\frac{1}{2\pi\sigma^2}\right) e^{-\left\{\frac{(x^2+y^2)}{2\sigma^2}\right\}}$$

This distribution is shown in Figure 2.5

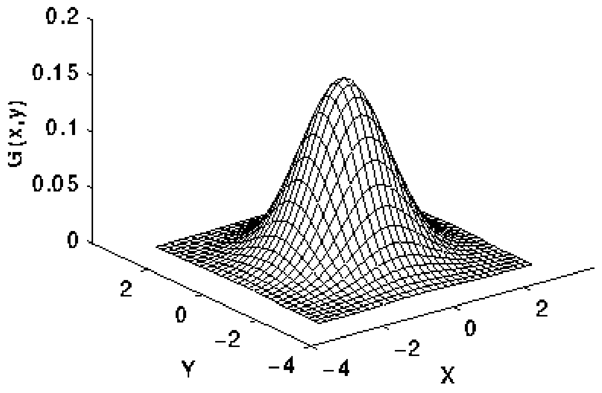

Figure 2.5: *2-D Gaussian distribution with mean (0,0) and* $\boldsymbol{\sigma}$*=1*

The idea of Gaussian smoothing is to use this 2-D distribution as a `point-spread' function, and this is achieved by convolution. Since the image is stored as a collection of discrete pixels we need to produce a discrete approximation to the Gaussian function before we can perform the convolution. In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the kernel at this point. Figure 3 shows a suitable integer-valued convolution kernel that approximates a Gaussian with a $\boldsymbol{\sigma}$of 1.0. It is not obvious how to pick the values of the mask to approximate a Gaussian. One could use the value of the Gaussian at the center of a pixel in the mask, but this is not accurate because the value of the Gaussian varies non-linearly across the pixel. We integrated the value of the Gaussian over the whole pixel (by summing the Gaussian at 0.001 increments). The integrals are not integers: we rescaled the array so that the corners had the value 1. Finally, the 273 is the sum of all the values in the mask [9].

$$\frac{1}{273} \times \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 7 & 4 & 1 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 7 & 26 & 41 & 26 & 7 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 1 & 4 & 7 & 4 & 1 \\ \hline \end{array}$$

***Figure 2.6**: Discrete approximation to Gaussian function with **σ**=1.0*

Once a suitable kernel has been calculated, then the Gaussian smoothing can be performed using standard convolution method. The convolution can in fact be performed fairly quickly since the equation for the 2-D isotropic Gaussian shown above is separable into *x* and *y* components. Thus the 2-D convolution can be performed by first convolving with a 1-D Gaussian in the *x* direction, and then convolving with another 1-D Gaussian in the *y* direction. (The Gaussian is in fact the *only* completely circularly symmetric operator which can be decomposed in such a way.) Figure 4 shows the 1-D *x* component kernel that would be used to produce the full kernel shown in Figure 2.6 (after scaling by 273, rounding and truncating one row of pixels around the boundary because they mostly have the value 0. This reduces the 7x7 matrix to the 5x5 shown above.). The *y* component is exactly the same but is oriented vertically.

.006	.061	.242	.383	.242	.061	.006

Figure 2.7: *One of the pair of 1-D convolution kernels used to calculate the full kernel shown in Figure 2.6 more quickly.*

A further way to compute a Gaussian smoothing with a large standard deviation is to convolve an image several times with a smaller Gaussian. While this is computationally complex, it can have applicability if the processing is carried out using a hardware pipeline.

The Gaussian filter not only has utility in engineering applications. It is also attracting attention from computational biologists because it has been attributed with some amount of biological plausibility, *e.g.* some cells in the visual pathways of the brain often have an approximately Gaussian response.

Usability

The effect of Gaussian smoothing is to blur an image, in a similar fashion to the mean filter. The degree of smoothing is determined by the standard deviation of the Gaussian. (Larger standard deviation Gaussians, of course, require larger convolution kernels in order to be accurately represented.)

The Gaussian outputs a `weighted average' of each pixel's neighborhood, with the average weighted more towards the value of the central pixels. This is in contrast to the mean filter's uniformly weighted average. Because of this, a Gaussian provides gentler smoothing and preserves edges better than a similarly sized mean filter.

One of the principle justifications for using the Gaussian as a smoothing filter is due to its frequency response. Most convolution-based smoothing filters act as lowpass frequency filter. This means that their effect is to remove high spatial frequency components from an image. The frequency response of a convolution filter, *i.e.* its effect on different spatial frequencies, can be seen by taking the transform of the filter. Figure 2.8 shows the frequency responses of a 1-D mean filter with width 5 and also of a Gaussian filter with **σ** = 3

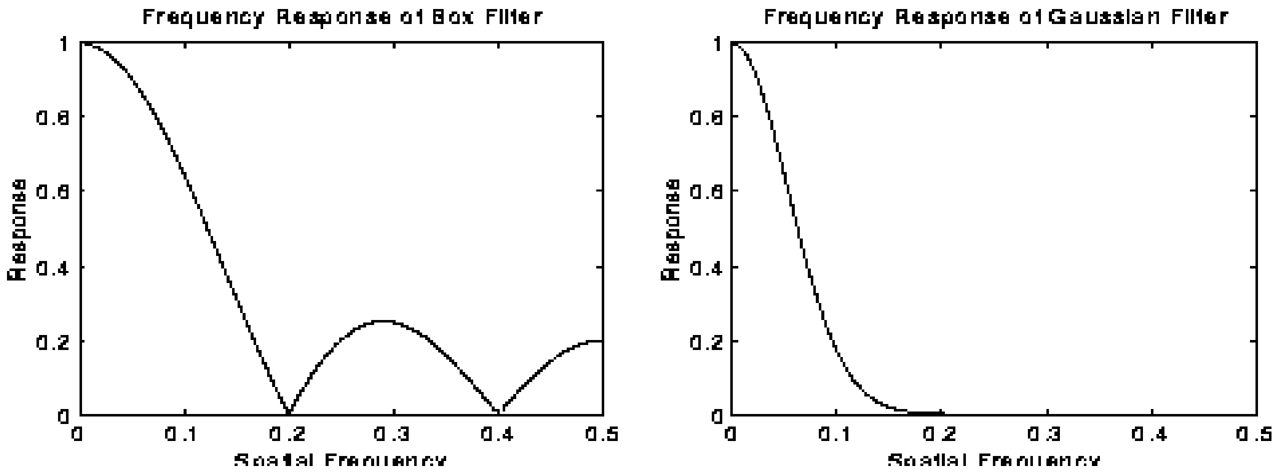

Figure 2.8: *Frequency responses of Box (i.e. mean) filter (width 5 pixels) and Gaussian filter (**σ** = 3 pixels). The spatial frequency axis is marked in cycles per pixel, and hence no value above 0.5 has a real meaning.*

2.3 The Process of Kalman Filtering

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process in several aspects: it supports estimations of past, present, and even future states, and it can do the same even when the precise nature of the modelled system is unknown [5].

The Kalman filter model assumes that the state of a system at a time *t* evolved from the prior state at time *t*-1 according to the equation

$$X_t = F_t X_{t-1} + B_t u_t + w_t \tag{2.3}$$

Where,

- X_t Is the state vector containing the terms of interest for the system (e.g., position, velocity, and heading) at time *t.*
- u_t Is the vector containing any control inputs (steering angle, throttle setting, braking force).
- F_t Is the state transition matrix which applies the effect of each system state parameter at time t-1 on the system state at time t (e.g., the position and velocity at time t-1 both affect the position at time t).
- B_t Is the control input matrix which applies the effect of each control input parameter in the vector $u_t$on the state vector (e.g., applies the effect of the throttle setting on the system velocity and position).
- w_t Is the vector containing the process noise terms for each parameter in the state vector. The process noise is assumed to be drawn from a zero mean multivariate normal distribution with covariance given by the covariance matrixQ_t.

Measurements of the system can also be performed, according to the model

$$z_t = H_t \times X_t + v_t \tag{2.4}$$

Where,

- z_t Is the vector of measurement.
- H_t Is the transformation matrix that maps the state vector parameters into the measurement domain

- v_t Is the vector containing the measurement noise terms for each observation in the measurement vector. Like the process noise, the measurement noise is assumed to be zero mean Gaussian white noise with covariance R_t.

In the derivation that follows, we will consider a simple one-dimensional tracking problem, particularly that of a train moving along a railway line (see
Figure 2.9). We can therefore consider some example vectors and matrices in this problem. The state vector $X_t$contains the position and velocity of the train

$$X_t = \begin{bmatrix} X_t \\ \dot{X}_t \end{bmatrix}$$

The train driver may apply a braking or accelerating input to the system, which we will consider here as a function of an applied force$f_t$and the mass of the train m.

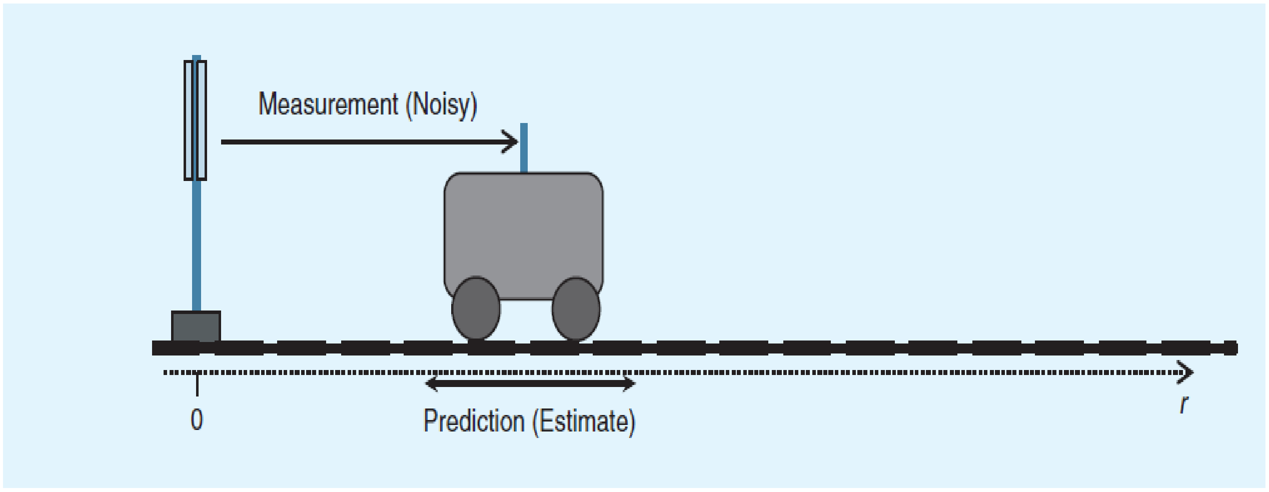

Figure 2.9: *This figure shows the one-dimensional system under consideration.*

Such control information is stored within the control vector u_t

$$u_t = \frac{f_t}{m}.$$

The relationship between the force applied via the brake or throttle during the time period dt (the time elapsed between time epochs t-1 and t) and the position and velocity of the train is given by the following equations:

$$X_t = X_{t-1} + (X_{t-1} * dt) + \frac{f_t (dt)^2}{2m}.$$

$$X_t = \dot{X}_{t-1} + \frac{(f_t dt)}{m}.$$

These linear equations can be written in matrix form as

$$\begin{bmatrix} X_t \\ \dot{X}_t \end{bmatrix} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_{t-1} \\ \dot{X}_{t-1} \end{bmatrix} + \begin{bmatrix} (dt)^2/2 \\ dt \end{bmatrix} * \frac{f_t}{m}.$$

And so by comparison with (1), we can see for this example that

$$F_t = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix}$$
$$B_t = \begin{bmatrix} (dt)^2/2 \\ dt \end{bmatrix}$$

The true state of the system $X_t$cannot be directly observed, and the Kalman filter provides an algorithm to determine an estimate $\hat{X}_t$by combining models of the system and noisy measurements of certain parameters or linear functions of parameters. The estimates of the parameters of interest in the state vector are therefore now provided by probability density functions (pdfs), rather than discrete values.

To fully describe the Gaussian functions, we need to know their variance and covariance, and these are stored in the covariance matrix P_t. The terms along the main diagonal of $P_t$are the variances associated with the corresponding terms in the state vector. The off-diagonal terms of $P_t$provide the covariance between terms in the state vector. In the case of a well-modeled, one-dimensional linear system with measurement errors drawn from a zero-mean Gaussian distribution, the Kalman filter has been shown to be the optimal estimator [10].

The Kalman filter algorithm involves two stages: prediction and measurement update. The standard Kalman filter equations for the prediction stage are

$$\hat{X}_{t|t-1} = F_t \hat{X}_{t-1|t-1} + B_t u_t \tag{2.5}$$

$$P_{t|t-1} = A_t P_{t-1|t-1} A_t^T + Q_t \tag{2.6}$$

Where $Q_t$the process noise covariance matrix is associated with noisy control inputs. Equation (2.5) was derived explicitly in the discussion above. We can derive (2.6) as follows. The variance associated with the prediction $\hat{X}_{t|t-1}$ of an unknown true value $X_t$is given by

$$P_{t|t-1} = E[\left(X_t - \hat{X}_{t|t-1}\right)\left(X_t - \hat{X}_{t|t-1}\right)^T]$$

And taking the difference between (2.5) and (2.3) gives

$$X_t - \hat{X}_{t|t-1} = F\left(X_{t-1} - \hat{X}_{t|t-1}\right) + w_t$$

$$=> P_{t|t-1} = E\left[\left(F\left(X_{t-1} - \hat{X}_{t-1|t-1}\right) + w_t\right) \times \left(F\left(X_{t-1} - \hat{X}_{t-1|t-1}\right) + w_t\right)^T\right)$$

$$= FE\left[\left(X_{t-1} - \hat{X}_{t-1|t-1}\right) \times \left(X_{t-1} - \hat{X}_{t-1|t-1}\right)^T\right] \times F^t$$

$$+ FE\left[\left(X_{t-1} - \hat{X}_{t-1|t-1}\right)w_t^T\right] + E\left[w_t X_{t-1} - \hat{X}_{t-1|t-1}^T\right]F^T + E[w_t w_t^T].$$

Noting that the state estimation errors and process noise are uncorrelated

$$E\left[\left(X_{t-1}-\hat{X}_{t-1|t-1}\right)w_t^T\right]=E\left[w_t\left(X_{t-1}-\hat{X}_{t-1|t-1}\right)^T\right]=0$$

$$\Rightarrow P_{t|t=1}=FE\left[\left(X_{t-1}-\hat{X}_{t-1|t-1}\right)\left(X_{t-1}-\hat{X}_{t-1|t-1}\right)^T\right]F^T+E[w_tw_t^T]\Rightarrow P_{t|t-1}$$
$$=FP_{t-1|t-1}F^T+Q_t.$$

The measurement update equations are given by

$$\hat{X}_{t|t}=\hat{X}_{t|t-1}+K_t\left(z_t-H_t\hat{X}_{t|t-1}\right) \tag{2.7}$$

$$P_{t|t}=P_{t|t-1}-K_tH_tP_{t|t-1}$$

Where,

$$K_t=P_{t|t-1}H_t^T\left(H_tP_{t|t-1}H_t^T+R_t\right)^{-1} \tag{2.8}$$

Derivation

The Kalman filter will be derived here by considering a simple one-dimensional tracking problem specifically that of a train is moving along a railway line. At every measurement epoch we wish to know the best possible estimate of the location of the train (or more precisely, the location of the radio antenna mounted on the train roof). Information is available from two sources:

1) Predictions based on the last known position and velocity of the train. And

2) Measurements from a radio ranging system deployed at the track side.

The information from the predictions and measurements are combined to provide the best possible estimate of the location of the train [11]. The system is shown graphically in Figure 2.9.

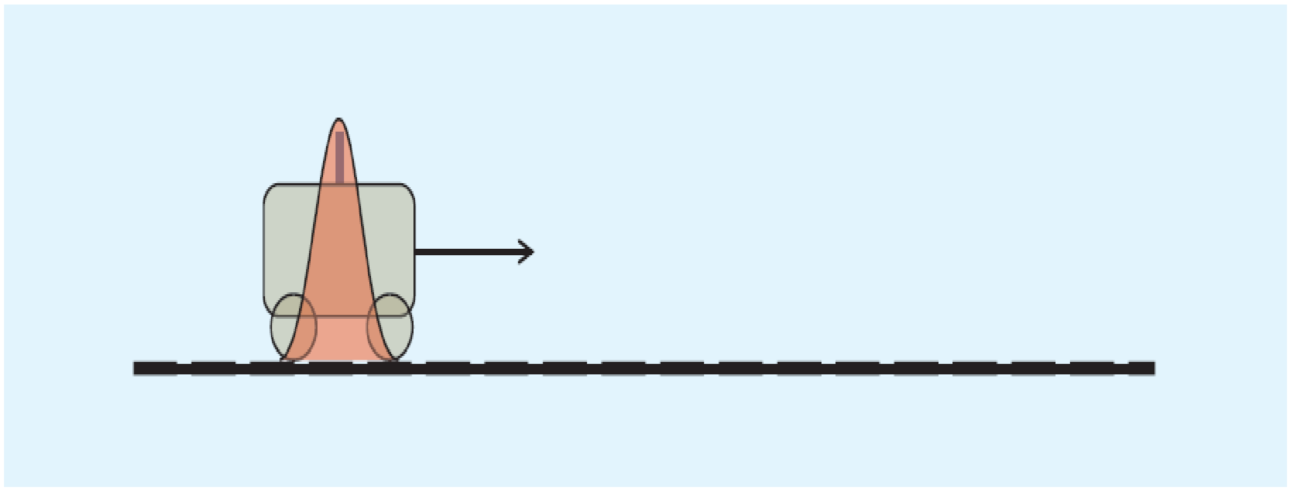

Figure 2.10: *The initial knowledge of the system at time t = 0. The red Gaussian distribution represents the pdf providing the initial confidence in the estimate of the position of the train. The arrow pointing to the right represents the known initial velocity of the train.*

The initial state of the system (at time t = 0 s) is known to a reasonable accuracy, as shown in Figure 2.10. The location of the train is given by a Gaussian pdf. At the next time epoch (t = 1 s), we can estimate the new position of the train, based on known limitations such as its position and velocity at t = 0, its maximum possible acceleration and deceleration, etc. In practice, we may have some knowledge of the control inputs on the brake or accelerator by the driver. In any case, we have a prediction of the new position of the train, represented in Figure 3 by a new Gaussian pdf with a new mean and variance. Mathematically this step is represented by (2.3). The variance has increased, representing our reduced certainty in the accuracy of our position estimate compared to t = 0, due to the uncertainty associated with any process noise from accelerations or decelerations undertaken from time t = 0 to time t = 1.

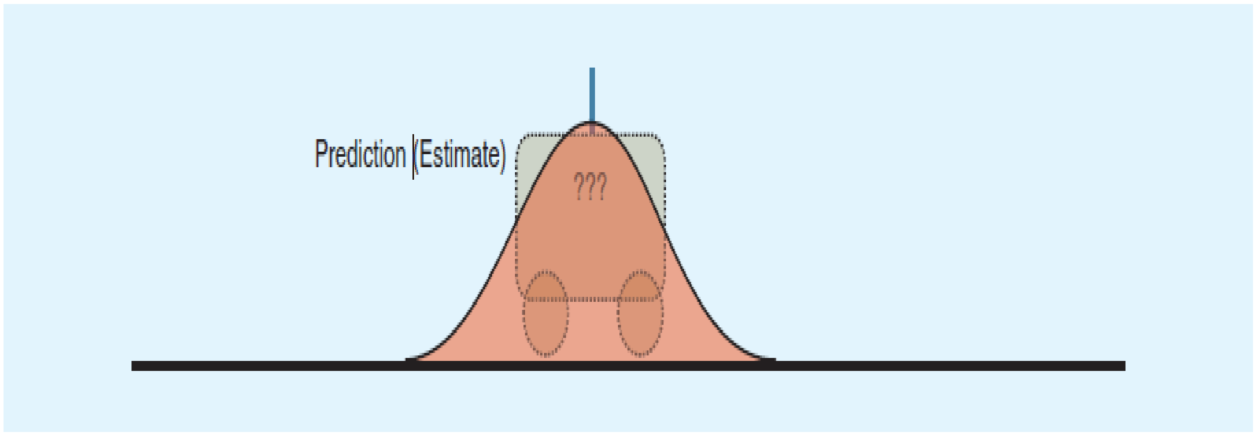

Figure 2.11: *Here, the predict ion of the location of the train at time t = 1 and the level of uncertainty in that prediction is shown. The confidence in the knowledge of the position of the train has decreased, as we are not certain if the train has undergone any accelerations or decelerations in the intervening period from t = 0 to t = 1.*

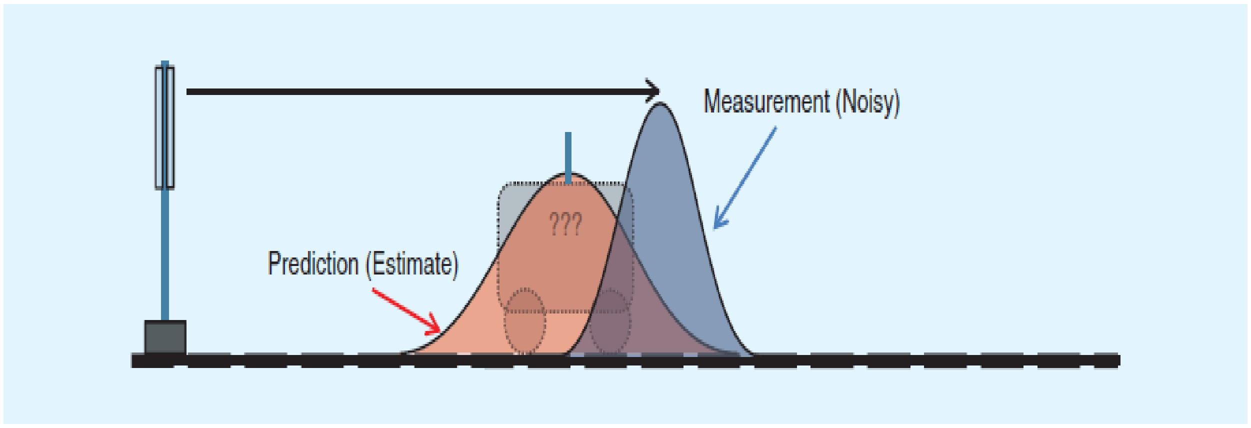

Figure 2.12: *Shows the measurement of the location of the train at time t = 1 and the level of uncertainty in that noisy measurement, represented by the blue Gaussian pdf. The combined knowledge of this system is provided by multiplying these two pdfs together [11].*

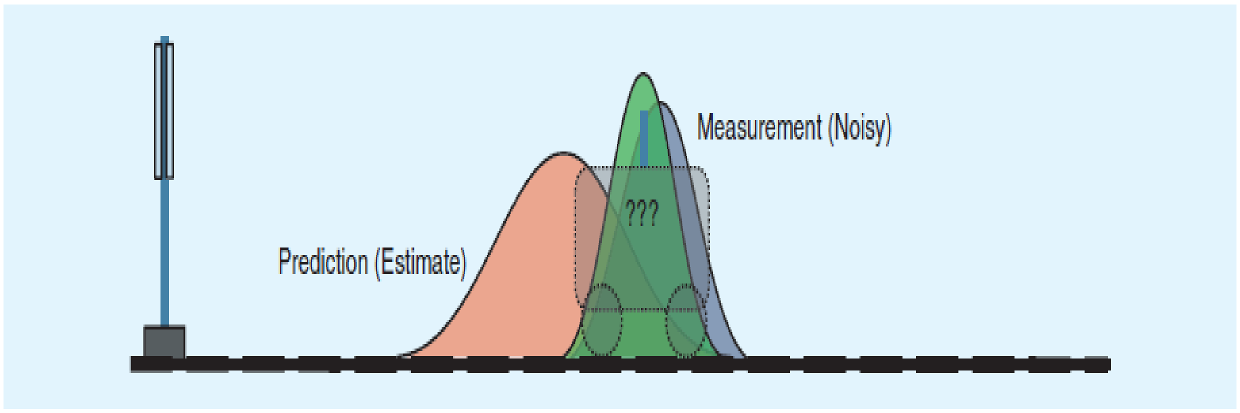

Figure 2.13: *Shows the new pdf (green) generated by multiplying the pdfs associated with the prediction and measurement of the train's location at time t = 1. This new pdf provides the best estimate of the location of the train, by fusing the data from the prediction and the measurement.*

At t = 1, we also make a measurement of the location of the train using the radio positioning system, and this is represented by the blue Gaussian pdf in Figure 2.12. The best estimate we can make of the location of the train is provided by combining our knowledge from the prediction and the measurement. This is achieved by multiplying the two corresponding pdfs together. This is represented by the green pdf in Figure 2.13.

A key property of the Gaussian function is exploited at this point: the product of two Gaussian functions is another Gaussian function. This is critical as it permits an endless number of Gaussian pdfs to be multiplied over time, but the resulting function does not increase in complexity or number of terms; after each time epoch the new pdf is fully represented by a Gaussian function. This is the key to the elegant recursive properties of the Kalman filter.

The stages described above in the figures are now considered again mathematically to derive the Kalman filter measurement update equations.

The prediction pdf represented by the red Gaussian function in Figure 2.11 is given by the equation

$$y_1(r; \mu_1, \sigma_1 \triangleq \left(\frac{1}{\sqrt{2\pi\sigma_1^2}}\right) e^{-\frac{(r-\mu_1)^2}{2\sigma_1^2}} \tag{2.9}$$

The measurement pdf represented by the blue Gaussian function in Figure 2.12 is given by

$$y_2(r; \mu_2, \sigma_2) \triangleq \left(\frac{1}{\sqrt{2\pi\sigma_2^2}}\right) e^{-\frac{(r-\mu_2)^2}{2\sigma_2^2}} \tag{2.10}$$

The information provided by these two pdfs is fused by multiplying the two together, i.e., considering the prediction and the measurement together (see Figure 2.11). The new pdf representing the fusion of the information from the prediction and measurement, and our best current estimate of the system, is therefore given by the product of these two Gaussian functions

$$\begin{aligned} y_{fused}(r; \mu_1, \sigma_1, \mu_2, \sigma_2) &= \left(\frac{1}{\sqrt{2\pi\sigma_1^2}}\right) e^{-\frac{(r-\mu_1)^2}{2\sigma_1^2}} \times \left(\frac{1}{\sqrt{2\pi\sigma_2^2}}\right) e^{-\frac{(r-\mu_2)^2}{2\sigma_2^2}} \\ &= \left(\frac{1}{2\pi\sqrt{\sigma_1^2\sigma_1^2}}\right) \boldsymbol{e}^{-((\boldsymbol{r}-\mu_1)^2 \div 2\boldsymbol{\sigma}_1^2)+((\boldsymbol{r}-\mu_2)^2/2\boldsymbol{\sigma}_1^2)} \end{aligned} \tag{2.11}$$

The quadratic terms in this new function can expanded and then the whole expression rewritten in Gaussian form [11]

$$y_{fused}(r; \mu_{fused}, \sigma_{fused}) = \left(\frac{1}{\sqrt{2\pi\sigma_{fused}^2}}\right) e^{-\frac{\left(r-\mu_{fused}\right)^2}{2\sigma_{fused}^2}} \tag{2.12}$$

Where,

$$\mu_{fused} = \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2} = \mu_1 + \frac{\sigma_1^2(\mu_2 - \mu_1)}{(\sigma_1^2 + \sigma_2^2)} \tag{2.13}$$

And,

$$\sigma_{fused}^2 = \frac{\sigma_1^2\sigma_2^2}{(\sigma_1^2 + \sigma_2^2)} = \sigma_1^2 - \left(\frac{\sigma_1^4}{\sigma_1^2 + \sigma_2^2}\right) \tag{2.14}$$

These last two equations represent the measurement update steps of the Kalman filter algorithm, as will be shown explicitly below. However, to present a more general case, we need to consider an extension to this example.

In the example above, it was assumed that the predictions and measurements were made in the same coordinate frame and in the same units. This has resulted in a particularly concise pair of equations representing the prediction and measurement update stages. It is important to note however that in reality a function is usually required to map predictions and measurements into the same domain. In a more realistic extension to our example, the position of the train will be predicted directly as a new distance along the railway line in units of meters, but the time of flight measurements are recorded in units of seconds.

To allow the prediction and measurement pdfs to be multiplied together, one must be converted into the domain of the other, and it is standard practice to map the predictions into the measurement domain via the transformation matrix $H_{t.}$

We now revisit (2.9) and (2.10) and, instead of allowing y_1 and y_2 to both represent values in meters along the railway track, we consider the distribution $y2$ to represent the time of flight in seconds for a radio signal propagating from a transmitter positioned at x = 0 to the antenna on the train. The spatial prediction pdf $y1$ is converted into the measurement domain by scaling the function by c, the speed of light. Equations (2.9) and (2.10) therefore must be rewritten as

$$y_1(s; \mu_1, \sigma_1, c) \triangleq \left(\frac{1}{\sqrt{2\pi\left(\frac{\sigma_1}{c}\right)^2}}\right) e^{-\frac{\left(s-\left(\frac{\mu_1}{c}\right)\right)^2}{2\left(\frac{\sigma_1}{c}\right)^2}} \tag{2.15}$$

And,

$$y_2(s; \mu_2, \sigma_2) \triangleq \left(\frac{1}{\sqrt{2\pi\sigma_2^2}}\right) e^{-\frac{(s-\mu_2)^2}{2\sigma_2^2}} \tag{2.16}$$

Where both distributions are now defined in the measurement domain, radio signals propagate along the time "*s*" axis, and the measurement unit is the second. Following the derivation as before we now find

$$\frac{\mu_{fused}}{c} = \left(\frac{\mu_1}{c}\right) + \frac{\left\{\left(\frac{\sigma_1}{c}\right)\text{^}2\left(\mu_2 - \left(\frac{\mu_1}{c}\right)\right)\right\}}{\left\{\left(\frac{\sigma_1}{c}\right)^2 + \sigma_2^2\right\}}.$$

$$\Rightarrow \mu_{fused} = \mu_1 + \left(\frac{\frac{\sigma_1^2}{c}}{\left(\left(\frac{\sigma_1}{c}\right)^2 + \sigma_1^2\right)}\right) \times \left(\mu_2 - \left(\frac{\mu_1}{c}\right)\right) \tag{2.17}$$

Substituting H= 1/c and $K = \left(H\sigma_1^2/(H^2\sigma_1^2 + \sigma_2^2)\right)$ results in

$$\mu_{fused} = \mu_1 + K \times (\mu_2 - H\mu_1) \tag{2.18}$$

Similarly the fused variance estimate becomes

$$\frac{\mu_{fused}^2}{c^2} = \left(\frac{\sigma_1}{c}\right)^2 - \frac{\left(\frac{\sigma_1}{c}\right)^4}{\left\{\left(\frac{\sigma_1}{c}\right)^2 + \sigma_2^2\right\}}.$$

$$\Longrightarrow \sigma_{fused}^2 = \sigma_1^2 - \frac{\left(\frac{\left(\frac{\sigma_1^2}{c}\right)}{\left\{\left(\frac{\sigma_1}{c}\right)^2 + \sigma_2^2\right\}}\right)\sigma_1^2}{c}.$$

$$= \sigma_1^2 - KH\sigma_1^2 \tag{2.19}$$

We can now compare certain terms resulting from this scalar derivation with the standard vectors and matrices used in the Kalman filter algorithm:

$\mu_{fused} \rightarrow \hat{X}_{t\mid t}$	The state vector following data fusion.
$\mu_1 \rightarrow \hat{X}_{t\mid t-1}$	The state vector before data fusion, i.e., the prediction.
$\sigma_{fused}^2 \rightarrow P_{t\mid t}$	The covariance matrix (confidence) following data fusion.
$\sigma_1^2 \rightarrow P_{t\mid t-1}$	The covariance matrix (confidence) before data fusion.
$\mu_2 \rightarrow z_t$	The measurement vector.
$\sigma_2^2 \rightarrow R_t$	The uncertainty matrix associated with a noisy set of measurements.
$H \rightarrow H_t$	The transformation matrix used to map state vector parameters into the measurement domain.

$$K = \frac{(H\sigma_1^2)}{(H^2\sigma_1^2 + \sigma_2^2)} \rightarrow K_t = P_{t\mid t-1}H_t^T\left(H_tP_{t\mid t-1}H_t^T + R_t\right)^{-1}$$ K= The Kalman Gain.

It is now easy to see how the standard Kalman filter equations relate to (2.18) and (2.19) derived above:

$$\mu_{fused} = \mu_1 + \left\{\frac{H\sigma_1^2}{(H^2\sigma_1^2 + \sigma_2^2)}\right\} \times (\mu_2 - H\mu_1)$$
$$\rightarrow \hat{X}_{t|t} = \hat{X}_{t|t-1} + K_t\left(z_t = H_t\hat{X}_{t|t-1}\right)$$
$$\sigma_{fused}^2 = \sigma_1^2 - \left\{\frac{H\sigma_1^2}{(H^2\sigma_1^2 + \sigma_2^2)}\right\} H\sigma_1^2$$
$$\rightarrow P_{t|t} = P_{t|t-1} - K_t H_t P_{t|t-1}.$$

The Kalman filter can be taught using a simple derivation involving scalar mathematics, basic algebraic manipulations, and an easy-to-follow thought experiment.

Chapter 3

Object Tracking and Implementation

3.1 Object Detection

Block diagram representation of steps for object detection is shown as,

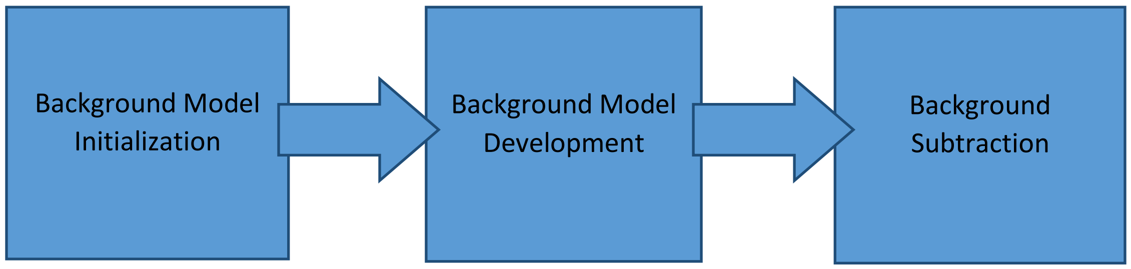

Figure 3.1: *Block diagram for Object Detection*

Background is a stationary layer located behind all other layers. Anything that is not background is likely foreground i.e., moving objects. Background model is initialized by considering first ‘n’ frames. Here ‘30’ frames are considered. The background model is developed by averaging these initial ‘30’ frames [1].

For implementing the object detection process, we apply the following methods:

3.1.1 Make Average of Background Images

For making average of background we read first N-th number of images and then take the average image of them. We use this process because it’s make a good image model of what the background looks like.

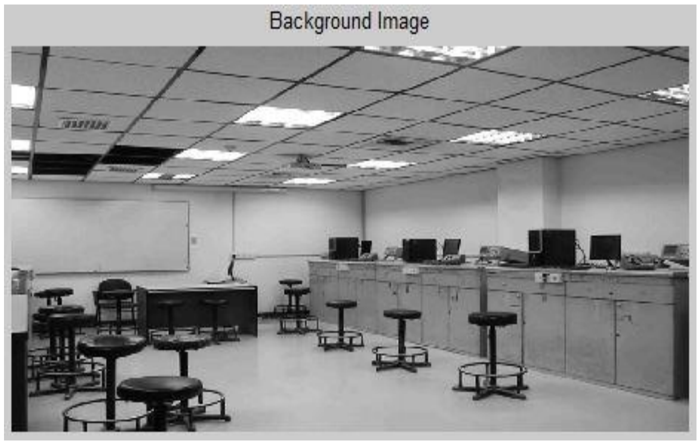

Figure 3.2: *Average Background Image*

3.1.2 Subtract Background

Background subtraction is a process for removing background from an image. We subtract background from the current images to find the moving object. That is the best useful process for finding moving object.

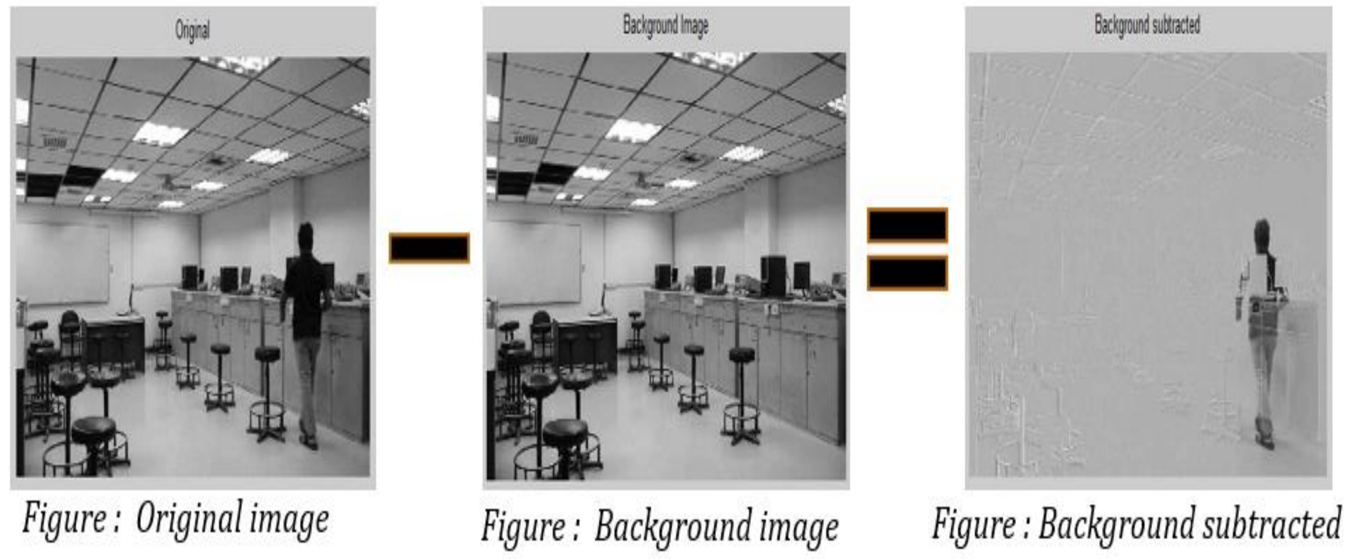

Figure 3.3: *Subtract Background Image*

3.1.3 Gaussian Filter

We use filter for removing noise and shadows from the images. Here, we use Gaussian filter for filtering the images. Gaussian filter is a default filter in mat lab, we can just change the filter size and its sigma value. Here we use filter size is 20 and sigma is 10.

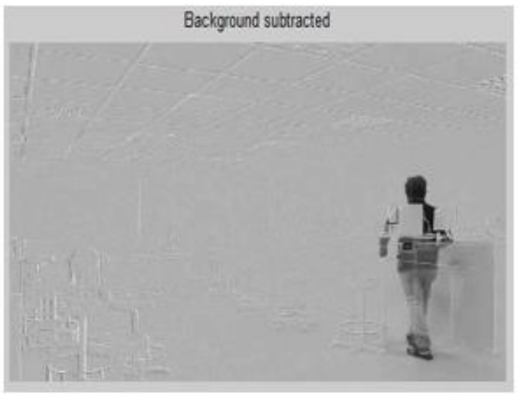

Figure : Image with noise

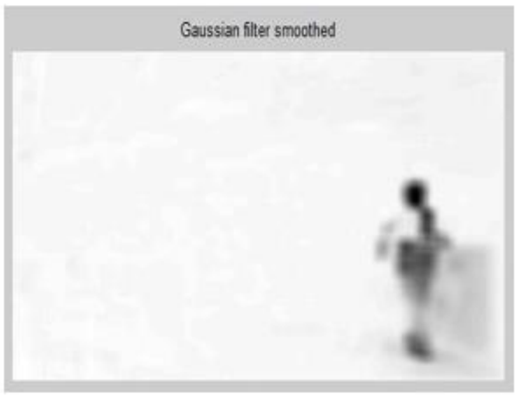

Figure : Filtered image

Figure 3.4: *Gaussian Filtering*

3.1.4 Threshold Process and find Centroid

Thresholding process is an image processing technique for converting grayscale or color image to a binary image based on a threshold value [1]. For finding more visible object we apply thresholding. Here, we applying histogram based thresholding process. From the histogram we find the value for thresholding. Then we find out the centroid value of the moving object from the threshold images. This centroid value is used for measurement value in object tracking.

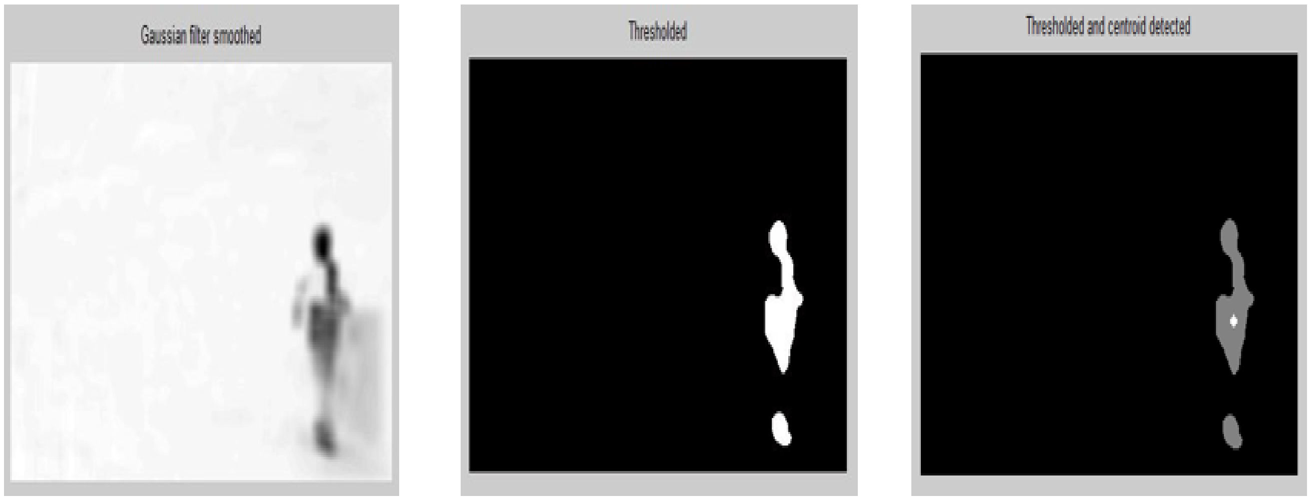

Figure : Filtered image *Figure : Image with noise* *Figure : Centroid Detect*

Figure 3.5: *Threshold and Detect Centroid*

3.2 Object Tracking Using Kalman Filter

Once the object has been detected then it can be tracked along its path. Many standard methods are available for object tracking Wiz. Kalman filter, Particle filter, Mean-shift based kernel tracking etc. [5]. Here, for our project we use Kalman Filter. Because, this Kalman Filter is one of the greater discoveries in the history of statistical estimation theory and possibly the greatest discovery in the twentieth century. It has enabled mankind to do many things that could not have been done without it, and it has become as indispensable as silicon in the makeup of many electronic systems.

As we already give the detail description of Kalman filter method in above, here we give short note about the Kalman filter for understanding our simulation precisely.

The equations for Kalman filters fall in two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimate for the next time step. The measurement update equations are responsible for the feedback. That is used for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate. The time update equations can also be thought of as predictor equations, while the measurement update equations can be thought of as corrector equations. Given figure shows the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems .The time update projects the current state estimate ahead in time. The measurement update adjusts the projected estimate by an actual measurement at that time [5].

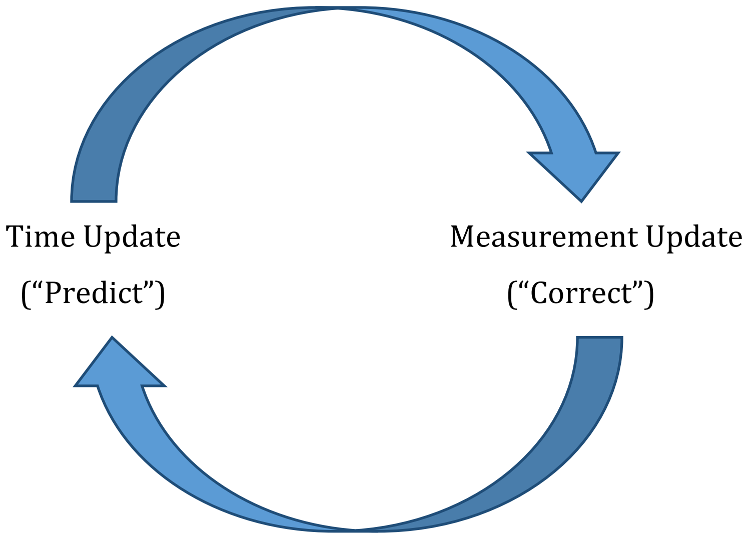

Figure 3.6: *Discrete Kalman Filter Cycle*

- The time update projects the current state estimate ahead in time. Time update equations:

$$\textit{State Prediction,} \qquad Xpred_t = A \times X_{t-1} + B \times u_t + w_{t-1} \tag{3.2}$$

$$\textit{Error Covariance Prediction,} \qquad Ppred_t = A \times P_{t-1}A^T + Q \tag{3.3}$$

In eq. (3.2) $Xpred_t$ is vector representing predicted process state at time *t.* X is a 4-dimensional vector [x y dx dy], where x and y represent the coordinates of the object's center, and dx and dy represent its velocity. $X_{t-1}$is vector representing process state at time *t-1.* A is a 4x4 process transition matrix of the form.

$$A = \begin{matrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$u_t$is a control vector and B relates optional control vector u_t into state space. $w_{t-1}$is a process noise. In eq. (3.3) $Ppred_t$ is predicted error covariance at time t. P_{t-1} is a matrix representing error covariance in the state prediction at time *t-1*, and Q is the process noise covariance.

- The Measurement Update equations:

After predicting the state $Xpred_t$ and its error covariance at time t using the time update steps, the Kalman filter next uses measurement to correct its prediction during the measurement update steps.

$$Kalman\ Gain, K_t = Ppred_t \times H^T \times (H \times Ppred_t \times H^T + R)^{-1} \tag{3.4}$$

$$State\ Update, X_t = Xpred_t + K_t \times (Z_t - H \times Xpred_t) \tag{3.5}$$

$$Error\ Covariance\ Update, P_t = (I - K_t \times H) \times Ppred_t \tag{3.6}$$

In eq. (3.4) K_t is Kalman gain. H is matrix converting state space into measurement space and R is measurement noise covariance. Determining R_t for set of measurement is difficult, many Kalman implementations statistically analyse training data to determine fixed R for all future time updates. In eq. (3.5) X_t is a process actual state. Using Kalman gain K_t and measurement Z_t process state X_t can be updated. Here $Z_t$is the most likely x and y coordinates of the target objects in the frame. The final step in Kalman filter is to update the error covariance $Ppred_t$into P_t as given in eq. (3.6). After each time and measurement update pair, the process is repeated with previous posteriori estimates used to project or predict the new priori estimate.

3.3 Implementation

Tracking of moving object has been done using Kalman filter. Here tracking of any object can be done by providing the frame number from which tracking has to be started.
Following steps have been implemented for tracking a single object.

- Background frame has been calculated by taking average of first 30 frames.
- Then, subtract average background image (frame) from current image.
- Apply Gaussian Filter for smoothing the subtracted image and removing the noise and shadows from the subtracted image.
- Thresholding process has been applied based on histogram for finding where the object is locate.
- Calculate the centroid position of the object and store the data in array for measurement values.
- For selected object its centroid position has been found out and from centroid information all the equation of time and measurement update have been calculated.
- Then, for selected frame the actual position X and error P has been calculated.

Our proposed method:

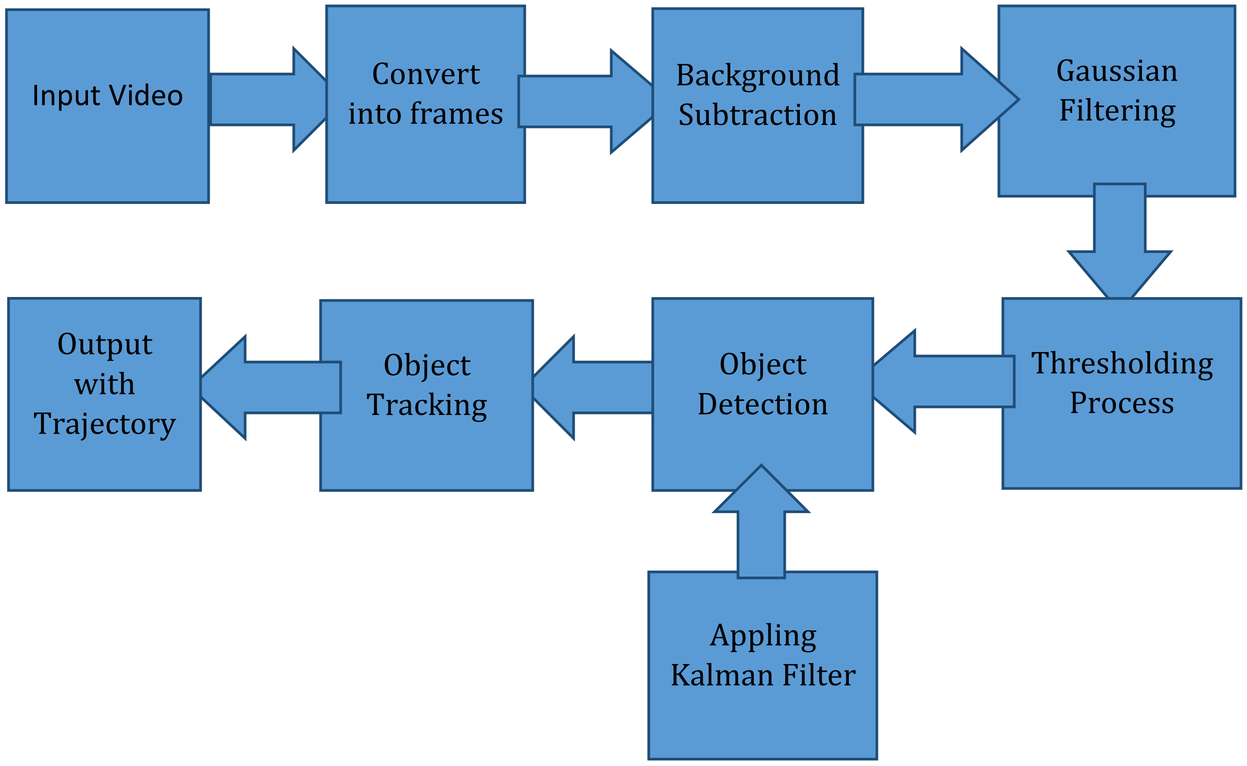

Figure 3.7: *Block diagram for object detection and tracking*

- For all remaining frames following steps have been repeated-
  - ✓ Background subtraction has been done to find out all the moving regions in the frame.
  - ✓ From the found regions, region with the lowest distance from the region selected in previous frame has been selected, that is done with thresholding process by using histogram.
  - ✓ Selected region's centroid and other parameter have been used to calculate time and measurement update equations.
  - ✓ Obtained state position values X has been stored in Array for every frame.
  - ✓ Line joining each stored point has been drawn in final frame which shows the trajectory of the selected moving object [5].

Chapter 4

Results and Analysis

A simulation of the proposed algorithm to determine the real position of the moving object in a video which are describe in chapter 03. Object tracking and implementation was performed to verify this algorithm. This experiment was designed in 2-D space. At first we assume that there is no object in the first ‘n’ frames. We take the average of the ‘n’th image frames for make our background image. We take here average image because it makes sharper image for background. Here, n is an integer numbers. We consider n=30 for our method.

The tracking algorithm has been successfully applied on two sample video. The algorithm have been implemented and tested on Matlab2013a (64bit) with the operating system windows 10.

Video-1: This is a person moving video. It consists of 1028 frames. Resolution of the video 480*640.The video captured in indoor environments.

Video-2: This is a car moving video. It consists of 687 frames. Resolution of the video 480*640. The video captured in outdoor environments.

The different stages of results of different frames for the input video are shown in below with figure.

Video-1(Indoor environment):

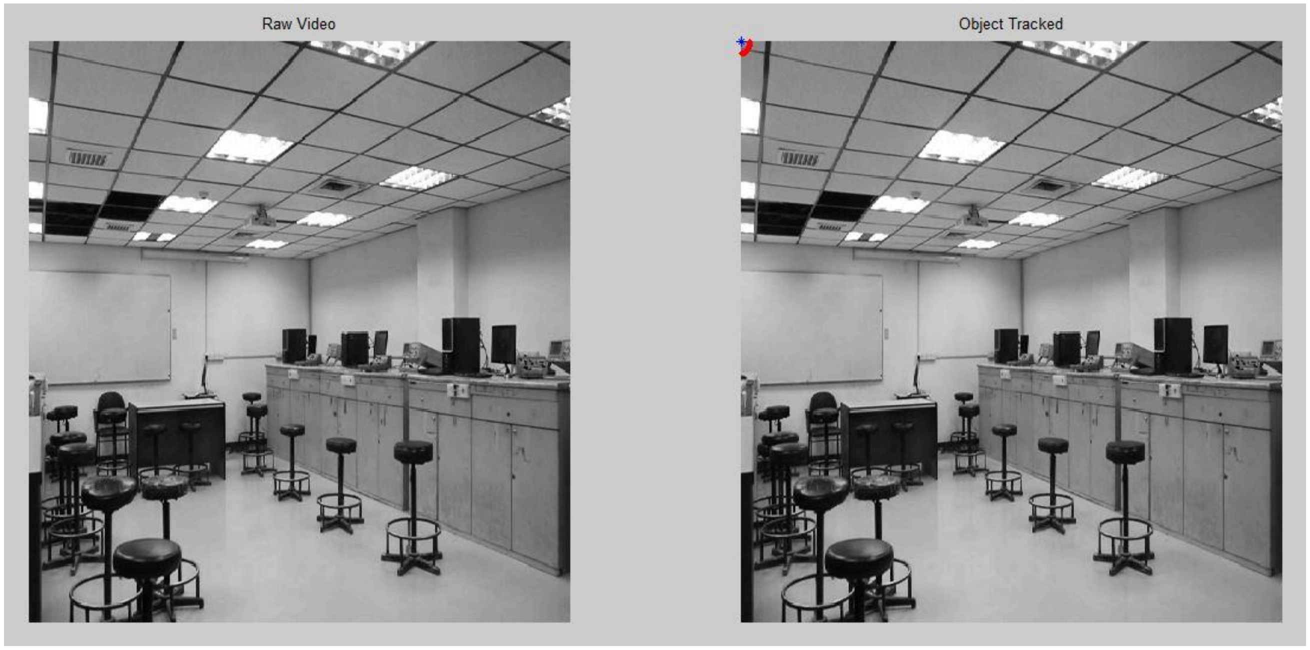

Figure-4.1: Result for frame number: 1

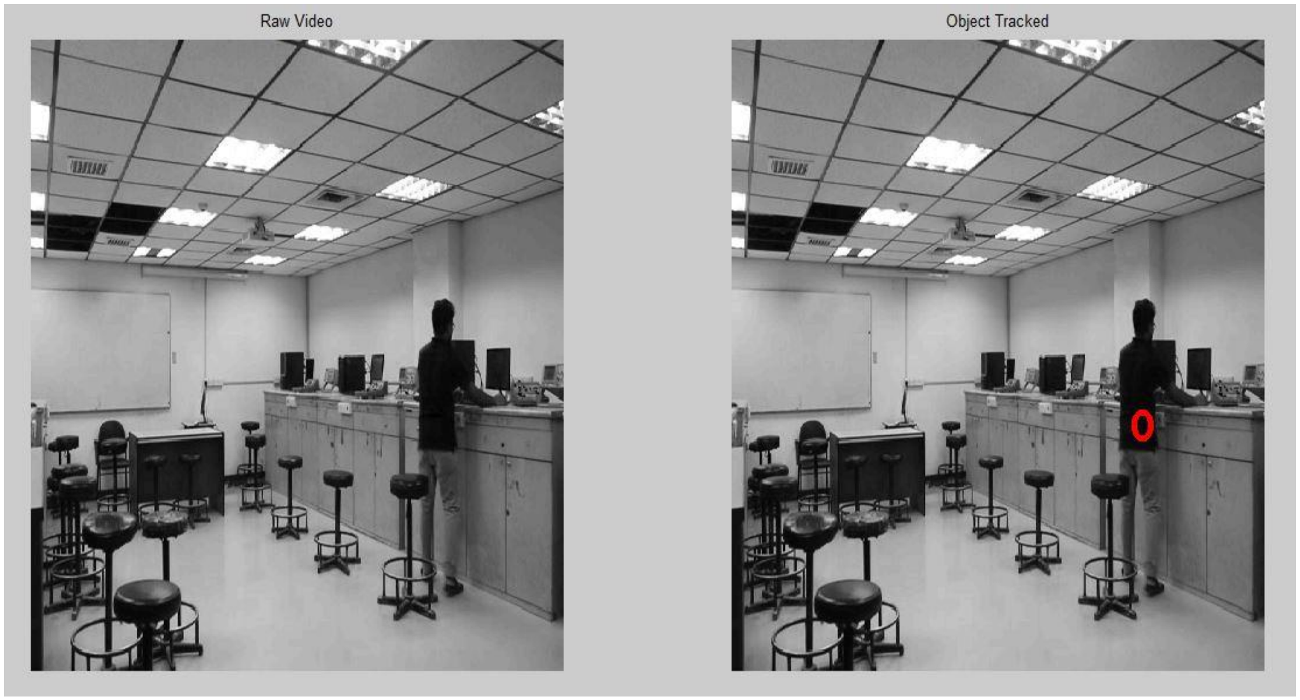

Figure-4.2: Result for frame number: 100

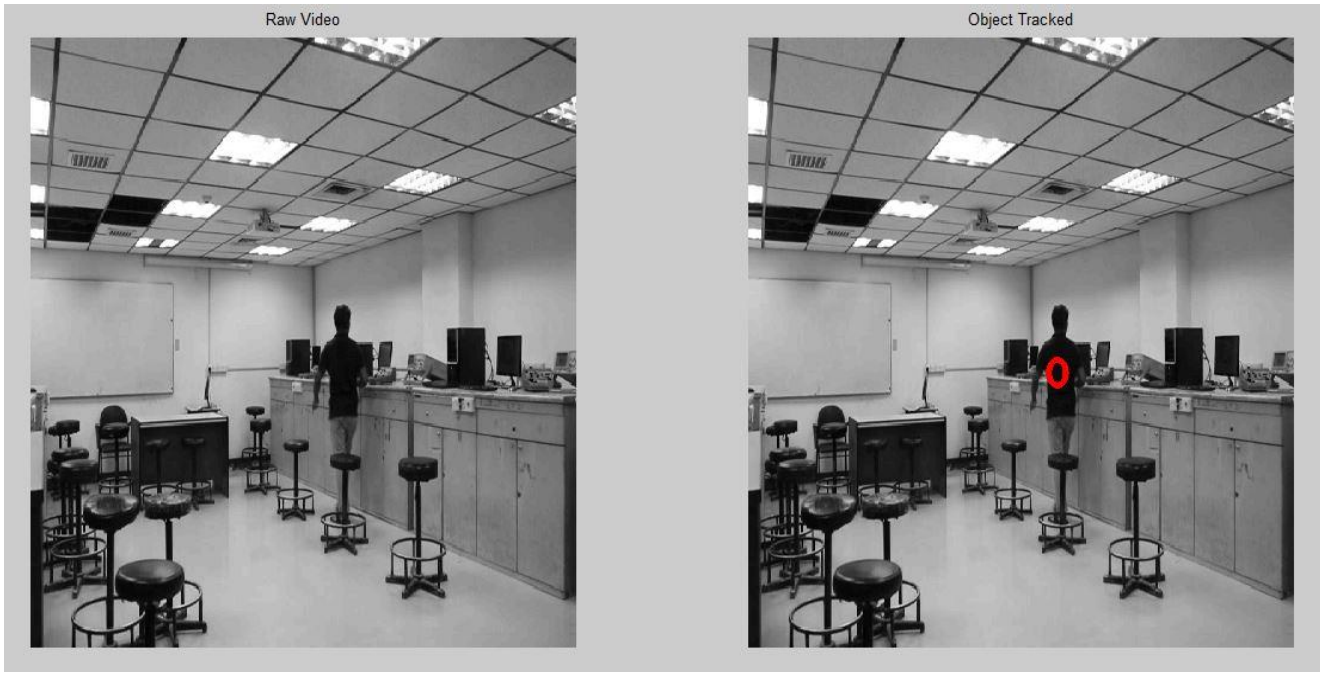

Figure-4.3: Result for frame number: 300

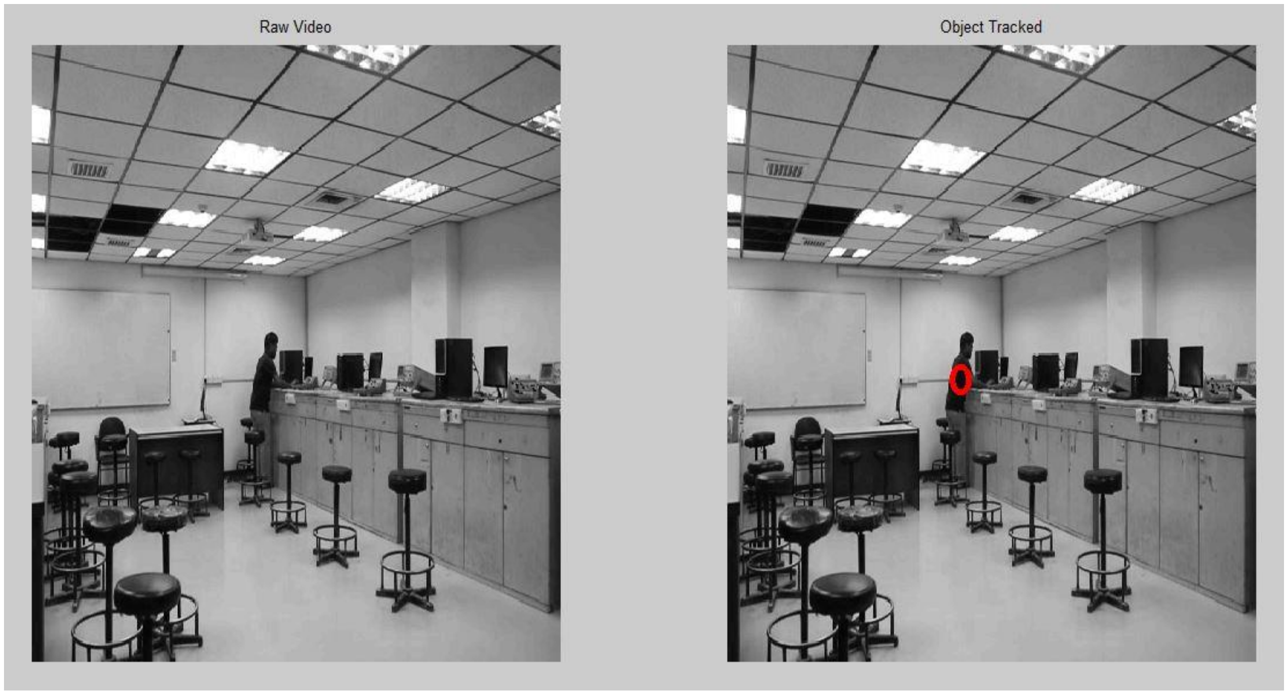

Figure-4.4: Result for frame number: 500

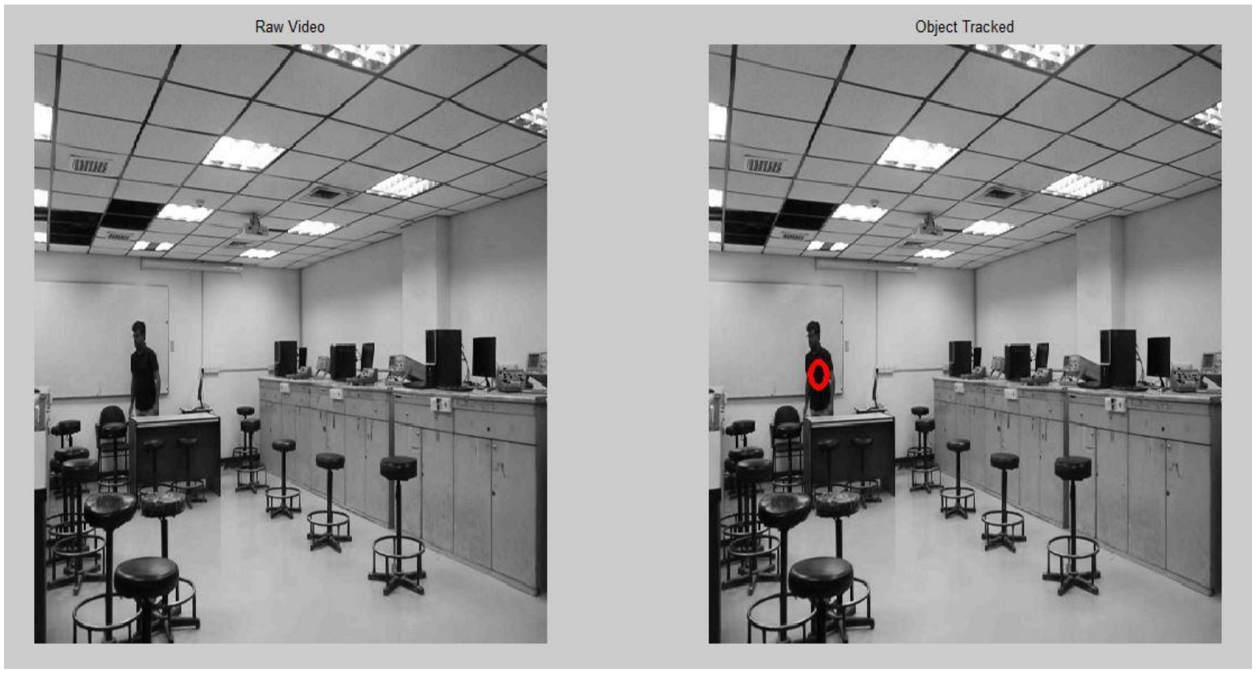

Figure-4.5: Result for frame number: 700

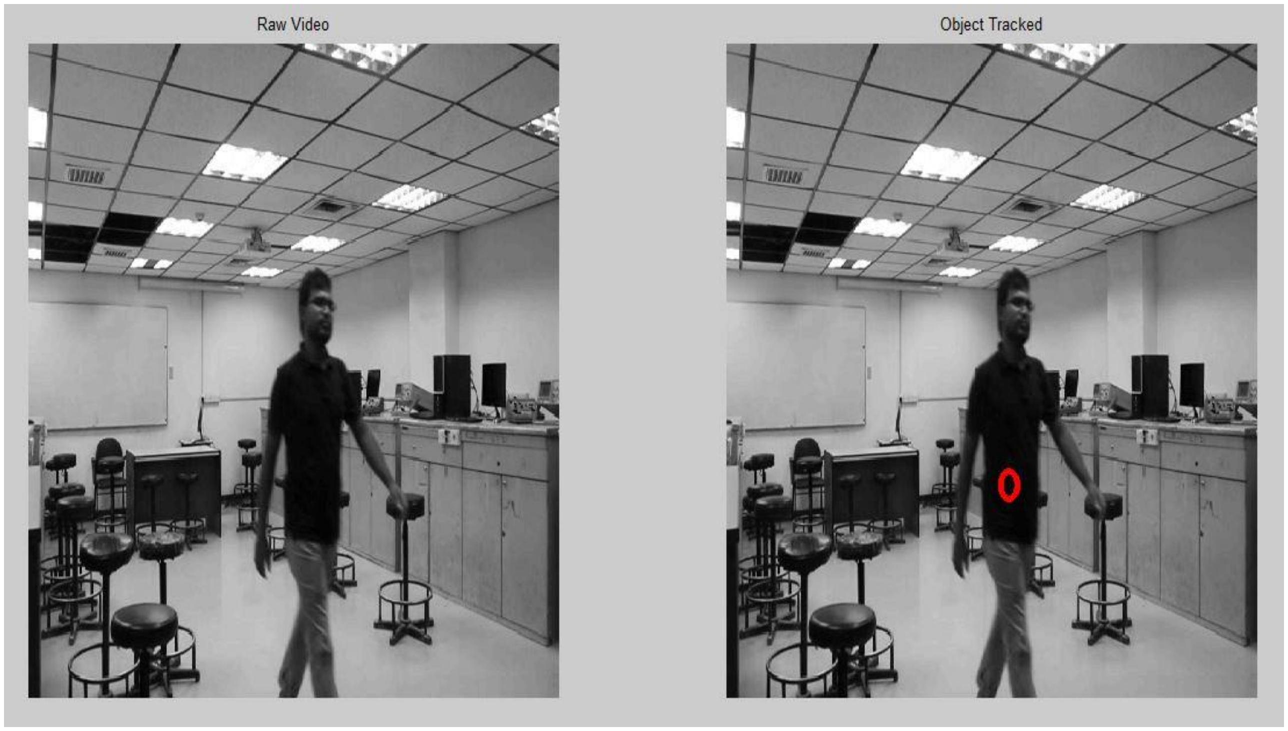

Figure-4.6: Result for frame number: 900

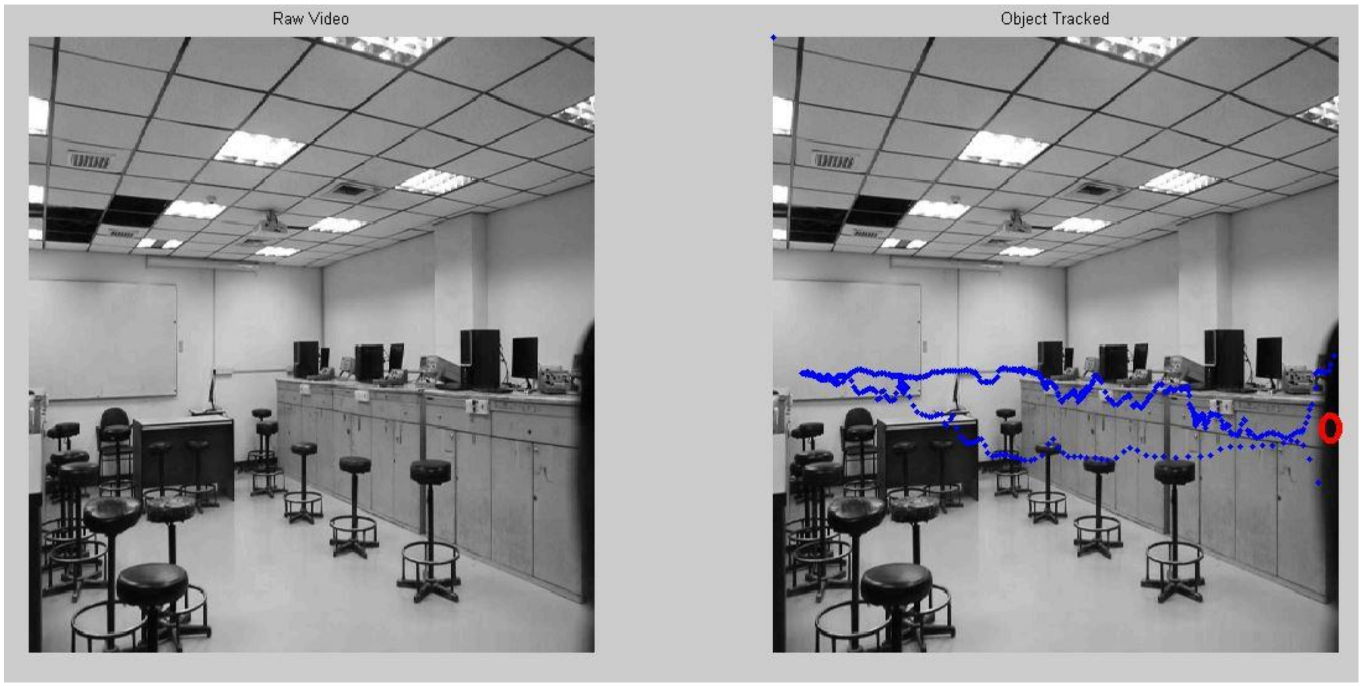

Figure-4.7: Traveling path of the person in frame position

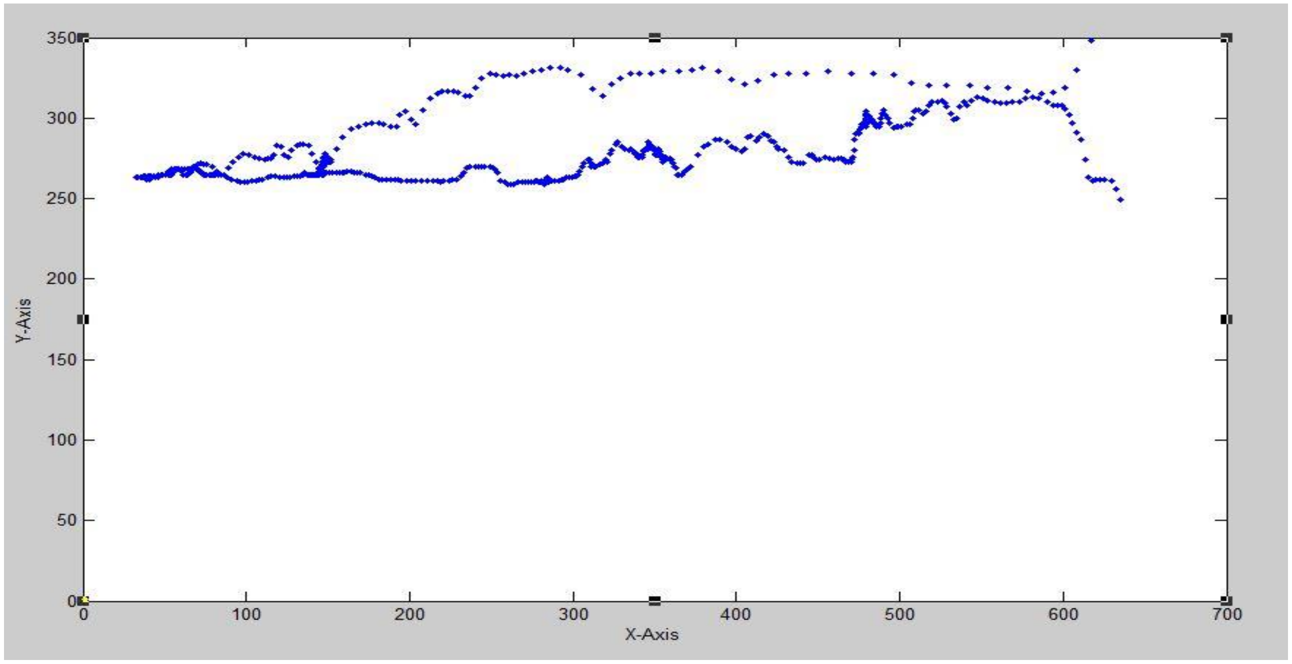

Figure-4.8: Graphical representation of object path (person path)

Video-2(outdoor environment):

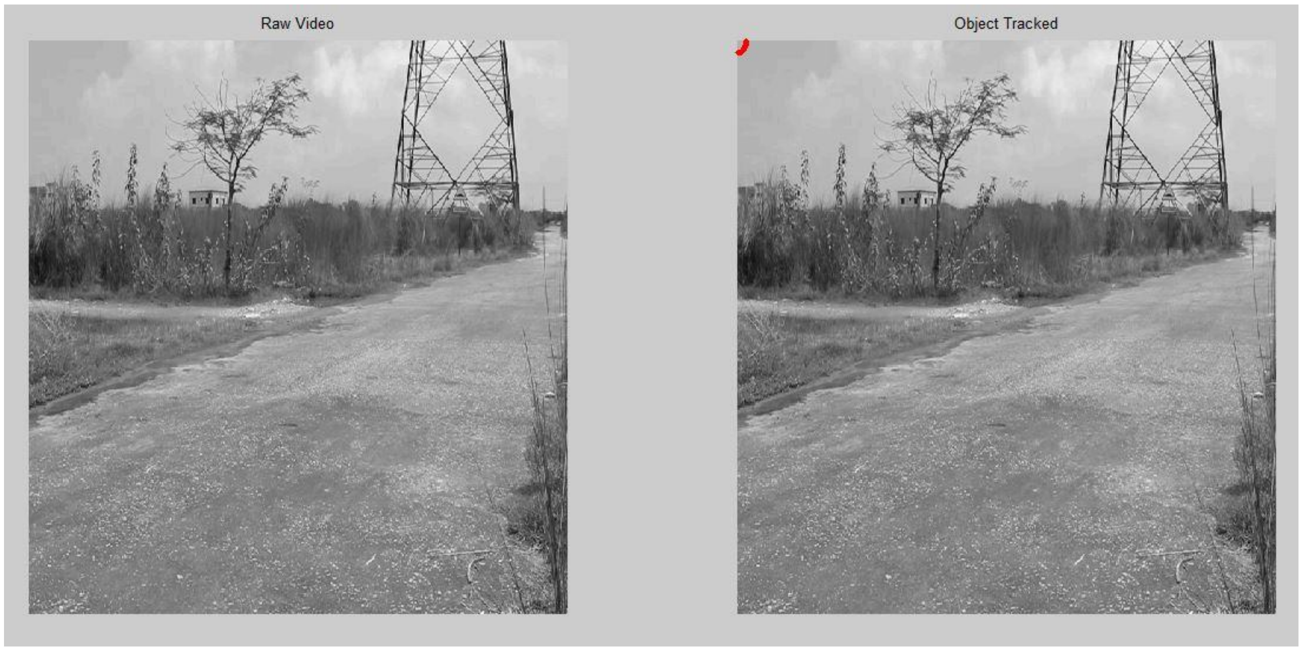

Figure-4.9: Result for frame number: 1

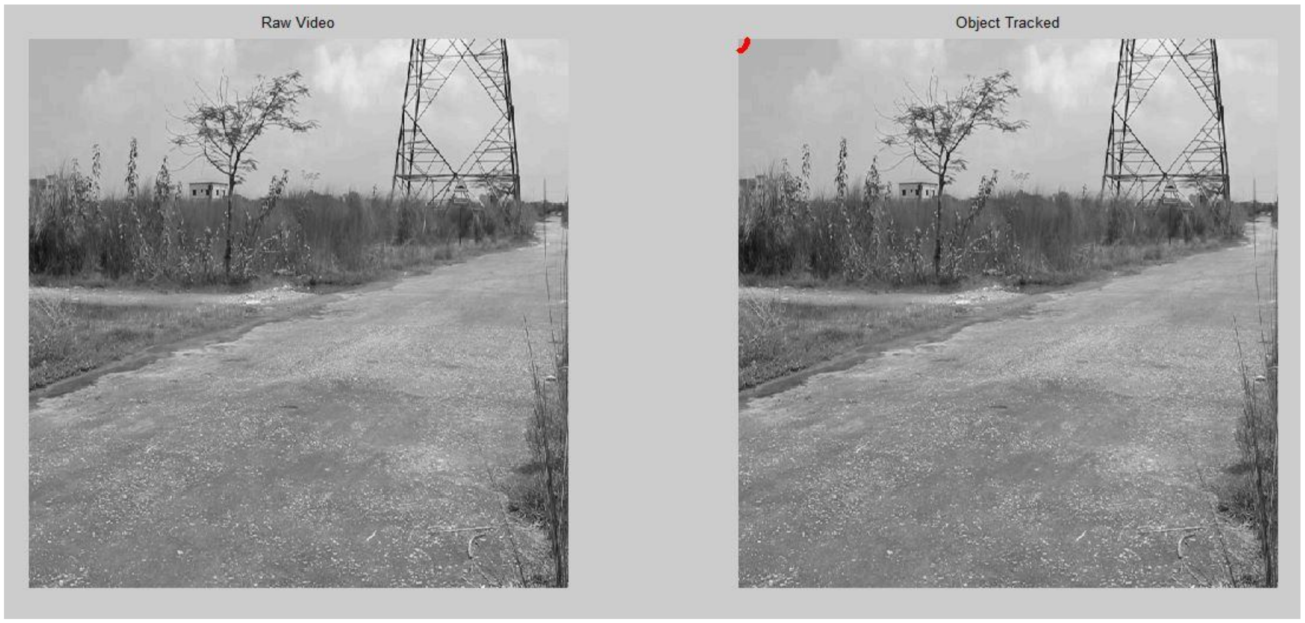

Figure-4.10: Result for frame number: 200

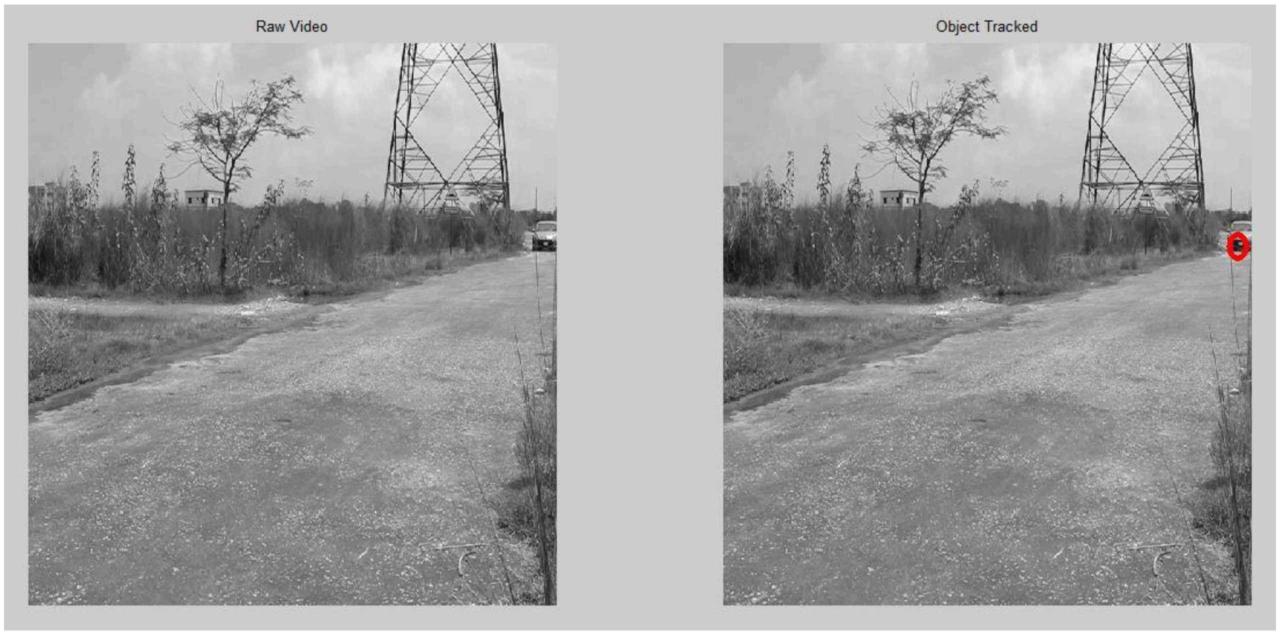

Figure-4.11: Result for frame number: 300

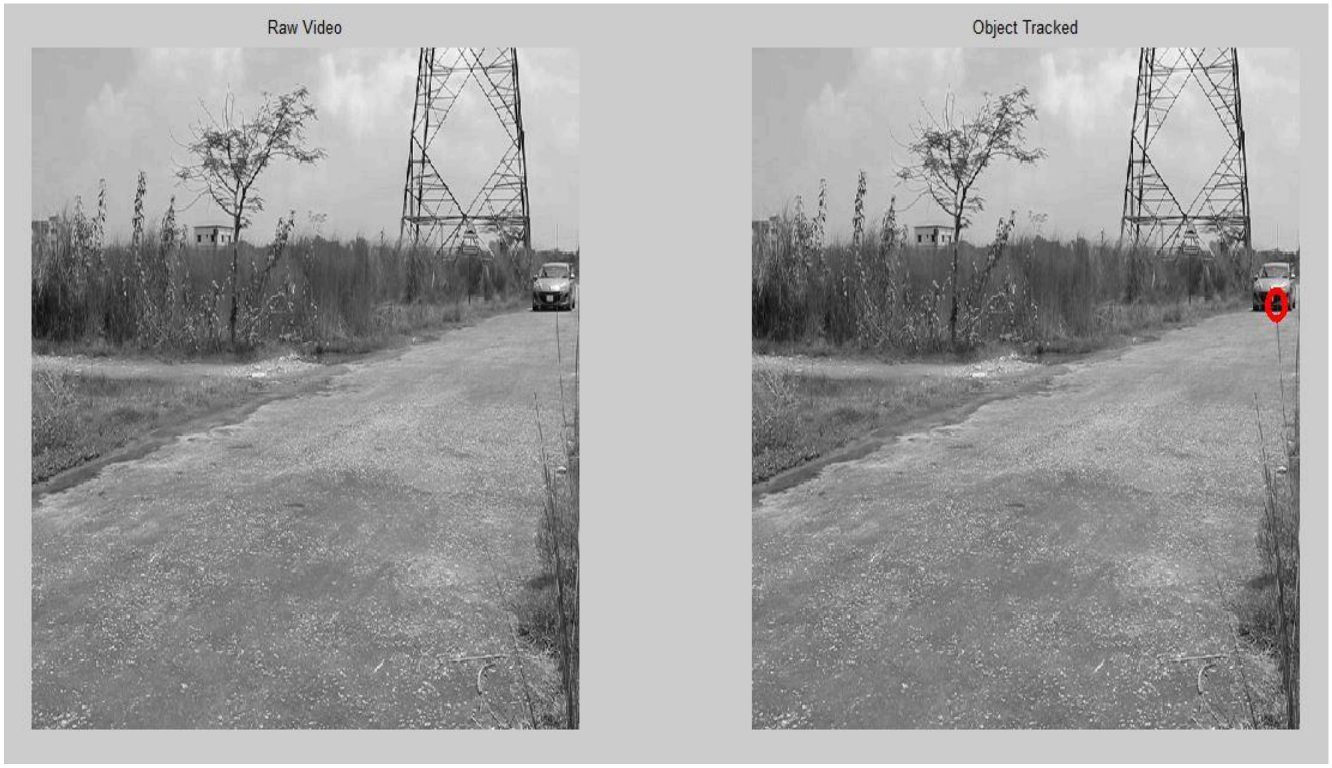

Figure-4.12: Result for frame number: 450

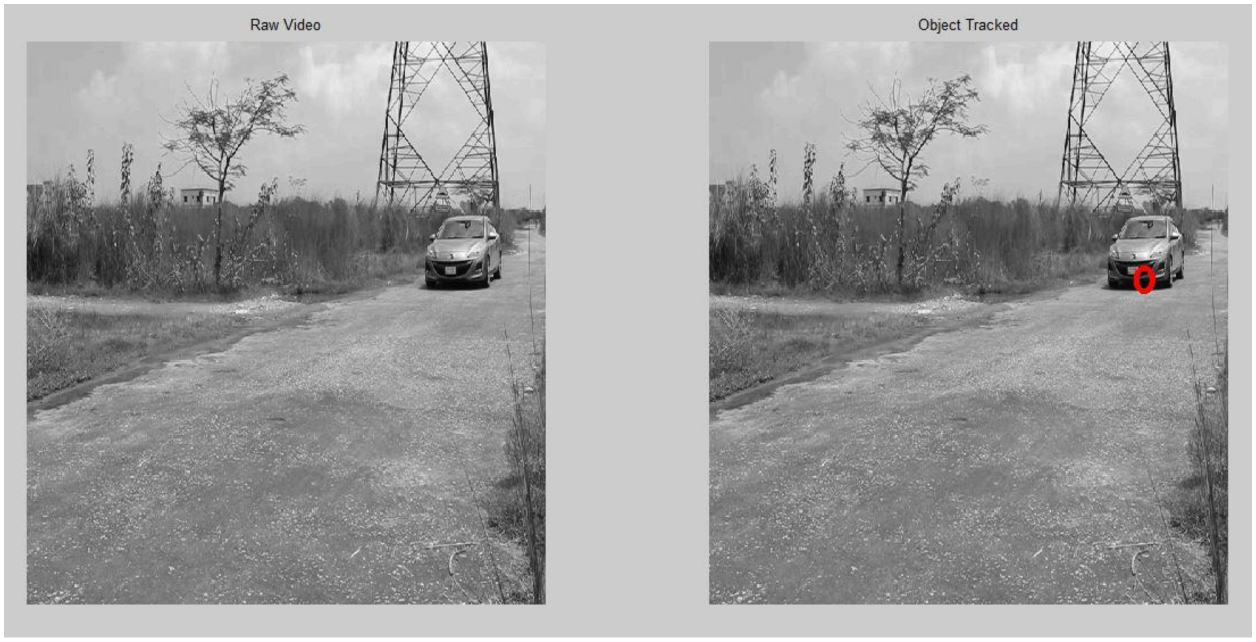

Figure-4.13: Result for frame number: 550

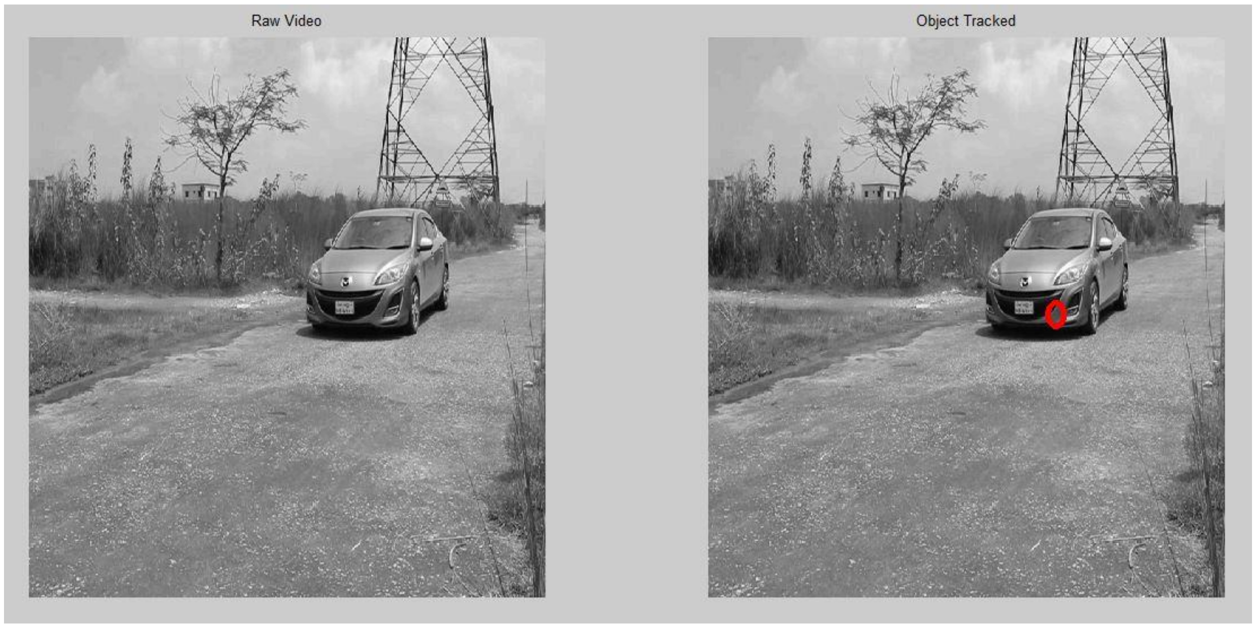

Figure-4.14: Result for frame number: 600

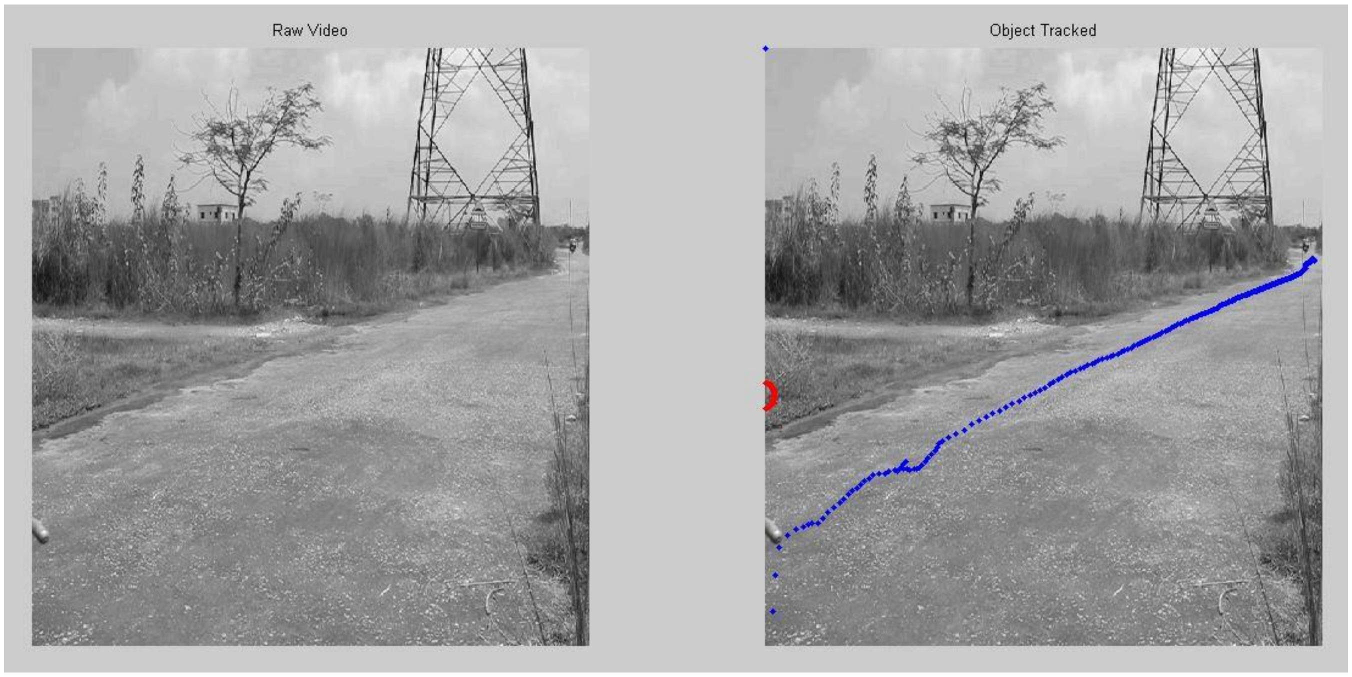

Figure-4.15: Traveling path of the car in frame position

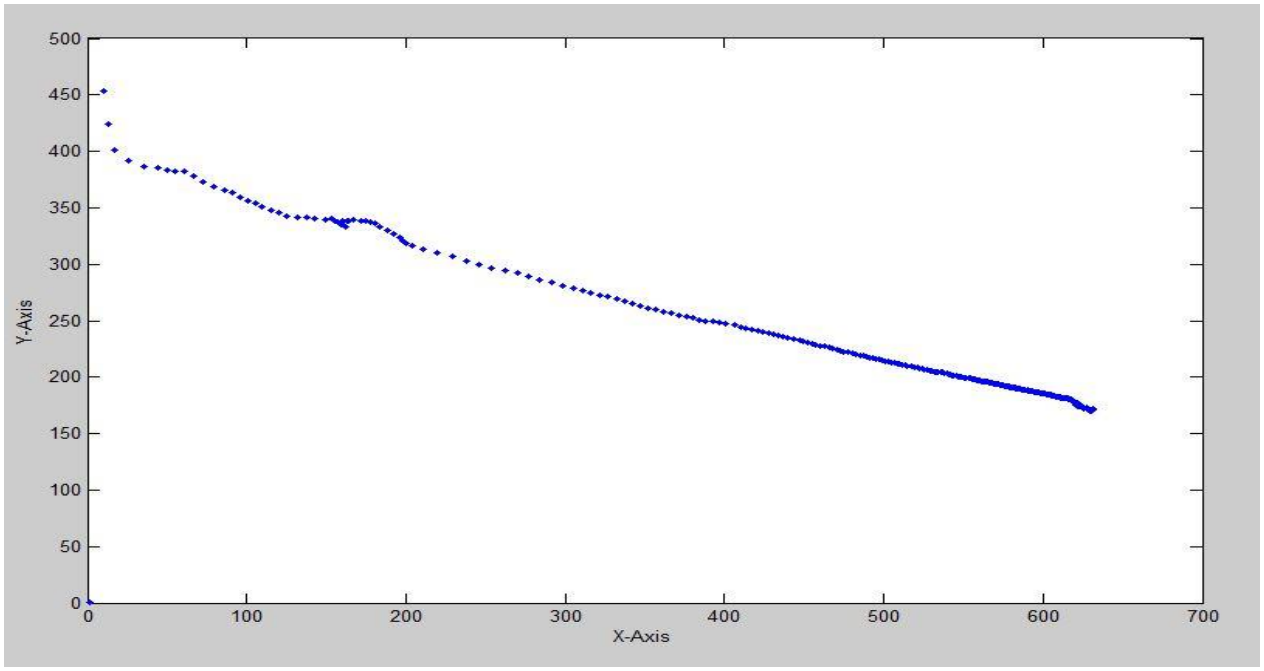

Figure-4.16: Graphical representation of object path (Car path)

Our proposed method only works when the video is collected from a stationary camera. Here, we used background subtraction method for object detection .So, if we collect the video from moving camera then the proposed method couldn't works.

When we detect the object we have to know the centroid value of the object. This centroid value is used for measurement value for object tracking. If there is no object on this frame, then we set the value of the centroid is 1. In this case, if we pick random value then it makes a noise measurement value in centroid value.

We couldn't show the detected object and it's centroid in the same image. So, we detect the object first time and then find the centroid in the second. Then we combined these two image for showing the centroid point on the detected object.

Here, in the above picture we show the object tracking in different frame and showing the object traveling path on the image. Also make a graphical representation figure for the object traveling path.

After analyzing the following result of the tracking method, we conclude that our algorithm is able to track any single moving object from the stationary video and it also able to shown the traveling path of that moving object. This algorithm is works in indoor environments as well as outdoor environments.

Chapter 5
Conclusion and Future Work

5.1 Conclusion

In this paper, an accurate method for moving object detection and tracking capability has been presented with the merits and demerits of Kalman filter with various filtering methods that existed in the current demerits like lack real time, reliability etc. The system works on videos of indoor as well as outdoor environment taken using static camera under moderate to complex background condition. This implemented module can be applied to any computer vision application for moving object detection and tracking and mostly used in video surveillance applications.

5.2 Future Work

Our work can be used in artificial intelligence to detect the path of any moving objects because robot's sensor work fast but it must be accurately to be perfect to see the moving objects path. So tracking using Kalman filter must be useful there.

As Kalman filter algorithm predict the next position of any mobile object so it can be used to detect the object position which is invisible for a certain time.

This work can be extended to track multiple moving objects. We also planning to track our own vehicle with respect to other vehicle to avoid major accidents. Car collision avoidance is very similar to the target tracking problem. We are interested in predicting the own car and other object's future position. The complete system with capabilities of detection and tracking can be used for applications domain like security, human computer interaction, scene analysis and activity recognition, event detection etc.

References

[1] THOTA VINOD RAJA, M. TIRUPATHAMMA,” Object Detection and Tracking in video using Kalman Filter”, International Journal of Research In advanced Engineering Technology, Volume 5, Issue 5 OCT 2016.

[2] http://ieeexplore.ieee.org/document/7528889/

[3] Massimo Piccardi,”Background subtraction techniques: a review”, 2004IEEEInternational Conference on Systems, Man and Cybernetics.

[4] Pravin A. Dhulekar, Vaishnavi D. Hire, Mandar S. Agnihotri,”Moving Object Tracking Using Kalman Filter”,IJAFRC Volume 2, Issue 1, January 2015. ISSN 2348 – 4853.

[5] Hitesh A Patel, Darshak A thakure, “Moving Object Tracking Using Kalman Filter”, International Journal of Computer Science and Mobile Computing, April 2013, pg.326-332.

[6] http://studentdavestutorials.weebly.com/basic-image-processing-with-matlab.html

[7] http://docs.opencv.org/trunk/d1/dc5/tutorial_background_subtraction.html

[8] https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.html

[9]https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.html

[10] B. D. O. Anderson and J. B. Moore, Optimal Filtering. New York: Dover, 2005.

[11] Ramsey Faragher.”Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation”,IEEE SIGNAL PROCESSING MAGAZINE [**128**] SEPTEMBER 2012.

[13] http://ieeexplore.ieee.org/document/7528889/

[14] https://www.youtube.com/channel/UCUxiT_SKEUs1oWT6i9P3vPQ

[15] http://studentdavestutorials.weebly.com/object-tracking-2d-kalman-filter.html

Appendix

In this section we put the matlab implementation code of our algorithm. We divided our implementation into two steps. (1) Image processing section and (2) Kalman filter Apply.

(1) Image processing section:

```
%% This code for process the images for object tracking .

clear all;  % free up system memory
close all;  % removed specified figure

% creat the frames address as a directory
directory = 'E:\MATLab2013\bin\ALL_TRACKING\Person_trk\person (26-Jul-17 6-
18-32 PM)\';

% by using this command enter the directory
cd(directory);

% for get the list of frames
frame_list=  dir('*jpg');

%%Make average of background images

N = 30;  %num of frames to use for averaged background

%define image stack for averaging
image_avg = zeros(480,640,N);

fori = 1:N
```

```
image_temp = imread(frame_list(i).name); % for reading given image
image_avg(:,:,i) =image_temp(:,:,1); % use first dimension of the image

end

background_image = (mean(image_avg,3)); % take the average image
subplot(121);
imagesc(background_image)

subplot(122);
imagesc(image_avg(:,:,1))
colormap(gray)

%%gaussian filter initialization

hsize = 20;
sigma = 10;
gaussian_filter = fspecial('gaussian',hsize, sigma); %create 2D Gaussian filter

subplot(121);
imagesc(gaussian_filter)
subplot(122);
mesh(gaussian_filter)
colormap(gray)


%for making the coordinate locations more visible.

SE = strel('diamond', 7);
```

```
% initialize the variable that will store the object locations(x,y)

Track_data = zeros(length(frame_list),2);

%% iteratively find the object

fori = 1:length(frame_list)

image_temp = double(imread(frame_list(i).name)); % load the image and convert it
into double for computation.
image = image_temp(:,:,1); % reduce image in first dimension

subplot(221);
imagesc(image);
title('Original');

  %subtract background from the image
subtract_image = (image - background_image);
subplot(222);
imagesc(subtract_image);
title('Background subtracted');

  %gaussian filtering for image
gaussian_image = filter2(gaussian_filter,subtract_image,'same');
subplot(223);imagesc(gaussian_image);
title('Gaussian filter smoothed');

  %threshold the image based on histogram
subplot(224);
hist(gaussian_image(:)); % create histogram
threshold_image = (gaussian_image< -60);
```

```
subplot(224);
imagesc(threshold_image);
title('Thresholded');


   % Tracking object and finding center of the object

   [x,y] = find (threshold_image); % find the nonzero value of threshold image

if ~isempty(x)

Track_data(i,:) = ceil([mean(x) mean(y)]+1); % ceiling to avoid zero indices

else

Track_data(i,:) =1;

end


   % make binary image with single coordinate of object

object_image = zeros(size(threshold_image));  % crate array with zeros
object_image(Track_data(i,1),Track_data(i,2)) = 1;  % make the object centroid is 1


object_image = imdilate(object_image, SE);  % to make more visible centroid
subplot(224);
imagesc(threshold_image + object_image);  % show object with centroid
title('thresholded and point extracted');
```

```
        pause(0.0001)
        end


        %save the coordinates value
        save('Track_data_store.mat', 'Track_data');
```

(2) **Kalman filter application :**

```
%%This code is for applying kalman filter.


clear all;  %free up system memory

close all;

clc;


% creat the frames address as a directory

directory = 'E:\MATLab2013\bin\ALL_TRACKING\Person_trk\person (26-Jul-17 6-18-32 PM)\';


%by using this command enter the directory

cd(directory);
```

```
% for get the list of frames

frame_list=  dir('*jpg');


% for load tracking data

load('Track_data_store.mat');


%% define variables for kalman filter


dt = 1;  % sampling rate


Starting_frame = 10; % starting frame


u = .005;  % define acceleration magnitude


Q   =   [Track_data(Starting_frame,1);    Track_data(Starting_frame,2);    0;    0];    %    initized-
[positionX; positionY; velocityX; velocityY] of the object


Q_estimate = Q;  % estimate of initial location estimation of where the object is.


Accel_noise_mag = .1; % process noise


noise_x = 1;  % measurement noise(x axis).

noise_y = 1;  % measurement noise(y axis).
```

```
Ez = [noise_x 0; 0 noise_y];

Ex = [dt^4/4 0 dt^3/2 0; ...
     0 dt^4/4 0 dt^3/2; ...
dt^3/2 0 dt^2 0; ...
     0 dt^3/2 0 dt^2].*Accel_noise_mag^2; % convert the process noise into covariance matrice

P = Ex;  % estimate of initial object position variance

%% Define update equations in 2-D

A = [1 0 dt 0; 0 1 0 dt; 0 0 1 0; 0 0 0 1]; % state update matrice
B = [(dt^2/2); (dt^2/2); dt; dt];
C = [1 0 0 0; 0 1 0 0];  % this is measurement function C, that apply to the state estimate Q to get expect next measurement

%%Initialize result variables

Q_loc = []; % actual object motion path
vel = [];  % actual object velocity
```

```
Q_location_measurement = []; % the object path extracted by the tracking algorithm



%% Initize estimation variables


Q_location_estimate = []; % position estimate

velocity_estimate = []; % velocity estimate

P_estimate = P;

predic_state = [];

predic_var = [];


r = 15; % define radius for the plotting circle

j=0:.01:2*pi; % for make the plotting circle



for t = Starting_frame:length(frame_list)


   % loading image

img_tmp = double(imread(frame_list(t).name));

img = img_tmp(:,:,1);

subplot(121);imagesc(img);

axis off
```

```
title('Raw Video');

   % Load the given tracking

Q_location_measurement(:,t) = [ Track_data(t,1); Track_data(t,2)];   %measurement value
from image processing

   %% Applying of kalman filter

   % Predict next state of the object with the last state and predicted motion.
Q_estimate = A * Q_estimate + B * u;
predic_state = [predic_state; Q_estimate(1)] ;

   % Predict next covariance
   P = A * P * A' + Ex;
predic_var = [predic_var; P] ;

   % Kalman gain
   K = P*C'*inv(C*P*C'+Ez);% P=P(predict),Ez=measurement noise covarience

   % Update the state estimate.
```

```
if ~isnan(Q_location_measurement(:,t))  % Not a number(not zero)check


Q_estimate = Q_estimate + K * (Q_location_measurement(:,t) - C * Q_estimate); %Updated
real position


end




   % Update error covariance estimation.
   P =  (eye(4)-K*C)*P;


   % Store data
Q_location_estimate = [Q_location_estimate; Q_estimate(1:2)];
velocity_estimate = [velocity_estimate; Q_estimate(3:4)];




   % Plot the images with the tracking
subplot(122);
imagesc(img);
axis off
colormap(gray);


hold on;
```

```
plot(r*sin(j)+Q_estimate(2),r*cos(j)+Q_estimate(1),'.r'); % The kalman filtered tracking

   %plot(r*sin(j)+Q_location_measurement(2,t),r*cos(j)+Q_location_measurement(1,t),'.g');
% The measurement tracking

hold off;

title('Object Tracked');

pause(0.0001)
end

for t = 1:length(frame_list)

hold on;

plot(Track_data(:,2),Track_data(:,1),'.b');

hold off

end
```

```
figure;

plot(Track_data(:,2),Track_data(:,1),'.b');

xlabel('X-Axis');

ylabel('Y-Axis');

title(' Object traveling path') [15].
```