Chapter 3

Approach for Tracking of Mobile Robot with Vision Sensor

3.1 Introduction

Robotics research has been using vision sensors in a wide variety of applications such as surveillance, vehicle navigation and autonomous robot navigation. These applications involve detection and tracking of the moving object accurately in real time. Accurate detection i.e. localizes the appearance of object in the current frame and tracking is prerequisite for successful navigation in real time environments, particularly when global models are available such as maps, topological descriptions etc. The basic strength of vision sensor lies in solving the detection and tracking problem of captured image or video of the environment and hence building of a real-time map of its surrounding environment through which the mobile robot is directed to navigate through desired path. The problems related to object localization, object recognition, object presence classification, and pose estimation can be solved using different object detection techniques [Astua et al. (2014), Deori and Thounaojam, (2014)]. For above activities, object detection method should be robust and accurate to detect the moving object because the mobile robot would be used for realistic navigation [Jung & Sukhatme, (2010)]. There has been in-depth research on computer vision for mobile robot navigation since it offers relatively large amount of environmental information from an image or video which can be extracted [Culler and Long, (2016)]. Detection and tracking of mobile robot with vision sensor facilitates passive sensing of the environment and provide valuable information about the scene that is unavailable through other sensors. There is no single algorithm which can meet all kinds of robot path tracking control. Many scholars have put forward the corresponding path tracking control algorithm according to different robots and environment condition [Wu et al. (2013), Luo et al. (2015)]. Tracking of mobile robot is challenging due to poor illumination, shadows and

parallax effect. In order to reduce the above effects, reasonable tracking accuracy with low false alarm and missing detection rates are necessary. Therefore, Kalman Filter process is integrated into the online tracking process to improve the tracking performance. Therefore, the vision based navigation offers promising results considering the limitations of existing systems like inertial sensors, GPS, range sensors [Coito et al. (2014)]. In this work the focus is to track a mobile robot which was detected earlier using Viola Jones algorithm. A new framework has been proposed to solve the tracking challenges such as noise in an image, difficult object motion, imperfect object occlusions and complex objects structures; which is based on KLT and Kalman filtering approach. Here the aim is to infer the status of mobile robot which is being tracked, to estimate its current state and to make predictions about its future states. This technique has been used in the present work to operate a mobile robot autonomously and track it through the desired path. A detailed error analysis on track estimator based on ground truth data for KLT and Kalman filter have been provided. In the experimental evaluation, the implementation of above method provides reliable detection and tracking estimation that can handle a variety of different conditions. Consequently, the robot can communicate the information with controller for navigation.

The rest of this chapter is organized as follows: Section 3.2 gives a systematic method to detect and track mobile robot. In Section 3.3, the system descriptions and experimental setup is explained, subsequently in Section 3.4, experimental results are analyzed and presented. The conclusions are given in Section 3.5.

3.2 Methods to Detect and Track Mobile Robot

In this work a systematic method combining Viola Jones, KLT and Kalman Filter based algorithms is discussed to track and detect a mobile robot.

The steps used to detect and track a mobile robot are discussed below:

Step 1 Opens a mobile robot image and place a bounding box around the mobile robot by rescales to $m \times n$ pixels then Saves the rescaled mobile robot as an image. (start of V-J Algorithm)

Step 2 Positive and Negative images are captured and used as input for classification.

Step 3 Each stage in the cascade was trained using a Positive and Negative images (with specified Region of Interest).

Step 4 Threshold values of false alarm rate and number of cascading stages are chosen for better detection of mobile robot.

Step 5 The classifier returns the coordinates of the corners of the bounding box, which encloses the detected mobile robot. (end of V-J Algorithm)

Step 6 Initialize the tracking process to specify the initial locations of the points and the initial moving frame. (start of KLT Algorithm)

Step 7 The frames of the detected mobile robot within the bounding box are scanned for feature points to initialize the tracking process to specify the initial locations. The output feature points contain an $M\times2$ array of $[x, y]$ coordinates that correspond to the new locations in the input frame.

Step 8 The eigenvalue criterion is used to select the corners against the vague features in window, where the intensity patterns are very irregular. The background, as well as the relatively uniform areas in window, contains no features.

Step 9 To determine λ (predetermine threshold), first measure the eigenvalues for images of a region of approximately uniform brightness, taken from the camera for tracking.

Step 10 If the number of feature point decrease below the minimum eigenvalue then the detection process goes to step 8 and repeats the process.

Step 11 A tracker object is created which read, displays and tracks these feature points through the subsequent frames as long as a minimum number of feature points are available. (end of KLT Algorithm)

Step 12 The measured location of the mobile robot is taken as the centroid of the detected bounding box which is used as measured input to the Kalman filter. (start of Kalman Filter Algorithm)

Step 13 The location of the specified path is used as the predicted input to the Kalman filter.

Step 14 An optimal estimate of the current location of mobile robot is determined using the Kalman filter. (End of Kalman Filter Algorithm)

The details of each of the algorithms i.e. Viola Jones, KLT and Kalman Filter, used for above task are discussed in subsequent sections.

3.2.1 Viola Jones Algorithm

Object detection proposed by Viola and Jones is the robust approach for real-time object detection [Viola & Jones, (2001)]. This algorithm has four important stages: (i) Haar feature Selection, (ii) Creating an Integral Image, (iii) Adaboost Training and (iv) Cascading classifiers [Felzenszwal et al. (2010), Ehsan et al. (2015), (Bineesh & Simon, 2012), Yang et al. (2013)]. The integral image is a step for quick and efficient calculation for the sum of intensity values in a rectangular subset of an image [Guennouni et al. (2015)]. The integral image is defined as:

$$ii(x, y) = \sum_{x' \le x, y' \le y} i(x', y') \tag{3.1}$$

where $ii(x, y)$ the integral is image and $i(x', y')$ is the original image. This computational advantage enabled scaling the features for multi-scale detection at no additional cost because it requires the same number of operations despite of size. This approach utilizes the Adaboost algorithm which selects a sequence of rectangle features that indicate the presence of a mobile robot. Here the weak learner is designed to select the feature which best separates the positive and negative images. A weak classifier $h(x, f, p, \mu)$ is defined as:

$$h(x, f, p, \mu) = \begin{cases} 1 & if\ pf(x) \le p\mu \\ 0 & otherwise \end{cases} \tag{3.2}$$

where f is a feature from the huge set spanning different sizes of the Haar like features, p is a polarity indicating the direction of the inequality, μ is a threshold and x is a training sub window of size $m \times n$ pixel. The Adaboost is used both to select features and to train the classifier.

The cascade of classifiers [Shaikh et al. (2014)] is used to combine increasingly more complex classifiers successively which allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions. The object detection method belongs to the single detection window method based on applying an object detector for all possible sub-windows in a given image. In [Zhu et al. (2006)], an efficient detector applicable to videos was built using a cascade of Adaboost classifiers relying also on Haar descriptors but extracted from spatio-temporal difference. It was extended in [Dalal and Triggs, (2005)] to videos using histograms of differential optical flow features in addition to HOG. The HOG features focuses on the structure or the shape of an object. The HOG feature [Tian et al. (2013)] has been used successfully for object detection, which allows real-time detection comparable to the rectangular Haar-like detectors used by

Viola-Jones. HOG feature have more discriminative capacity than Haar features and covariance matrix. The HOG features are chosen to provide a good representation of object contour, invariant to illumination changes and small image movements and can be computed in a constant time. The HOG features are calculated by taking orientation histograms of image intensity in a local region. Integral image method for HOG features with fixed cell size is used for detection. In the present work, the goal is to detect and track the mobile robot in real time using captured images from stationary camera.

3.2.2 Kanade-Lucas-Tomasi Algorithm

Tracking objects can be complex due to loss of information caused by projection of the 3D world on a 2D image, noise in images, complex object motion, partial and full object occlusions, scene illumination changes, and real-time processing requirements etc. In general, many tracking algorithms use a combination of visual features like color, edges, optical flow and texture. In order to avoid tracking of all pixels in the resulting image and within a given foreground object, many techniques for tracking based on a limited set of feature points have been proposed in literature. Out of these techniques Kanade-Lucas-Tomasi (KLT) method has been chosen [Morlier & Michon, (2010)]. The KLT tracks an object in two steps; it locates the trackable features in the initial frame and then tracks each one of the detected features in the rest of the frames by means of its displacement [Bernes et al. (2014)]. In this work, the selected trackable features is used and subsequently the whole set of features are tracked together instead of tracking each feature separately. Stages in the cascade are constructed by training classifiers using Adaboost and then adjusting the threshold to minimize false negatives.

The aim is to track a region of interest in the image plane. The shape of this region is fixed a priori through the definition of a zero-centered window W. It can be an ellipse or a rectangular box as shown in Figure 3.1. In this case, there is no restriction on the class of shapes that can be used. Image gradients provide information about the linear intensity of image texture; thus the selection of "trackable features" can rely on texture. Texture pattern exists when multiple pixels in a certain area have different distinguishable values [Yilmaz et al. (2006)]. The features to be tracked can be precisely defined by the texture within a finite size window. The window size would determine the number of features detected. For each pixel in the window, the pixel gradient values φ_x, φ_y are calculated to form a 2×1 gradient vector, $\varphi = \begin{pmatrix} \varphi_x & \varphi_y \end{pmatrix}^T$.

Thus,

$$\varphi\varphi^T = \begin{bmatrix} \varphi_x{}^2 & \varphi_x\varphi_y \\ \varphi_x\varphi_y & \varphi_y{}^2 \end{bmatrix} \tag{3.3}$$

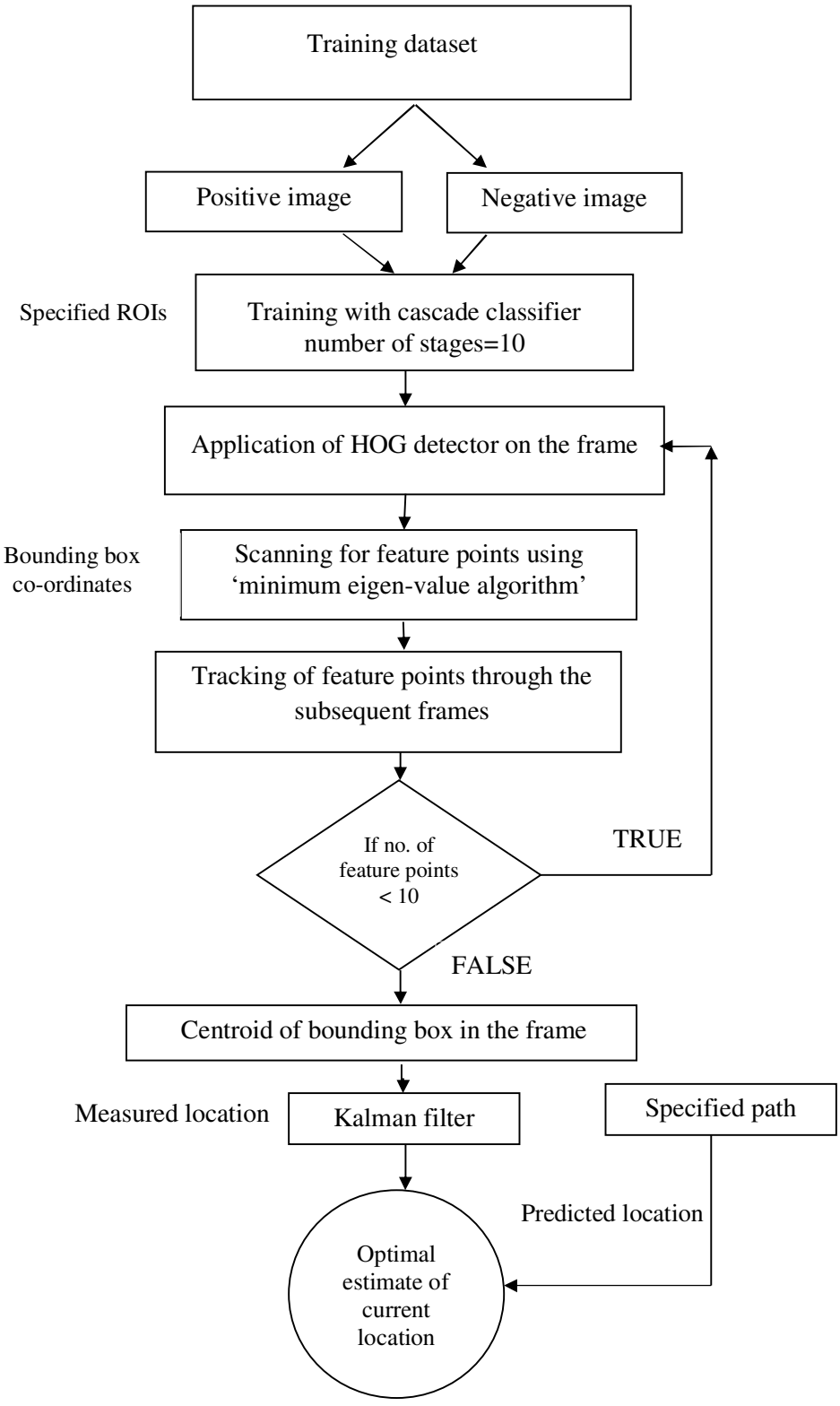


Figure 3.1: Flowchart for mobile robot detection, tracking and denoising using Kalman filter

By considering all pixels within the window, matrix *w* is defined as follows:

$$w = \sum_{i=1}^{k} \sum_{j=1}^{k} \phi_{ij} \phi_{ij}^{T} \tag{3.4}$$

Equation (3.4) forms the first part of the KLT tracking equation. The matrix w contains pure texture information. By analyzing the eigenvalues of w, the texture in the window can be classified. The KLT algorithm has been used to continuously track the mobile robot with the help of the frames which results in the decreased number of feature points being tracked. The rationale behind the big enough eigen values is to detect the presence of a feature points. Thus, it is critical to set up a threshold for the eigen values, which depends on the environment where tracking is performed. For experimental purposes, the threshold for $\lambda 1$ and $\lambda 2$ were found by trial and error and set to 1000. This caused reduction in the number of feature points as many points become invalid after being tracked through several frames. To detect the presence of a feature point, it is important to determine when the eigenvalues are big enough. Thus it is critical to set up a threshold for the eigenvalues, which depends on the environment where tracking is performed. In addition, it is also critical to keep the number of features to a minimum level. This is to ensure a reduction of the amount of calculations performed by the system for real-time purposes. The detected features are represented by white (+) symbols on the mobile robot as shown in Figure 3.2. Once the features are extracted, the tracking step begins. In tracking, the location of the object is determined in the scene by means of its detected features. The reason for tracking is to estimate the target objects in video sequences over an interval. However, issues related to tracking undermine the performance and the efficiency of tracking algorithms. The two fundamental issues associated with tracking are (i) determination of the spatial location of target object; and (ii) variations in lighting and sensor related noise. Additional issues may arise when the moving object is occluded and separation if any. Due to these issues, the contextual information of moving object is lost which results in uncertainty during tracking [Morlier et al. (2010), Yang et al. (2013)].

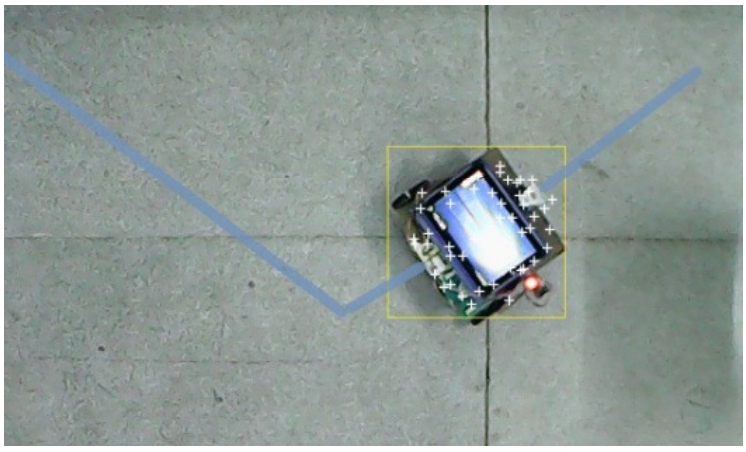


Figure 3.2: Detected feature of mobile robot

3.2.3 Kalman Filter Algorithm

The Kalman Filter (KF) has been widely used for mobile robot navigation [Casanova et al. (2008), Li et al. (2010), Chen (2012)] which is a set of mathematical equations that provides an efficient computation, recursive, and solution to the method of least-squares. In tracking systems, there are two problems which must be analyzed: prediction and correction. Prediction problem: predict the location of an object being tracked in the next frame which identifies a region of interest where the probability of finding object is high. Correction problem: identifies the object in the next frame within designated region. A well-known solution for prediction is Kalman filter, which is a recursive estimator of state of a dynamic system [Ghandour et al. (2015)].

The Kalman filter is an estimator that provides an efficient recursive method to estimate the state of a linear process, in a way that minimizes the mean of the squared error [Welch & Bishop (2006)]. The Kalman filter uses the time update (prediction) and the measurement update (correction). Time update is to advance the state based on state equation until the next measurement is obtained. Measurement update is to incorporate the measurement from sensors based on measurement equation. The mobile robot is supposed to move in a 2D coordinate system. So Kalman filter estimates the position, (*x*, *y*) in the frame, of object to be tracked. Since the standard Kalman filter uses one correct measurement, data association should be considered to classify true measurement and false measurements. Therefore, Kalman filter is configured as follows:

$$x_k = Ax_{k-1} + w_k \tag{3.5}$$

$$z_k = Hx_k + v_k \tag{3.6}$$

where, $x_k = \begin{bmatrix} p_x & p_y & v_x & v_y \end{bmatrix}^T$ $A = \begin{bmatrix} 1 & 0 & t & 0 \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$, $H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

p_x, p_y represent the center position of x-axis, the center position of y-axis and v_x, v_y are the velocity of x-axis and y-axis. Matrix A represents the transition matrix, matrix H is the measurement matrix, and t is the time interval between two adjacent frames. w_k, v_k are the Gaussian noises with the error co-variances Q_K and R_k. These noise values are entirely dependent on the system that is being tracked and adjusted empirically. The step by step implementation of Kalman filter is discussed below.

Step 1: Time update of the state estimate is

$$x_{k|k-1} = Hx_{k-1|k-1} \tag{3.7}$$

Step 2: The predicted measurement

$$z_{k|k-1} = Hx_{k-1} \tag{3.8}$$

Step 3: The Time update of the state error covariance

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q_k \tag{3.9}$$

Step 4: The data association process and determination Kalman Gain:

$$K_k = P_{k-1|k-1}H^T(HP_{k|k-1}H^T + R)^{-1} \tag{3.10}$$

Kalman gain depends on the accuracy of a measurement. If an accuracy of the measurement is high, the Kalman gain has high value. Otherwise, the Kalman gain has relatively low value.

Step 5: Measurement update of the state error covariance

$$P_{k/k} = (I - KH)P_{k/k-1} \tag{3.11}$$

where, I is a 4×4 unit matrix.

Step 6: Measurement update of the state estimate

$$x_{k|k} = x_{k|k-1} + K_k(z_k - z_{k|k-1}) \tag{3.12}$$

3.3 System Description

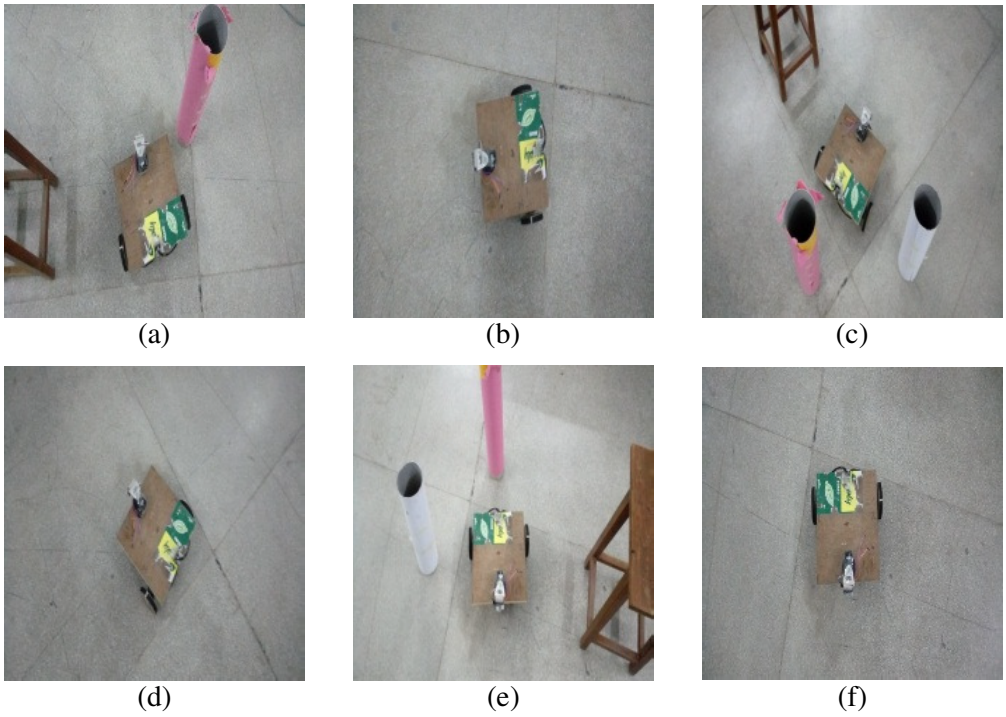
The experiments have been carried out using a three wheeled mobile robot. This mobile robot consists of two independently driven wheels in the rear, one unpowered unidirectional wheel in the front. The servo motor (Dynamixel) motor is chosen because it has many advantages compared to the other motors like dc motor, BLDC motor. Enhanced speed, a good dynamic response, longer operating life, noiseless operation, and high torque ranges etc. are few to mention [User's manual Dynamixel Ex-106 ROBOTIS]. The rear wheels are driven by EX-106 Dynamixel motor (Appendix A1) with No load speed: 69.9 rpm at 18.5V. These wheels are actuated by using on board computer Surface-3 with 1.6-GHz quad-core Intel processor, 4GB of RAM and 64GB onboard storage with MATLAB 2015b installed

software. A reliable and fast algorithm is developed (Appendix A2) to follow all types of motions such as, forwards, reverses, right, and so on. A flexible code with 'distance' and 'Speed' as user input is made which makes the robot to move in either forward or reverse direction. Algorithm basically converts the distance and speed into decimal format of hexadecimal input readable for actuator. Distance is calculated based on user input, circumference and hexadecimal degrees for one turn. Speed setting is actually the torque control of motor and exceeding certain limit changes the direction of rotation which is effectively utilized in single code. The EX-106 Dynamixel motors are operated simultaneously by an USB2Dynamixel converter using a chain-wire link with a suitable battery. It has three configurations for RS-232, RS-485 and TTL type of protocols for communication. Dynamixel EX-106 with a unique ID is controlled by serial packet communication on a BUS and supports network such as RS 485 asynchronous serial communication. To control EX-106 with PC, the USB2Dynmixel converter is used [User's manual Dynamixel Ex-106 ROBOTIS].

To capture the moving scenes an USB based web camera (iball-roboK20) is mounted overhead which is 365cm from the surface. The 640×480 pixels images are acquired and the information is sent to the laptop with configuration Core i5 3470 3.2 GHz/4GB-250GB with Intel HD Graphics/4400 with windows operating system. The scene observed is setup with 2D planes patterns, from which the features are extracted and matched to estimate the current and target images. The acquired image data is processed using the MATLAB programming. During the navigation, the system performs the tracking of the image using KLT and Kalman filter algorithm.

3.4 Experimental Results

This work focuses on the detection and tracking of a mobile robot moving on a specified path. The process of feature points based object tracking of the mobile robot begins with its detection. The Viola-Jones object detection framework has been implemented. In this application, a window of size 4×4 pixels is used (8×8 and 16×16 pixels can be used but the number of detected features will decrease as the pixels sizes increase).



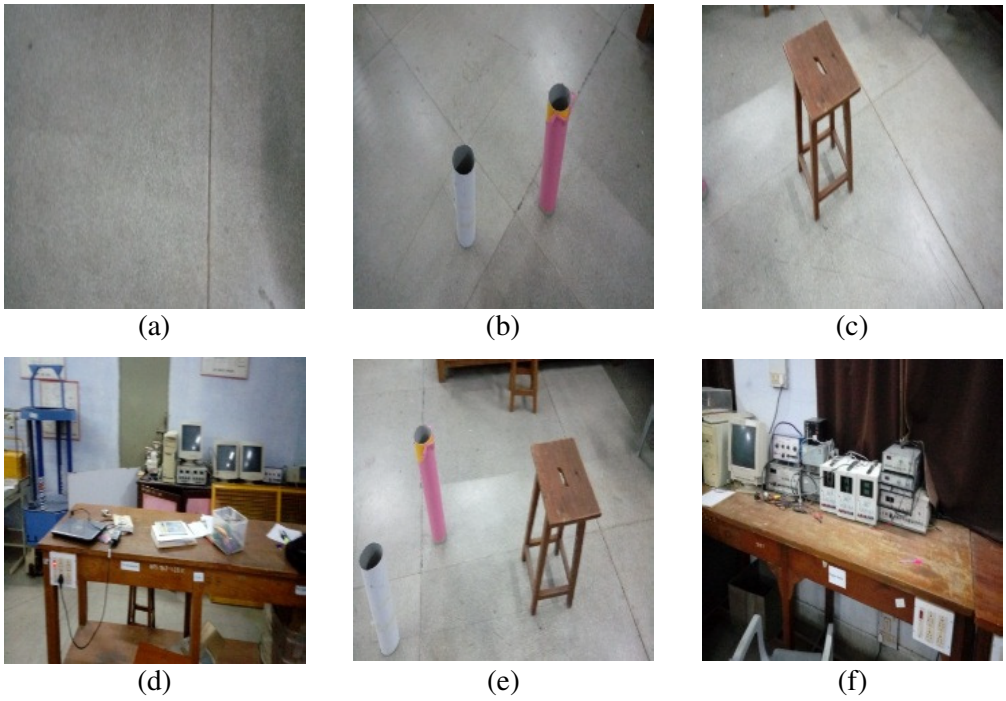
(a) (b) (c)

(d) (e) (f)

Figure 3.3 (a-f): Positive images for detection

To discriminate between the mobile robot and non- mobile robot image several feature images are used for training and those images will be selected as the stage classifier for the current stage. The Viola-Jones stage classifier training iteration ends on the false alarm rate reaching a predefined threshold. But for a certain set of training samples, the false alarm rate does not reach the predefined threshold based on this experiment. In the training process of the cascade classifier, approximately 200 images were added to reduce false positive rate to 0.2 which indicate a high detection of ROIs. The training set of mobile robot images was generated and classified into 'positive' and 'negative' images. The images which classified as 'positive' were the ones that contain the mobile robot Figure 3.3 (a-f).

The images which were devoid of the mobile robot were classified as 'negative' shown in Figure 3.4 (a-f). This figure depicts a few positive images, with Figure 3.4 depicting the negative ones. The training set had 62 positive images and 144 negative images. The above said detection framework was implemented with the help of Viola Jones library functions in MATLAB scripts. The results obtained after training met expectations and effectively improved the performance of its implementation in tracking of the mobile robot on the specified path using KLT.



(a) (b) (c)

(d) (e) (f)

Figure 3.4 (a-f): Negative images for detection

After the mobile robot was successfully detected, a yellow bounding box was positioned over the detected region for visualization. The position and size of this bounding box would be updated as it moves forward with the tracking procedure. The KLT algorithm tracks the mobile robot in two steps; it locates the trackable features in the detected bounding box region of the initial frame, and then tracks each one of the detected features in the rest of the frames by means of its displacement. The feature points being tracked were initially detected by the use of the minimum eigenvalue algorithm. The KLT algorithm continuously tracks the object through the frames which results in the decrease of the number of feature points being tracked. To detect the presence of a feature point, it is important to determine when the eigenvalues are big enough. Thus, it is critical to set up a threshold for the eigenvalues, which depends on the environment where tracking is performed. For experimental purposes, the threshold for λ_1 and λ_2 is set to 1000. The number of feature points decrease because many points become invalid after being tracked through several frames.

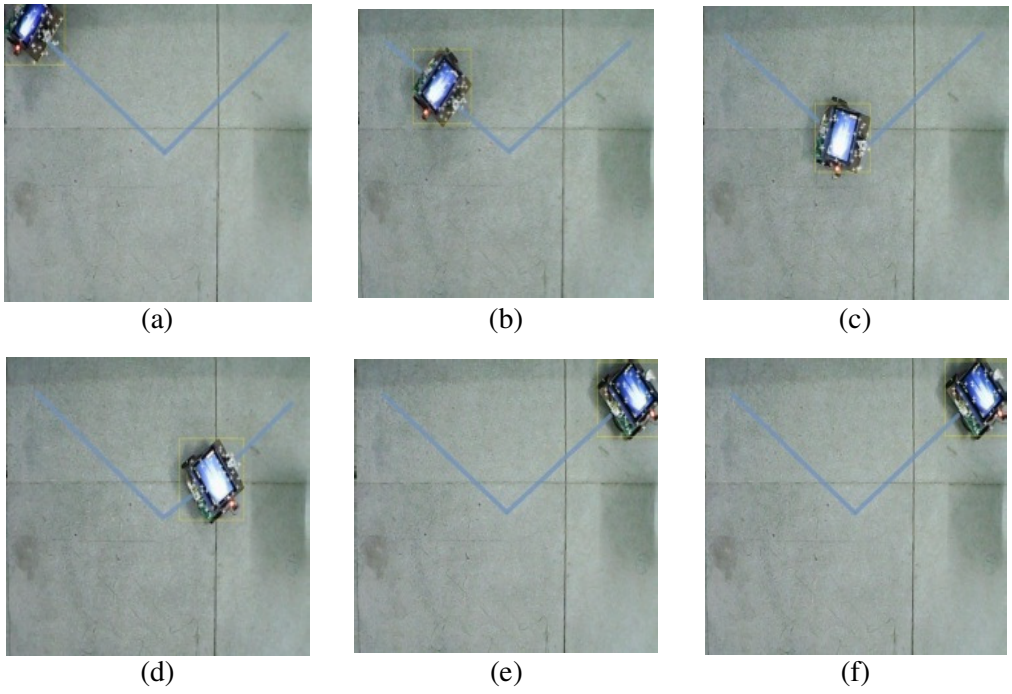


Figure 3.5 (a-f): Results of mobile robot tracking using KLT algorithm

The experimental brightness of mobile robot feature is constant over time. Number of mobile robot feature in the image plane moves in a similar manner since the velocity smoothness constraint. When the number of feature points being tracked becomes less than the set threshold, the detection algorithm gets activated again and then a new set of feature points are tracked again. The following Figure 3.5 (a-f) shows KLT tracking process, in which the 'yellow' bounding box is the detected object being tracked and the feature points being marked with '+'sign.

In this work, mobile robot being tracked is depicted visually by a surrounding bounding box in the shape of a 'yellow' rectangle. The size of the rectangle is determined according to the feature points detected near the object. The measured position (location) of the mobile robot in the filter is taken to be the centroid of the depicted rectangle in the image. The initial Kalman filter parameter values have been calculated experimentally with Q = 0.5, R = 0.6, A= 0.9. The motion vectors are used in depicting the 'motion' of the measured position of the mobile robot. Here motion vector depict the offset of the measured position of the mobile robot from the reference coordinates, which has taken as the upper left corner of the image. Figure 3.6(a-f) shows a case where the measured location (green circle) has suddenly

shifted upward because of the high density in the detected features at the top of the object. The Kalman predicted location (red circle) has accurately detected the sudden change and located the object at the right place using the information from previous frames. A plot of the *x*-coordinate and *y*-coordinate of the motion vectors is displayed. The distinction between the measured and filtered (estimated) path can be observed clearly.

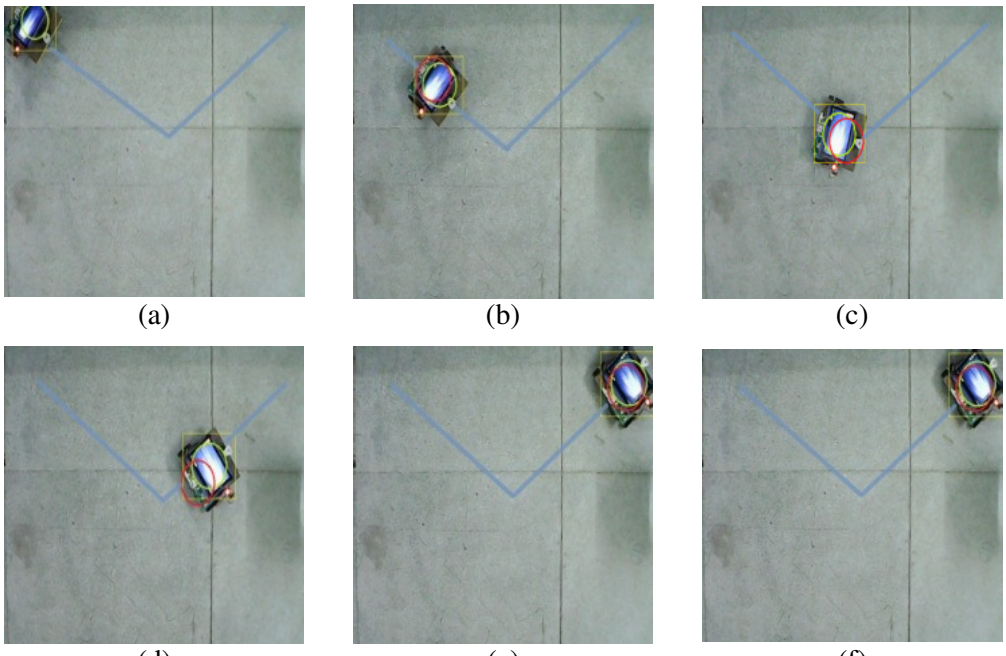


Figure 3.6 (a-f): Results after applying Kalman filter algorithm

The experimental map area on the ground is approximately $140\text{cm}\times200\text{cm}$. The resolution of the image of the map obtained is 480×640. So approximately, the side of each pixel in the image can be taken as 0.302085 cm on the ground. Using this assumption, the velocity vs. frames plot was found out from the experiments presented in Figure 3.6(a-f). The velocity refers to the displacement of the mobile robot through two consecutive frames. Instead of time, the velocity was found with respect to frames, because the frame rate of the camera was not constant throughout the experiment. The data from a five sets of experiments was taken to test the performance of proposed algorithms. The results of the computed xy co-ordinates of KLT tracking with respect to time and after Kalman filtering with respect to time are described below.

Error calculation: For the purpose of calculation of errors in experiments a mathematical model was formulated. For error calculation model, a self-determined error circle was taken. The Figure 3.7 illustrates this modeling procedure.

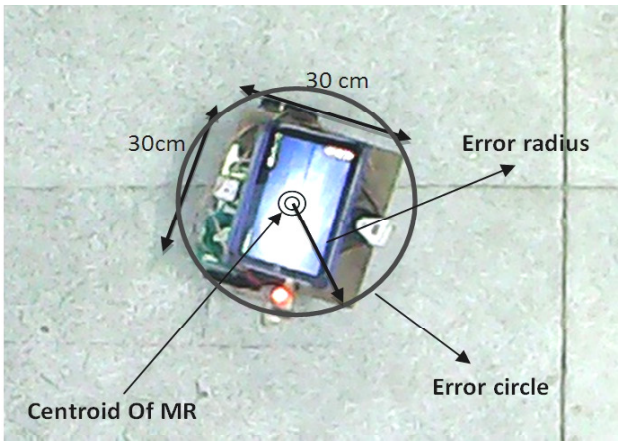
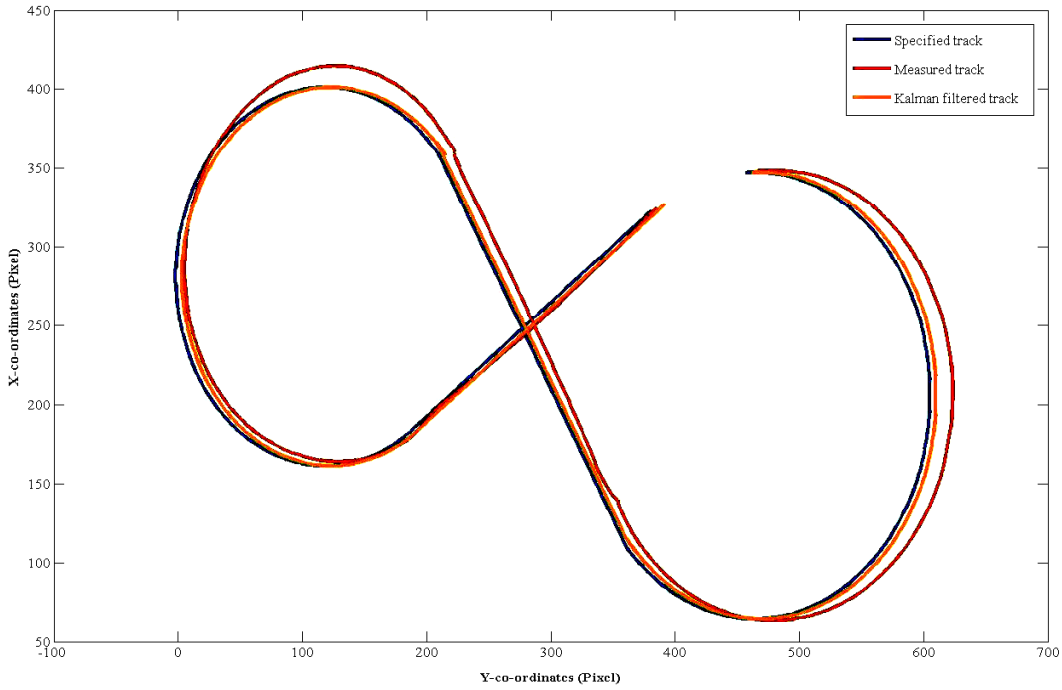


Figure 3.7: Cartesian coordinates of mobile robot and its error calculation

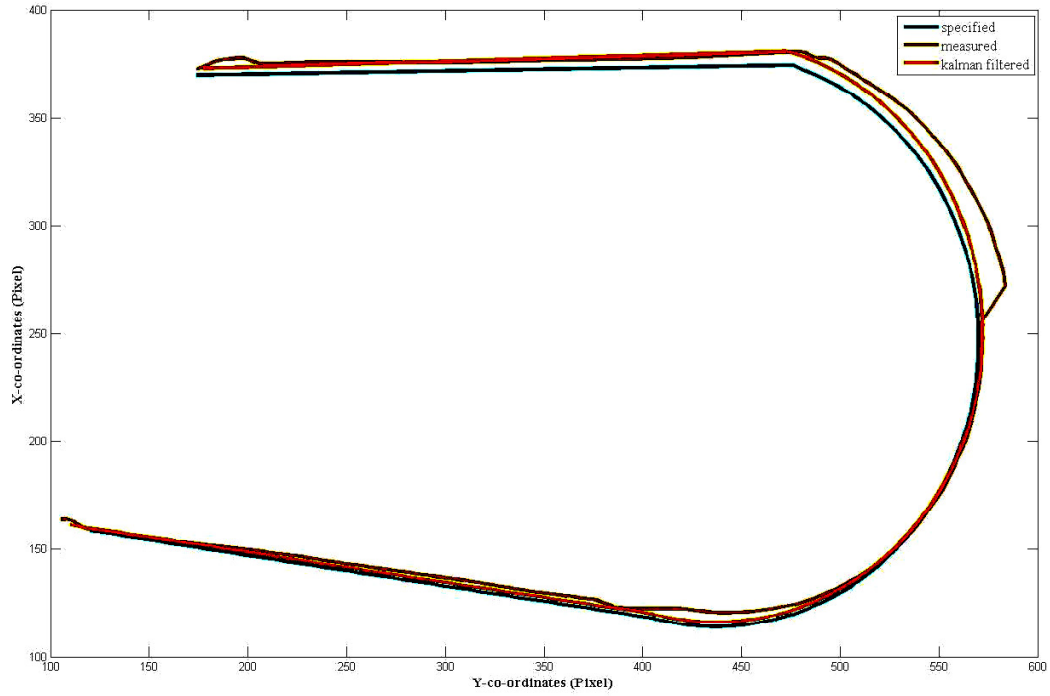
Here, mobile robot is considered as a square of side 30 cm. So, it's diagonal $(d) = 30\sqrt{2}$ cm. The self-determined 'error radius' as shown above is half of the diagonal. So, error radius $r = d/2 = 15\sqrt{2} = 21.2132\text{cm}$. It is calculated $1\,\text{pixel} = 0.091125\,\text{cm}^2$. So, one side of each pixel is $0.301869\,cm$. So the number of pixels in this 'error radius' = 21.2132/0.301869 = 70.273 pixels. KLT Algorithm provides features in the form of 3 matrices: x, y, and values with each having dimensions of n*f where 'n' is the number of features tracked and 'f' is the number of frames in the video. Each (i, j) of x and y matrices contain the coordinates or position of the feature in the image in a particular frame. If the KLT Measured coordinates are at the boundary of the error circle, then error = 100%. If the KLT Measured coordinates are at the centroid of the error circle, then error = 0%.

The results of the computed xy co-ordinates of KLT tracking with respect to time and after Kalman filtering are shown in Figure 3.8 (a-e). KLT tracker has some shortcoming owing to the lack of texture away from the central feature of the image thus feature points on image may not yield successful tracking. That is the reason why KLT tracking path is observed to be away from the specified path. Though this approach provides a reasonable measure of tracking accuracy, it does not explicitly derive the covariance from the image gradients. By applying Kalman filtering algorithm consistent covariance

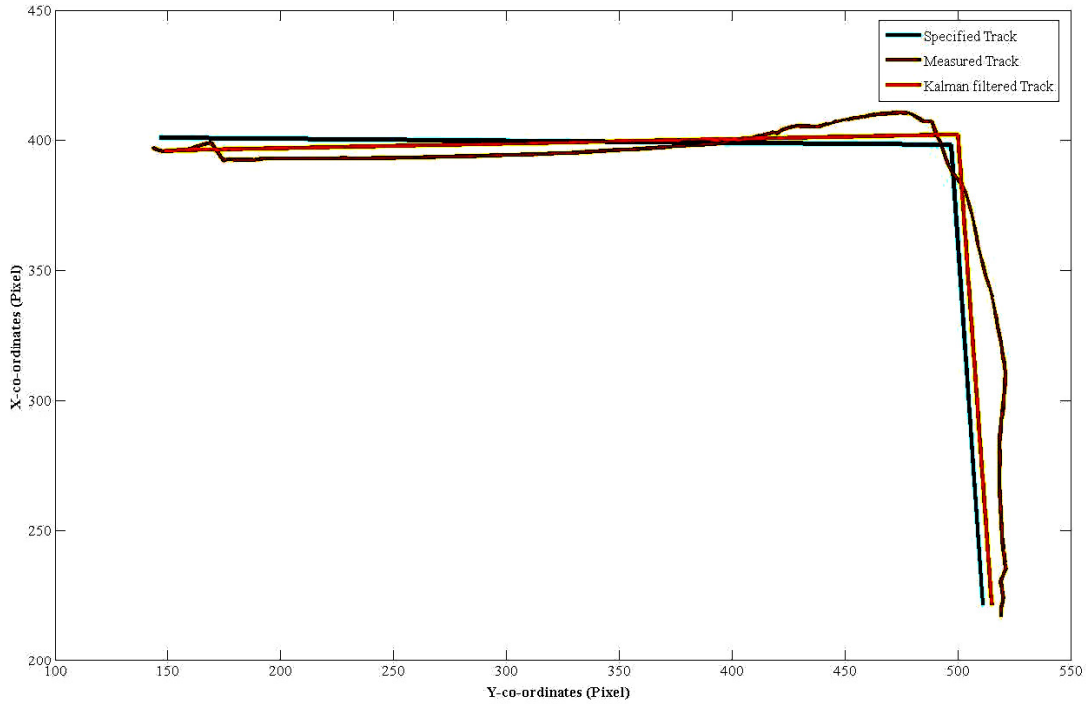
bounds are obtained. This implementation results in tracking of mobile robot path closer to the specified path.



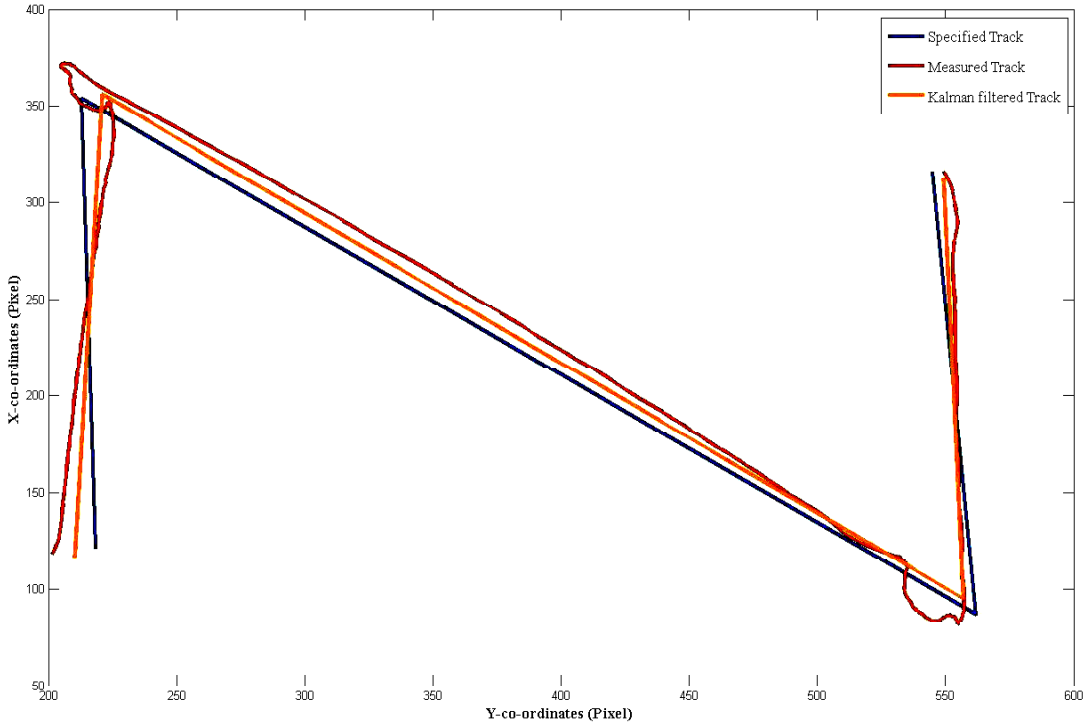
(a)



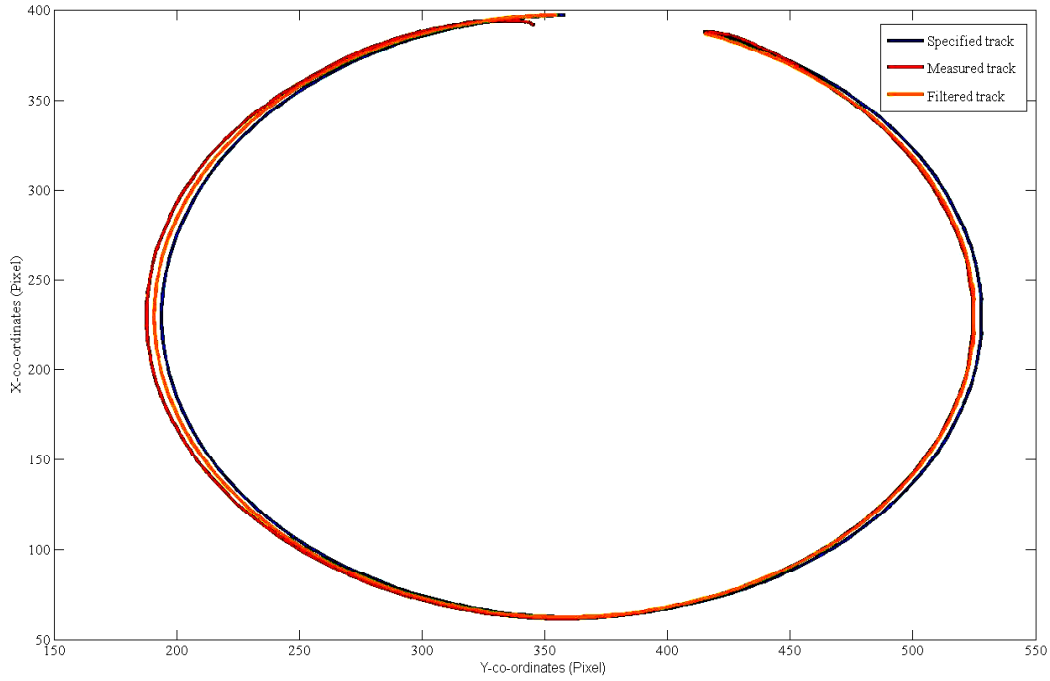
(b)


Specified Track
Measured Track
Kalman filtered Track
X-co-ordinates (Pixel)
Y-co-ordinates (Pixel)
450
400
350
300
250
200
100
150
200
250
300
350
400
450
500
550

(c)


Specified Track
Measured Track
Kalman filtered Track
X-co-ordinates (Pixel)
Y-co-ordinates (Pixel)
400
350
300
250
200
150
100
50
200
250
300
350
400
450
500
550
600

(d)



(e)

Figure 3.8 (a-e): Tracking results of the desired path using KLT and Kalman filtering algorithm

For five experiments, the average current position of the mobile robot is determined after implementation of the KLT and Kalman filter algorithms. The measured legend in Figure 3.9 is depicted by a surrounding bounding box in the shape of a 'yellow' rectangle. The size of the rectangle is determined according to the feature points detected near the object. The measured position (location) of the mobile robot in the filter is taken as the centroid of the depicted bounding box for the image. The Y-axis represents the percentage of error in tracking using KLT algorithm w.r.t. frame 'Red'. Similarly, percentage of error in tracking using KLT with Kalman filtering w.r.t. frame 'Blue' has been presented on Y-axis. For average position, average specific x-coordinate and average specific y-coordinate were calculated using the formula given in Equation 3.13.

$$\bar{x} = \sum_{i=1}^{5} \frac{x_i}{5} \; ; \; \bar{y} = \sum_{i=1}^{5} \frac{y_i}{5} \tag{3.13}$$

Likewise, $\Delta x = x_s - \bar{x}$ and $\Delta y = y_s - \bar{y}$ where x_s and y_s are the coordinates of specified path. Now, error % in x-direction is computed using following formula, Error % in $x\,(\varepsilon_x) = (\Delta x / r) \times 100\%$. Similarly, the error % in y-direction is computed as, Error % in $y\ (\varepsilon_y) = (\Delta y / r) \times 100\%$. Subsequently, the error % in positional error is computed using the following formula presented in Equation 3.14 below:

$$\text{Error \% in positional error } (\Delta P) = \sqrt{(\varepsilon_x)^2 + (\varepsilon_y)^2} \tag{3.14}$$

Five different sets of experiments of specified path tracking using Kalman filter and KLT tracking have been analyzed. The experiment was implemented using MATLAB 2015[b] on Microsoft Windows 7 and [Intel® CORE i5 CPU] with 16GB RAM. The resolution of each frame was 480×640. Based on the real-time results obtained, some interesting findings have been achieved in Figure 3.9 (a-e). Percentage error using KLT tracking and Kalman filtering with respect to frames provides more error in KLT tracking where it has been experiencing 70 to 80 % error due to unsteady velocity and change in the direction of the mobile robot.

The observed error using KLT or KLT with Kalman filtering can be attributed to

1. Motion of mobile robot (which is affected by operation hours), wheel diameter and its circularity and interaction between wheel and ground.
2. Steadiness of Vision sensor (depends on type of mounting); the overhead web camera was cable mounted and had some oscillation during experiment.
3. Associated error due to image processing (MATLAB software).
4. Associated due to image intensity noise and corresponding errors in the original feature detection.

Kalman filter provides better results with respect to KLT tracking. The proposed algorithm exhibited good performance in the tracking of the mobile robot on the specified path correctly.

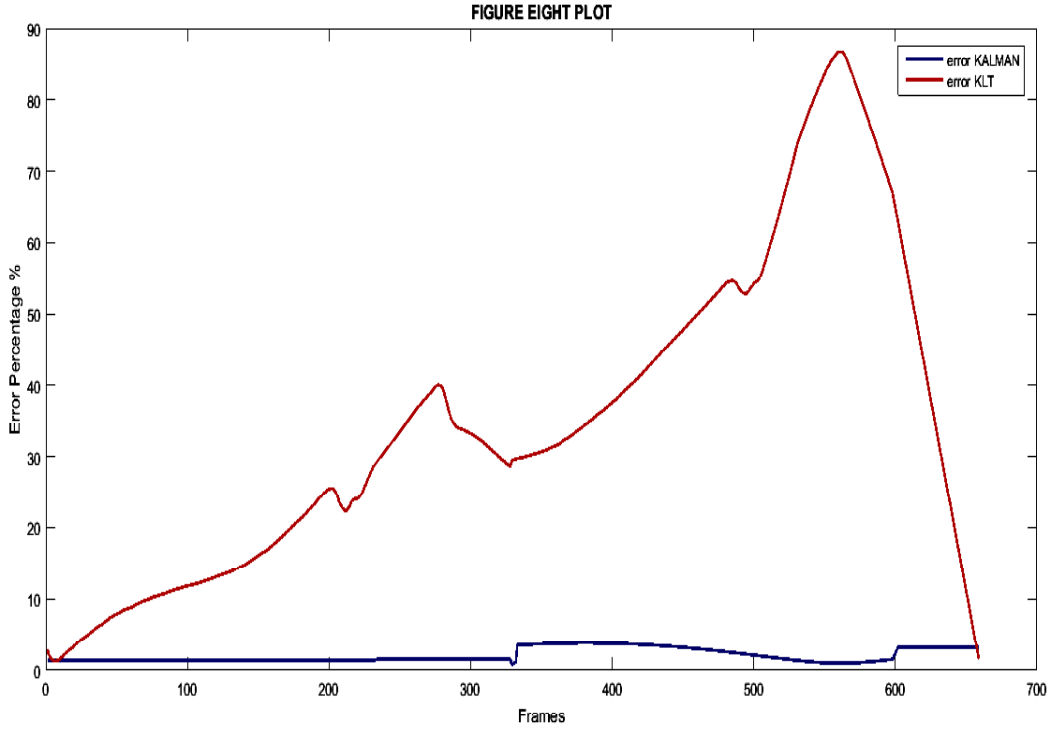


FIGURE EIGHT PLOT
error KALMAN
error KLT
Error Percentage %
Frames
0
10
20
30
40
50
60
70
80
90
0
100
200
300
400
500
600
700

(a)

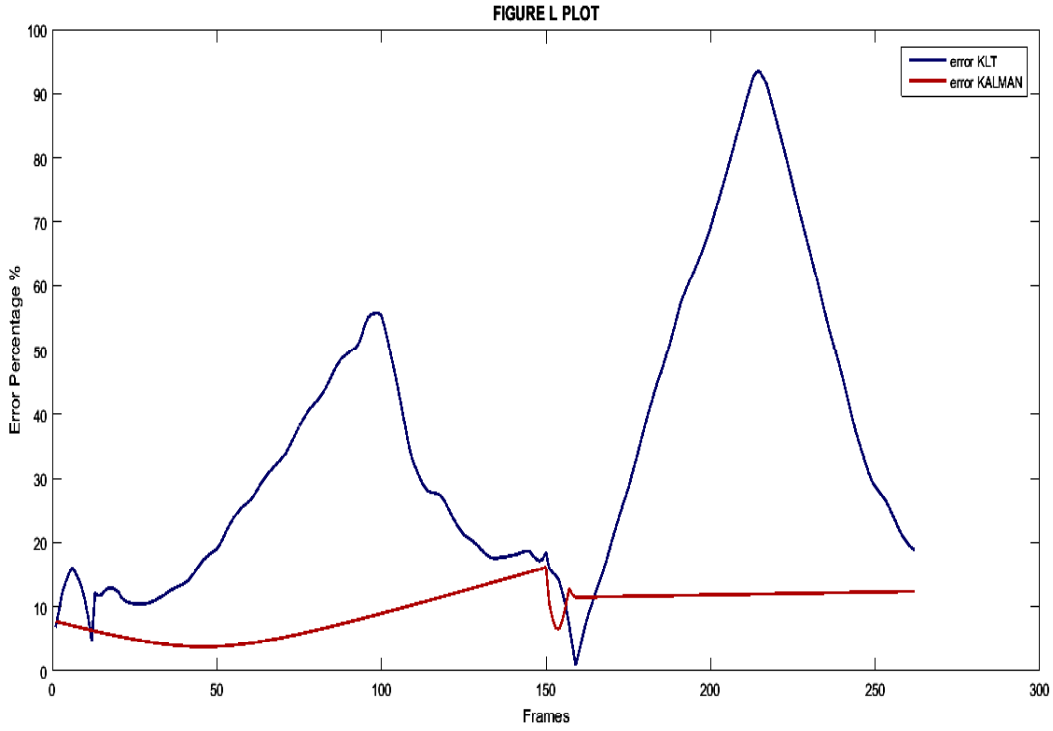
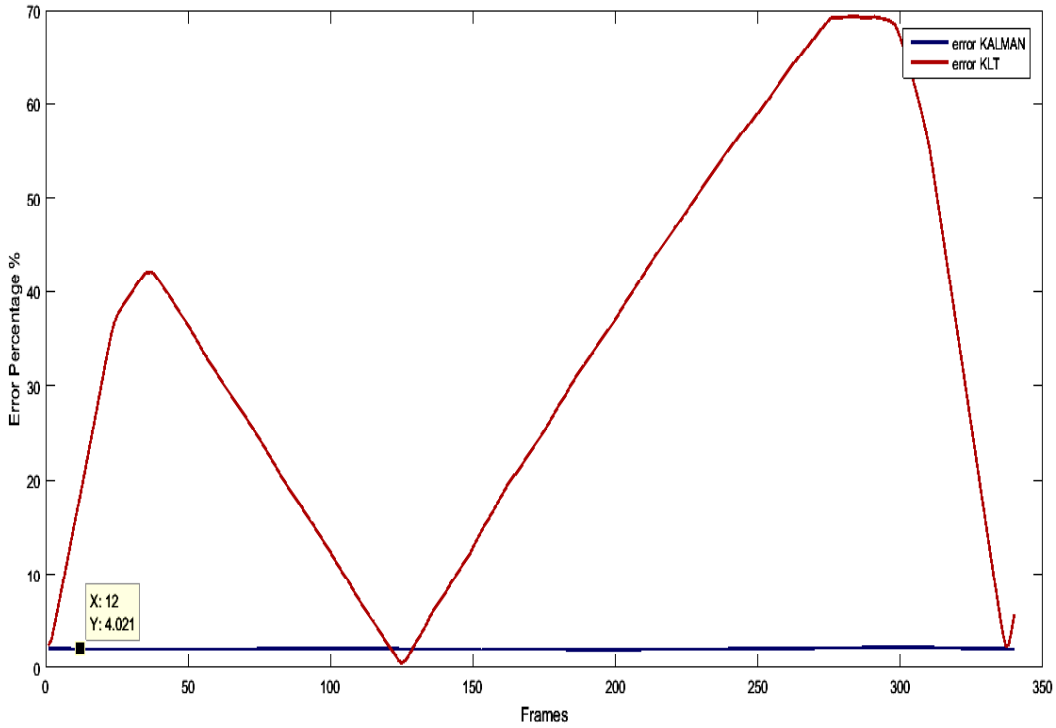


FIGURE L PLOT
error KLT
error KALMAN
Error Percentage %
Frames
0
10
20
30
40
50
60
70
80
90
100
0
50
100
150
200
250
300

(b)


error KALMAN
error KLT
Error Percentage %
Frames
X: 12
Y: 4.021
0
10
20
30
40
50
60
70
0
50
100
150
200
250
300
350

(c)

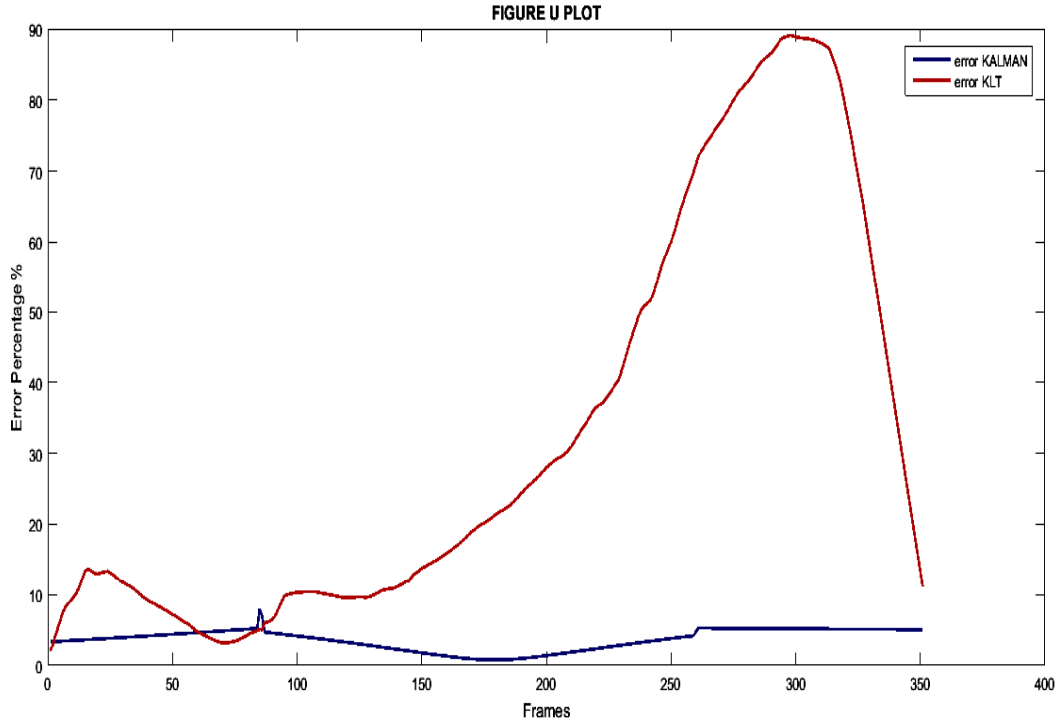
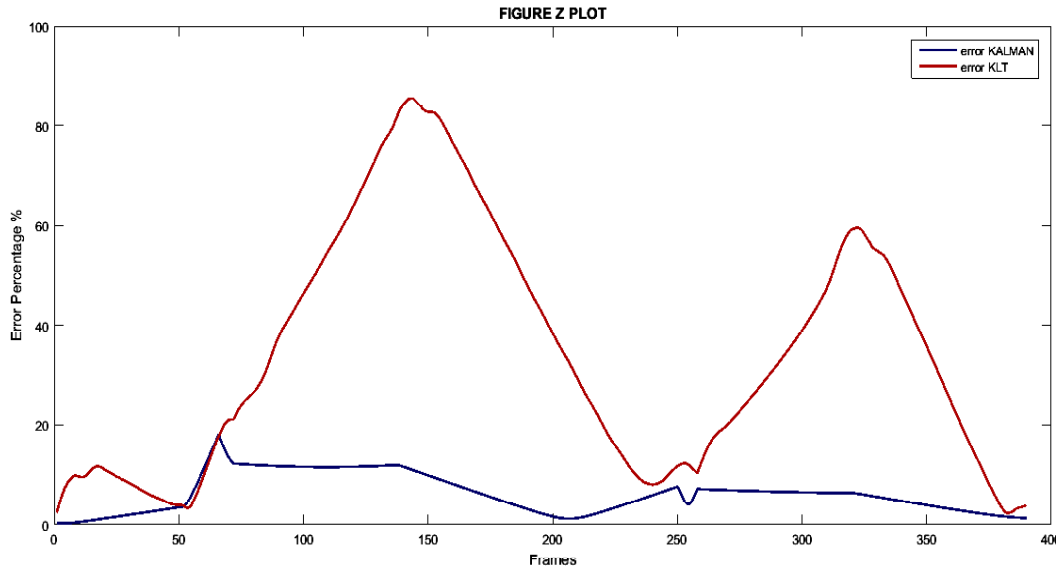


FIGURE U PLOT
error KALMAN
error KLT
Error Percentage %
Frames
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
400

(d)



(e)

Figure 3.9 (a-e): Percentage error using KLT and Kalman filtering

3.5 Conclusion

In this work, feature points based object detection and tracking method have been investigated for mobile robot perception using Kalman filter in a real-time environment. Tracking of mobile robot in a displacement sequence has been carried out using KLT algorithm whereas Viola-Jones is used for detecting mobile robot features. The mobile robot's image position with homography constraints was computed and a ROI window was set up with position in each frame to continue this tracking process. The whole tracking procedure has been completely automated and the system would recover on its own when the tracking information is lost. First, to establish Kalman filter motion model, KLT algorithm is used to choose centroid and tracking window as the features. The use of matching results, update the model of moving mobile robot, then updates the model as the next frame of the input parameters, so that continuous tracking of mobile robot is achieved. Here Kalman filter is used for prediction of matching and tracking in the case of mobile robot moving on a specified path. The motivation here is to create a visual surveillance system with real-time moving object detection, tracking, and activity analysis capabilities. In the future, this framework needs to detect and track objects in the complex real-time environment. The proposed algorithm has been validated for effectiveness and robustness using selected experiments with different specified path scenes. These experiments show accuracy ranging from 0.5% to 10%.