Video based Automatic Evaluation of White Line following Robot with Adaptive Cruise Control

MTP Stage-I Report

submitted in partial fulfillment of the requirements
for the award of the degree of

Master of Technology
in
Computer Science and Engineering

by

Khalid Waseem
(14305R008)

under the guidance of

Prof. Kavi Arya
&
Prof. Paritosh Pandya

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Oct, 2016

Contents

Abstract

e-Yantra project at IITB organizes various robotics competitions throughout the year. A participating student submits the video recording of a robot performing some predetermined task on arena. Evaluating these videos manually is a time consuming task. Following a trajectory correctly by a robot without any collision, is a core subtask of many of these themes.

In this report, we try to explore the possibility of automatically grading submitted videos. We focus on robot trajectory extraction from the submitted videos, and then comparing the observed trajectory with the ideal one. We explore ways to localize a robot on an arena with high precision. We show that if the height of the robot is fixed and known, we can use only a single calibrated camera to precisely locate its position (x,y) on the arena.

Chapter 1

Introduction

e-Yantra project at IITB sponsored by MHRD, aims to spread education and learning in the field of embedded systems and robotics programming. Various robotics competitions are organized throughout the year as part of e-Yantra project, in which thousands of students take part. A single competition has various themes which involves an arena, and a predetermined task, which the robot has to perform on the arena. An example of a theme would be *A white line following robot with adaptive cruise control*. A participating student when finished with robot programming, records the video of successful run and submits it for evaluation along with code and documentations. Manually evaluating all those video submissions for a theme is a tasking job. Also with increasing number of participations, it becomes extremely difficult to manually evaluate each and every video submission. We need a system which can separate videos which is performing the task correctly from those which are not, with least false positive and false negative cases. The system should also assign some grade based on how good the robot performs the task in the video.

Various informations can be extracted from those submitted videos, which can be used to evaluate them. For example: the trajectory followed by the robot in video while performing task. This trajectory information can be used to check whether the robot followed the correct trajectory, Whether it followed the shortest trajectory, How much time it spends in performing subtasks, etc. If we can get the robot's position on arena from the video at various time interval, we can generate this trajectory. This is also called as *Localization problem*. A successful solution to *Localization problem* has various other applications in entirely different scenarios. For example: it can be used to track vehicle location at traffic junctions [2], it can be used to track basketball trajectory in a recorded video of basketball game [6], etc.

Apart from trajectory information, we can also analyze the videos for collision with stationery objects, like walls present on arena. Blinking of led lights and buzzer sounds, which indicates subtask completion can also be detected from the videos. These information can be further used

for aiding video evaluation.

1.1 Problem Statement

This work aims at creating a system which can be quickly tuned to evaluate video submissions of a theme. Some of the components like trajectory analysis is common to all themes. We aim to create a subsystem which can localize the robot on arena with high precision. Then to use the generated trajectory in previous step to analyze path followed by the robot, detect collision and assign grades accordingly. We aim to achieve this using easily available video recording devices like webcam and mobile phone cameras.

1.2 Our Work

In this report we discuss two approaches we followed to extract robot trajectory from a recorded video. We also report the errors obtained.

1. **Homography and Patch Detection:** In this approach we converted each frame of submitted video from perspective to orthographic projection by calculating the homography between them. Then we detected the red patch on top of the robot in each frame to get the coordinates of robot. The method is discussed in detail in chapter 2. The coordinates thus obtained is stored as *trace files* along with timing information. These *trace files* acts as input to simulator.
2. **Camera Calibration:** In this approach we printed calibration rig on the arena. We took few pictures of arena from different orientations. Then used these pictures to calibrate the camera. Using calibration information obtained before and pixel coordinates of robot patch center we calculated its world coordinates (X_w, Y_w) for each frame of the video. The coordinates thus obtained is stored as *trace files* along with timing information.

We also developed a 2D simulator to show robot's trajectory extracted from the video. It uses the *trace files* generated before as input.

Chapter 2

Homography

Any two images of a planar surface is related by homography, if we assume pinhole camera model. Let a point P on the plane forms projection $p_1 = (x_1, y_1)$ on image 1. Similarly let point P forms projection $p_2 = (x_2, y_2)$ on image 2. Let the homogeneous coordinate representations of p_1 be (u_1, v_1, w_1) and p_2 be (u_2, v_2, w_2). Where,

$$x_1 = \frac{u_1}{w_1} \tag{2.1}$$

$$y_1 = \frac{v_1}{w_1} \tag{2.2}$$

$$x_2 = \frac{u_2}{w_2} \tag{2.3}$$

$$y_2 = \frac{v_2}{w_2} \tag{2.4}$$

Points p_1 and p_2 are related by homography. Therefore,

$$p_2 = Hp_1 \tag{2.5}$$

Where H is the homography matrix given by,

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \tag{2.6}$$

From equation 2.1 to 2.5 we have,

$$x_2 = \frac{H_{11}x_1 + H_{12}y_1 + H_{13}}{H_{31}x_1 + H_{32}y_1 + H_{33}} \tag{2.7}$$

$$y_2 = \frac{H_{21}x_1 + H_{22}y_1 + H_{23}}{H_{31}x_1 + H_{32}y_1 + H_{33}} \tag{2.8}$$

Rearranging equation 2.7 and 2.8 we get,

$$x_2x_1H_{31} + x_2y_1H_{32} + x_2H_{33} - x_1H_{11} - y_1H_{12} - H_{13} = 0 \tag{2.9}$$

$$y_2x_1H_{31} + y_2y_1H_{32} + y_2H_{33} - x_1H_{21} - y_1H_{22} - H_{23} = 0 \tag{2.10}$$

From one point P on the plane we obtained two equations above. In homography matrix H there are eight degrees of freedom $(H_{11}, H_{12}, ..., H_{32})$ Therefore we need at-least four point correspondence to solve the homography matrix H. Arranging equations 2.9 and 2.10 in matrix forms and using suitable number of point correspondence (at-least four), we get,

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -x_{1i} & -y_{1i} & -1 & 0 & 0 & 0 & x_{2i}x_{1i} & x_{2i}y_{1i} & x_{2i} \\ 0 & 0 & 0 & -x_{1i} & -y_{1i} & -1 & y_{2i}x_{1i} & y_{2i}y_{1i} & y_{2i} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \\ H_{33} \end{bmatrix} = 0 \tag{2.11}$$

In equation 2.11 above let the left-side matrix be A and the right-side matrix be h. Let there are n points of correspondence. Then we have,

$$A_{2n\times 9}h_{9\times 1} = 0 \tag{2.12}$$

Equation 2.12 can be solved by computing the SVD of A. Vector h is given by right singular vector corresponding to smallest singular value.

2.1 Experimental Results

We recorded multiple videos of a robot following black line on arena using a webcam (1280×720 pixels). Then we converted each frame of the video to orthographic projection. Afterwards we detected red patch on top of the robot. Using that we calculated the position of robot on the arena. Table 2.1 shows the absolute error and standard deviation obtained for each video separately.

Video No.	**Max Abs Error(cm)**	**Min Abs Error(cm)**	**Mean Abs Error(cm)**	**Std Dev(cm)**
Video 1	5.52	2.415	4.7759	0.6754
Video 2	2.585	0.25	1.407	0.3957
Video 3	3.355	0.02	1.2263	0.6205
Video 4	6.29	1.43	3.2465	1.4864

Table 2.1: *Absolute error for videos*

Figure 2.1 shows a single frame of a video before performing homography. Figure 2.2 shows one of the frames after performing homography. Figure 2.3 shows extracted coordinates from video 3. Most of the extracted coordinates are far away from the actual values ($X_w = 67.25$), depicted by red line.

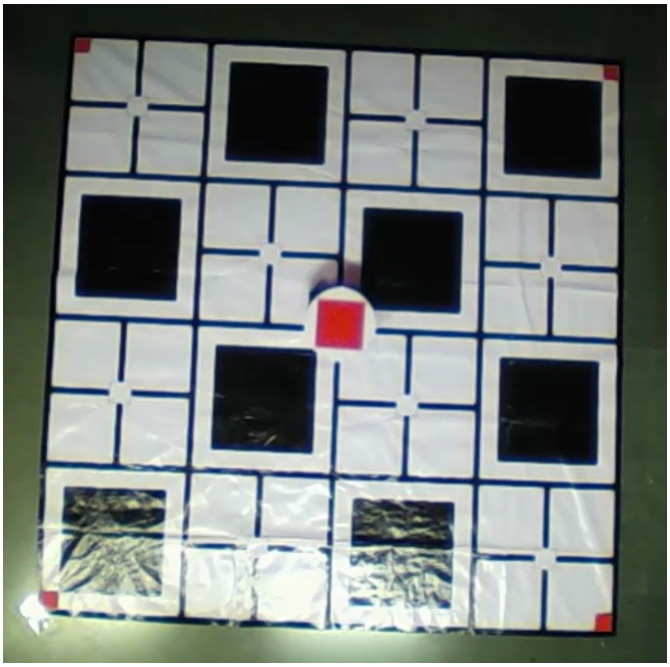

Figure 2.1: *A frame before homography*

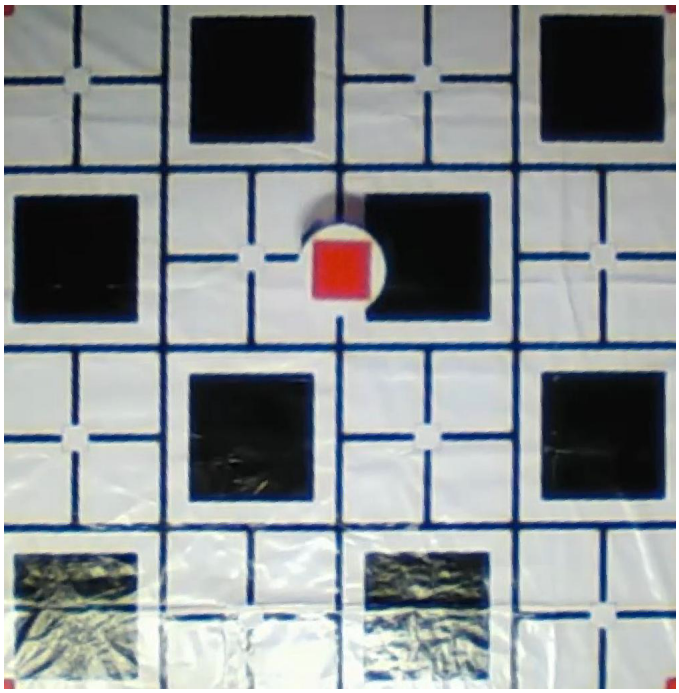

Figure 2.2: *A frame after homography*

2.2 Discussion

As is evident from the table 2.1 above, this technique produces a lot of error and hence cannot be used for precise localization of robot on arena.

Also for video 2 & 3 the mean absolute error is less as compared to video 1 & 4. The reason is that for video 2 & 3 the middle track on which the robot was moving was at the center of *field of view* of the camera. But for video 1 & 4 the track is at a distance from the center of *field of view* of the camera. As a result the red patches seems shifted from the track, as can be seen in figure 2.4.

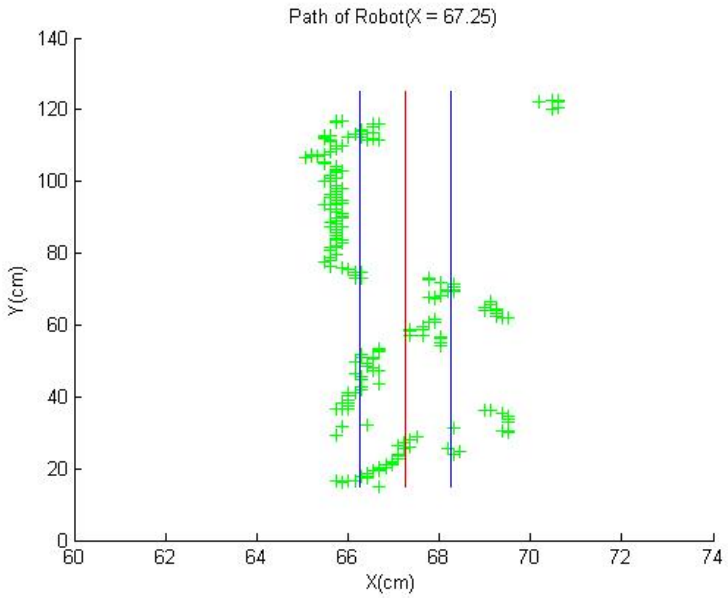

Figure 2.3: *Extracted coordinates vs actual values*

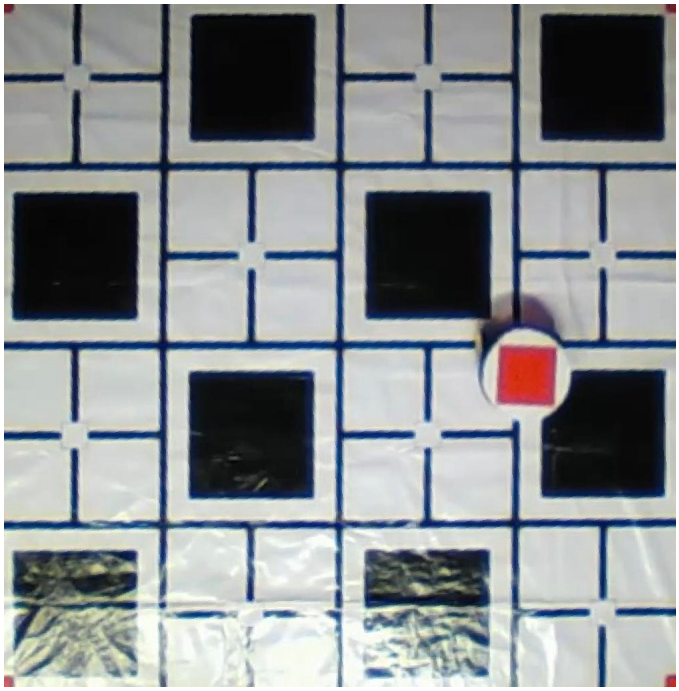

Figure 2.4: *Red patch shifted from track*

Chapter 3

Camera Calibration using Planar Rig

There has been much work in the field of camera calibration which can be broadly classified into two categories:

- **Photogrammetric calibration:** These techniques uses control points on a 3D rig to calibrate a camera. The geometry of the 3D rig as well as the location of those control points in 3D is well known in advance. Although calibration can be done very efficiently with high precision [1], these techniques are susceptible to measurement noise. So they require expensive calibration apparatus and complex setup to perform efficiently. These limitations render them improper for our use case.
- **Self-calibration:** These techniques uses correspondence between multiple images taken from same camera, along with features from scene to calculate both intrinsic and extrinsic parameters of a camera [3]. Although these techniques are very flexible but not yet fully developed to always give reliable results.

In this chapter we will discuss a novel approach for camera calibration introduced by Zhang [7], which uses a planar rig for calibration. The technique requires the camera to observe a planar rig from at-least three different orientations. The technique is very flexible and robust and easy to perform, which perfectly serves our intended use case. We will start with basics of camera geometry and then explain Zhang's [7] method for solving camera calibration problem.

3.1 Perspective Camera Model

The perspective or pinhole camera model defined by Trucco [5] is shown in figure 3.1. It's major components are:

- A *Camera Coordinate System* with origin at **O**, called as the *center* or *focus of projection*. **X**, **Y**, **Z** represents the x-axis, y-axis and z-axis of *camera coordinate system*.
- An *Image Plane* π, which is perpendicular to **Z**.
- An *Optical Axis* **OZ**, which passes through *center* **O** and is perpendicular to *image plane* π.
- A *Principal Center* or *Image Center* **o**, which is the point at which *optical axis* **OZ** intersects with *image plane* π.
- *Focal Length* **f**, which is the distance between *center* **O** and *image center* **o**.

For any given point $\mathbf{P} = [\mathbf{X_c}, \mathbf{Y_c}, \mathbf{Z_c}]^T$, its image will be formed at the point of intersection of **OP** and the *image plane* π. Let this point be $\mathbf{p} = [\mathbf{x_c}, \mathbf{y_c}, \mathbf{z_c}]^T$. Then the following equations holds.

$$x_c = f\frac{X_c}{Z_c} \tag{3.1}$$

$$y_c = f\frac{Y_c}{Z_c} \tag{3.2}$$

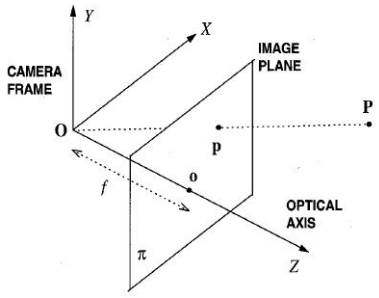

Figure 3.1: *The perspective camera model. Image courtesy Trucco [5]*

3.2 Camera Parameters

Equation 3.1 & 3.2 are defined in *camera reference frame*. It is extremely difficult and impractical to actually locate the *camera reference frame*. Hence Trucco [5] defines a new coordinate system called *world coordinate frame*, whose origin and orientation is known before hand. To use equations 3.1 & 3.2, we still need to find the transformation which will take us from the *world coordinate frame* to the *camera coordinate frame*. This transformation is called *Extrinsic Parameters* of camera. Similarly, *Intrinsic Parameters* of camera are needed to convert *pixel coordinates* to *image coordinates* (x_c, y_c) in *camera reference frame*.

3.2.1 Extrinsic Parameters

The *extrinsic parameters* of a camera defines the rotation **R**, and translation **T**, needed to align a known *world coordinate frame* with *camera reference frame* as shown in figure 3.2.

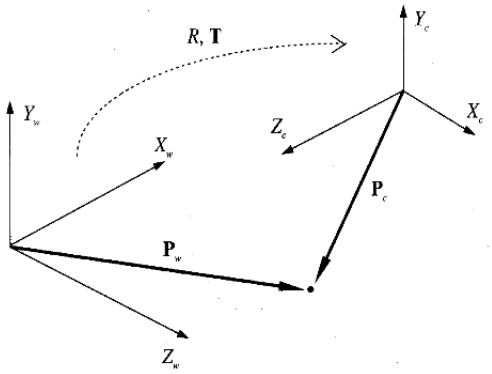

Figure 3.2: *The relation between camera and world coordinate frames. Image courtesy Trucco [5]*

Let $\mathbf{P_c}$ and $\mathbf{P_w}$ be coordinates of a point with respect to *camera reference frame* and *world coordinate frame*. Then the following equation hold.

$$P_{\mathrm{c}} = R(P_{\mathrm{w}} - T) \tag{3.3}$$

3.2.2 Intrinsic Parameters

The coordinates of a point obtained from image is in pixels. *Intrinsic Parameters* are needed to convert image coordinates in pixels from *image reference frame* to corresponding points in *camera reference frame*. Let $(\mathbf{x_{im}},\mathbf{y_{im}})$ in pixels be coordinates of a point in *image reference frame* and $(\mathbf{x_c},\mathbf{y_c})$ be the corresponding coordinates of the same point in *camera reference frame*. Then the following equations hold.

$$x_{\mathrm{c}} = -(x_{\mathrm{im}} - o_{\mathrm{x}})s_{\mathrm{x}} \tag{3.4}$$

$$y_{\mathrm{c}} = -(y_{\mathrm{im}} - o_{\mathrm{y}})s_{\mathrm{y}} \tag{3.5}$$

Where $(\mathbf{o_x},\mathbf{o_y})$ is coordinates of *image center* in pixels and $(\mathbf{s_x},\mathbf{s_y})$ is size of pixels in millimeters. So the equation relating pixel coordinates of a point with its corresponding world coordinate can be written as:

$$s \begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} = M_{int} M_{ext} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.6}$$

Where s is a constant.

3.3 Camera Calibration Problem

Camera Calibration Problem is defined as, Given a set of points in *image coordinate system* and their corresponding values in *world coordinate system*, we have to estimate the camera's *Intrinsic* and *Extrinsic parameters*.

As discussed in the beginning of the chapter, numerous methods exist to solve camera calibration problem. But we will discuss Zhang's [7] method, as it is most suitable for our intended use. Zhang [7] uses a planar rig observed from at-least three different orientation for solving camera calibration problem.

3.3.1 Notation Used

- A point in *image coordinate system*, in pixels is represented by $\mathbf{p_{im}} = [x_{\mathrm{im}}, y_{\mathrm{im}}]^{\mathrm{T}}$.
- A point in *world coordinate system* is represented by $\mathbf{P_w} = [X_{\mathrm{w}}, Y_{\mathrm{w}}, Z_{\mathrm{w}}]^{\mathrm{T}}$.
- Homogeneous coordinate system representations are $\overline{\mathbf{p_{im}}} = [x_{\mathrm{im}}, y_{\mathrm{im}}, 1]^{\mathrm{T}}$ and $\overline{\mathbf{P_w}} = [X_{\mathrm{w}}, Y_{\mathrm{w}}, Z_{\mathrm{w}}, 1]^{\mathrm{T}}$.
- Rotation matrix is represented by $\mathbf{R}$. $\mathbf{r_i}$ represent the i^{th} column of $\mathbf{R}$.
- Translation vector is represented by $\mathbf{T}$.
- Camera's *intrinsic matrix* is represented by $\mathbf{M_{int}}$. Where,

$$M_{\mathrm{int}} = \begin{bmatrix} \alpha & \gamma & o_{\mathrm{x}} \\ 0 & \beta & o_{\mathrm{y}} \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

- o_x, o_y are the coordinates of *image center*.
- α and β are the scale factors in *image coordinate system* axis.
- γ represents the skewness of the two axes of *image coordinate system*.

Let us assume that the planar rig lies at $(Z = 0)$ of *world coordinate system*. Using *homogeneous coordinate system* equation 3.6 can be written as:

$$s \begin{bmatrix} x_{\text{im}} \\ y_{\text{im}} \\ 1 \end{bmatrix} = M_{\text{int}} \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X_{\text{w}} \\ Y_{\text{w}} \\ 0 \\ 1 \end{bmatrix} \tag{3.8}$$

$$= M_{\text{int}} \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X_{\text{w}} \\ Y_{\text{w}} \\ 1 \end{bmatrix} \tag{3.9}$$

Here s is an arbitrary scale factor. Lets us still use $\mathbf{P_w}$ to denote a point on planar rig, but $\mathbf{P_w} = [X_{\text{w}}, Y_{\text{w}}]^{\text{T}}$ cause Z_{w} will always be zero. Clearly $\overline{\mathbf{p_{im}}}$ and $\overline{\mathbf{P_w}}$ are related by a homography $\mathbf{H}$.

$$s\overline{p_{\text{im}}} = H\overline{P_{\text{w}}} \tag{3.10}$$

$$H = M_{\text{int}} \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \tag{3.11}$$

Given an image of planar rig and its actual size, the homography matrix $\mathbf{H}$ can be easily calculated. There are various methods to calculate the homography matrix $\mathbf{H}$. We have discussed one method in chapter 2. Let $\mathbf{H} = [h_1\ h_2\ h_3]$. Then from equation 3.11, we have.

$$\begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = \lambda M_{\text{int}} \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \tag{3.12}$$

Where λ is a scalar. Since r_1 and r_2 are orthonormal to each other. We get below constraints on *intrinsic parameters* of camera:

$$h_1^T\ M_{int}^{-T}\ M_{int}^{-1}\ h_2 = 0 \tag{3.13}$$

$$h_1^T\ M_{int}^{-T}\ M_{int}^{-1}\ h_1 = h_2^T\ M_{int}^{-T}\ M_{int}^{-1}\ h_2 \tag{3.14}$$

3.3.2 Solving Camera Calibration

In this section we will first discuss the closed form solution suggested by Zahng [7], followed by a nonlinear optimization technique for camera calibration problem assuming no lens distortion.

Later we will introduce lens distortion and then derive both closed form and nonlinear solutions for camera calibration.

Let $B = M_{int}^{-T}\ M_{int}^{-1}$. Then B will be a symmetric matrix. let it be represented by a 6D vector b, such that.

$$b = \begin{bmatrix} B_{11} & B_{12} & B_{22} & B_{13} & B_{23} & B_{33} \end{bmatrix}^T \tag{3.15}$$

Let the i^{th} column vector of homography matrix H, be represented as $h_i = \begin{bmatrix} h_{i1} & h_{i2} & h_{i3} \end{bmatrix}^T$. Then putting B in equation 3.13 and 3.14 we get.

$$h_i^T B h_j = v_{ij}^T b \tag{3.16}$$

Where v_{ij} is defined as

$$v_{ij} = \begin{bmatrix} h_{i1}h_{j1} & h_{i1}h_{j2} + h_{i2}h_{j1} & h_{i2}h_{j2} & h_{i3}h_{j1} + h_{i1}h_{j3} & h_{i3}h_{j2} + h_{i2}h_{j3} & h_{i3}h_{j3} \end{bmatrix}^T \tag{3.17}$$

Using 3.16 we can write the two constraints on intrinsic parameters 3.13 and 3.14 in matrix form as:

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0 \tag{3.18}$$

Equation 3.18 is obtained from a single image of planar rig. If we have n images of the planar rig from n different orientations, by stacking n such equations as 3.18 we get

$$V_{2n\times 6}\ b_{6\times 1} = 0 \tag{3.19}$$

Equation 3.19 can be easily solved by finding the eigen vector of $V^T V$ corresponding to smallest eigen value. Using solution for b we can find all *intrinsic parameters* M_{int}, of camera.

$$o_y = (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2) \tag{3.20}$$
$$\lambda = B_{33} - [B_{13}^2 + o_y(B_{12}B_{13} - B_{11}B_{23})]/B_{11} \tag{3.21}$$
$$\alpha = \sqrt{\lambda/B_{11}} \tag{3.22}$$
$$\beta = \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)} \tag{3.23}$$
$$\gamma = -B_{12}\alpha^2\beta/\lambda \tag{3.24}$$
$$o_x = \gamma o_y/\beta - B_{13}\alpha^2/\lambda \tag{3.25}$$

Using value of M_{int} calculated above, we can calculate the *extrinsic parameters* as:

$$r_1 = \lambda M_{int}^{-1} h_1 \tag{3.26}$$
$$r_2 = \lambda M_{int}^{-1} h_2 \tag{3.27}$$
$$r_3 = r_1 \times r_2 \tag{3.28}$$
$$t = \lambda M_{int}^{-1} h_3 \tag{3.29}$$

$R = \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}$ calculated above will in general not satisfy the conditions of rotation matrix. But we can approximate the closest rotation matrix R from a given 3×3 matrix Q. The solution to this problem is well known ($R = UV^T$, where $Q = USV^T$). The proof can be found here [7].

The closed form solution described above performs poorly when measurement noise is present. The solution can be greatly improved by non linear optimization of below function.

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \|m_{ij} - \hat{m}(M_{int}, R_i, t_i, P_j)\|^2 \tag{3.30}$$

Where n is number of images, m is number of control points on each image, m_{ij} is measured pixel coordinate of j^{th} point in i^{th} image. $\hat{m}(M_{int}, R_i, t_i, P_j)$ is projection of point P_j on image i. Equation 3.30 is a nonlinear minimization problem which can be solved using *Levenberg-Marquardt Algorithm* [4].

3.3.3 Radial Distortion

Due to manufacturing defect lens may exhibit radial distortion. But it can be modeled as below.

$$\hat{x}_c = x_c + x_c[k_1(x_c^2 + y_c^2) + k_2(x_c^2 + y_c^2)^2] \tag{3.31}$$
$$\hat{y}_c = y_c + y_c[k_1(x_c^2 + y_c^2) + k_2(x_c^2 + y_c^2)^2] \tag{3.32}$$

Since the center of radial distortion is same as image center, we have

$$\hat{x}_{im} = o_x + \alpha\hat{x}_c + \gamma\hat{y}_c \tag{3.33}$$
$$\hat{y}_{im} = o_y + \beta\hat{y}_c \tag{3.34}$$

From above two equations we finally get

$$\hat{x}_{im} = x_{im} + (x_{im} - o_x)[k_1(x_c^2 + y_c^2) + k_2(x_c^2 + y_c^2)^2] \tag{3.35}$$
$$\hat{y}_{im} = y_{im} + (y_{im} - o_y)[k_1(x_c^2 + y_c^2) + k_2(x_c^2 + y_c^2)^2] \tag{3.36}$$

Where ($\hat{x}_{im}$,$\hat{y}_{im}$) is observed or distorted pixel coordinate of a point. (x_{im},y_{im}) is the corresponding ideal pixel coordinates. (x_c,y_c) is the ideal coordinate of point in *camera coordinate system*. ($\hat{x}_c$,$\hat{y}_c$) is the corresponding distorted coordinate. k_1 and k_2 are coefficients of radial distortion.

Equation 3.30 can be extended to include radial distortion as follows.

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\|m_{ij} - \hat{m}(M_{int}, k_1, k_2, R_i, t_i, P_j)\|^2 \tag{3.37}$$

Where $\hat{m}(M_{int}, k_1, k_2, R_i, t_i, P_j)$ is projection of point P_j on image i followed by distortion using equations 3.35 and 3.36

3.4 Obtaining World Coordinates from Pixel Coordinates using Single Calibrated Camera

Given the pixel coordinate of a point we aim to find its 3D coordinate in *world coordinate system*. Generally this requires two calibrated cameras. But if we know the Z_w coordinate of the point whose world coordinate we wish to calculate then it can be done using a single calibrate camera only. From equation 3.1, 3.2, 3.3, 3.4 and 3.5 we get

$$x_{im} - o_x = f_x \frac{r_{11}X_w + r_{12}Y_w + r_{13}Z_w + T_x}{r_{31}X_w + r_{32}Y_w + r_{33}Z_w + T_z} \tag{3.38}$$

$$y_{im} - o_y = f_y \frac{r_{21}X_w + r_{22}Y_w + r_{23}Z_w + T_y}{r_{31}X_w + r_{32}Y_w + r_{33}Z_w + T_z} \tag{3.39}$$

Where $f_x = f/s_x$, $f_y = f/s_y$, $r_{ij} = R[i][j]$, T_x, T_y, and T_z is x, y, z component of translation vector. Rearranging and writing in matrix form we get

$$\begin{bmatrix} (x_{im} - o_x)r_{31} - f_x r_{11} & (x_{im} - o_x)r_{32} - f_x r_{12} \\ (y_{im} - o_y)r_{31} - f_y r_{21} & (y_{im} - o_y)r_{32} - f_y r_{22} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \end{bmatrix} = \begin{bmatrix} Z_w[f_x r_{13} - (x_{im} - o_x)r_{33}] + f_x T_x - (x_{im} - o_x)T_z \\ Z_w[f_y r_{23} - (y_{im} - o_y)r_{33}] + f_y T_y - (y_{im} - o_y)T_z \end{bmatrix} \tag{3.40}$$

Equation 3.40 above can be solved to obtain X_w and Y_w.

3.5 Experimental Results

We performed two set of experiments. Both are described below:

3.5.1 Experiment on Static Images

In first experiment we took 25 static images of arena from different orientation, with a cube of known dimension placed on it using iPhone camera (3264×2448 pixels). We calibrated the camera using calibration technique discussed in section 3.3 using those images. Then we calculated the world coordinate (X_w,Y_w) for corner points on cube for each image, given its pixel coordinate (x_{im},y_{im}) and height Z_w as input. Table 3.1 shows the absolute error and standard deviation obtained.

Maximum Absolute Error(cm)	1.7519
Minimum Absolute Error(cm)	5.05E-02
Mean Absolute Error(cm)	0.7769
Std. Deviation(cm)	0.2731

Table 3.1: *Absolute error for 25 images*

In figure 3.3 we show the corners extracted from one of the image which was used during calibration. In figure 3.4 we show the four points on cube whose world coordinate (X_w,Y_w) is calculated from pixel coordinate (x_{im},y_{im}). The origin of *world coordinate system* is also shown. In

figure 3.5 we show the predicted corner coordinates for all 25 images. The blue asterisks are the actual corners. The green crosses are predicted corners.

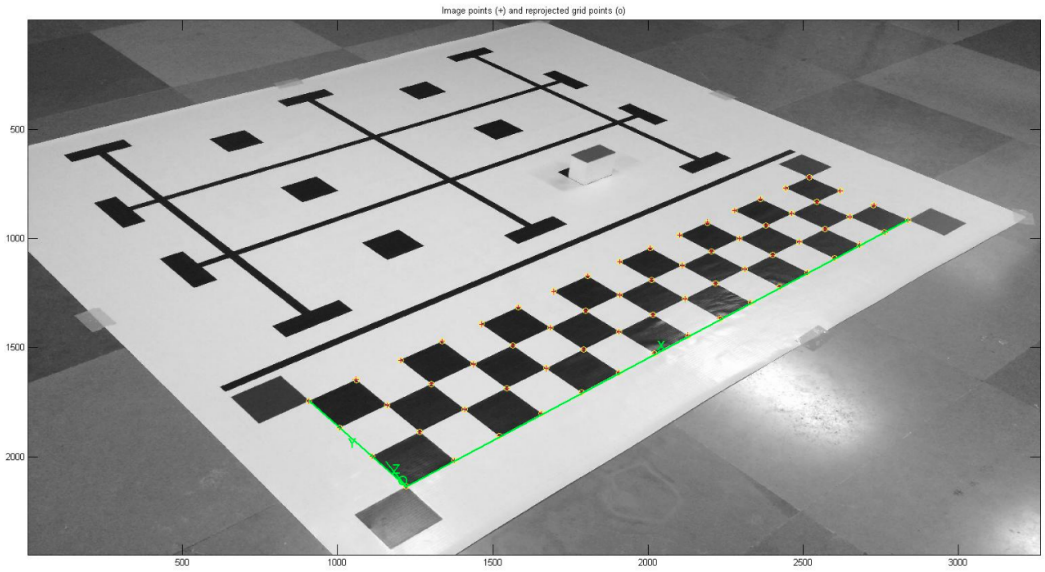

Figure 3.3: *Extracted corners during calibration process.*

3.5.2 Experiment on Video

In second experiment we took 7 videos of arena from different orientation, with a robot of known height moving on it using iPhone video camera (1920×1080 pixels). We calibrated the camera using calibration technique discussed in section 3.3. Then for each frame of each video we detected the center of red patch attached on top of robot. We calculated the world coordinate (X_w,Y_w) of the center of red patch for each frame of each video. Table 3.2 shows the absolute error and standard deviation obtained for each video separately.

Video No.	Max Abs Error(cm)	Min Abs Error(cm)	Mean Abs Error(cm)	Std Dev(cm)
Video 1	0.8672	0.0951	0.388	0.1447
Video 2	1.2079	0.4003	0.8704	0.1978
Video 3	0.6227	6.53E-04	0.1862	0.1997
Video 4	0.6458	0.2699	0.4382	0.0686
Video 5	0.8722	0.0024	0.247	0.1766
Video 6	0.5051	6.52E-04	0.223	0.1058
Video 7	0.9403	0.6835	0.8267	0.052

Table 3.2: *Absolute error for 7 videos*

In figure 3.6 we show the corners extracted from one of the frame from one of the video which

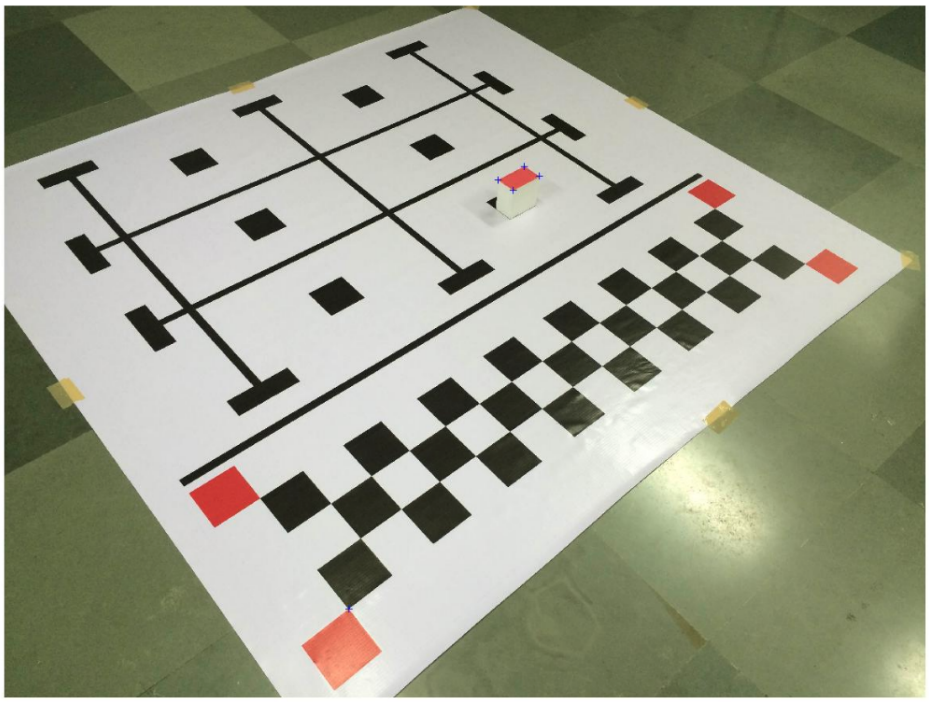

Figure 3.4: *Four points on the cube*

was used during calibration. In figure 3.7 we show the center of the extracted patch whose world coordinate (X_w,Y_w) is calculated from pixel coordinate (x_{im},y_{im}). The origin of *world coordinate system* is also shown. In figure 3.8 we show the predicted patch center coordinates for all frames of a video. The red line corresponds to ideal $X_w = 135\ cm$ value.

3.6 Discussion

As evident from table 3.2 and graph 3.8, the average absolute error is within 1 cm also the maximum absolute error is close to 1 cm. So, we can infer that, this is a good technique for robot localization on arena given the size of arena was $200 \times 200\ cm$.

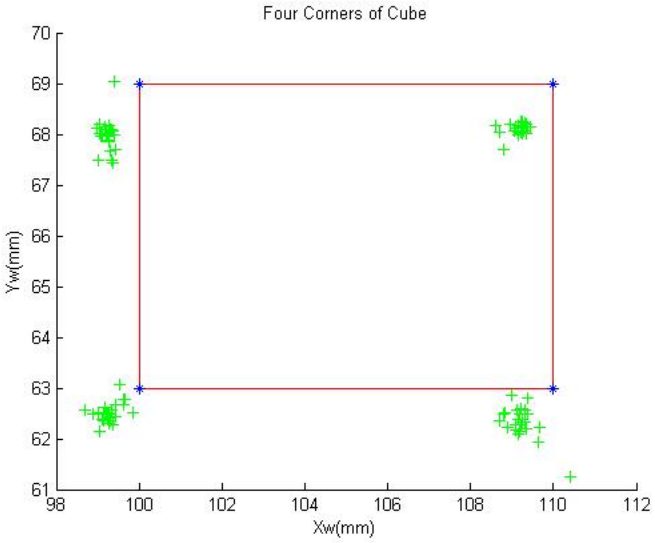

Figure 3.5: *Predicted corner coordinates for all 25 images*

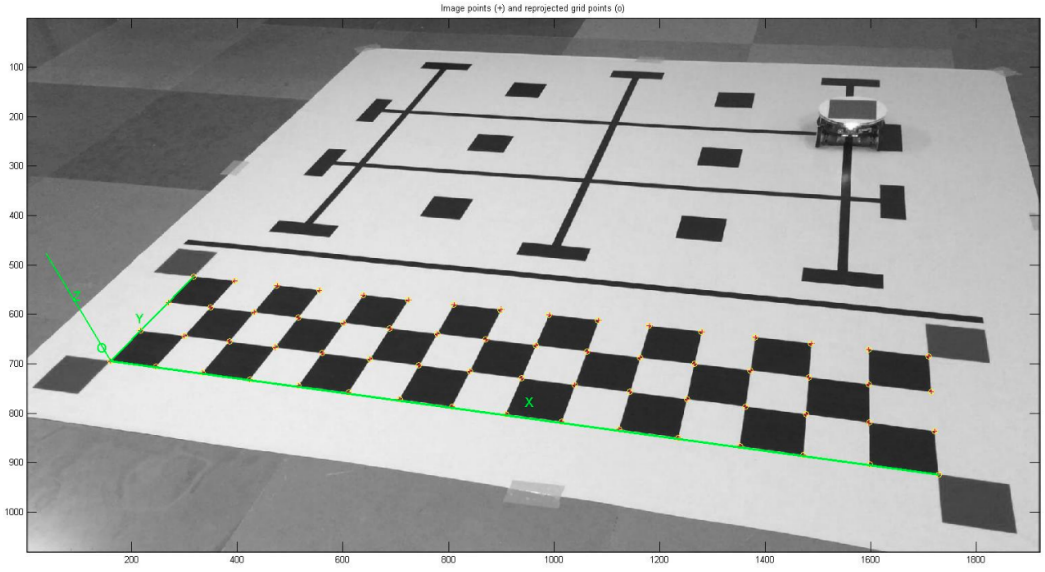

Figure 3.6: *Extracted corners during calibration*

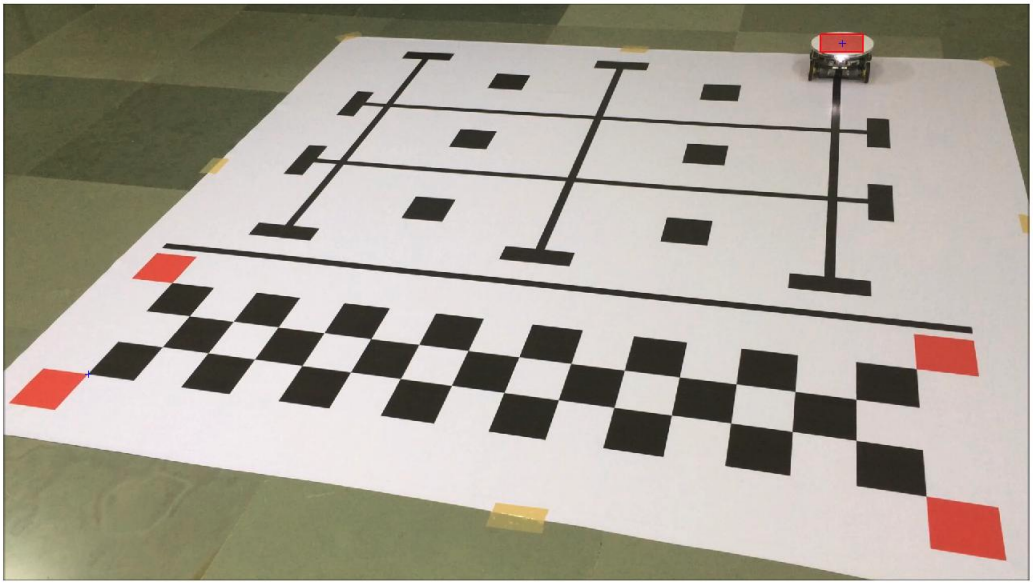

Figure 3.7: *Detected center of patch*

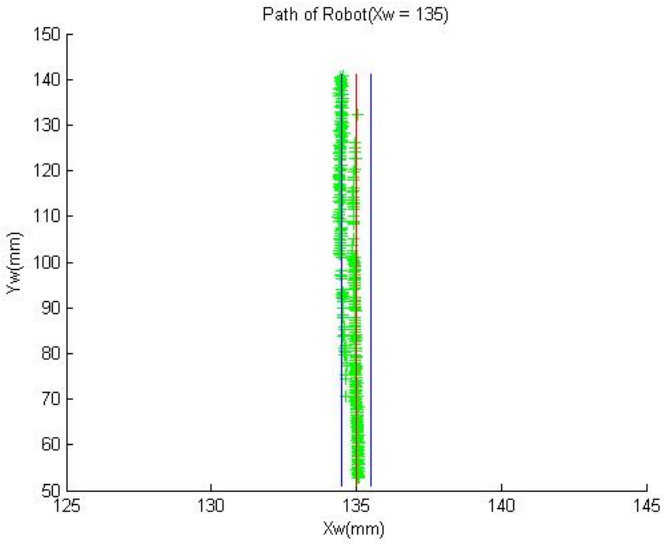

Figure 3.8: *Predicted patch center for all 7 videos*

Chapter 4

Robot Trajectory Simulator

To visualize robot trajectory extracted in previous chapter, we developed a 2D simulator. It takes robot position on arena (X_w, Y_w) along with time as input and then animates the robot's motion from corresponding video. Figure 4.1 below, shows screen-shot of the animation we performed for video 1.

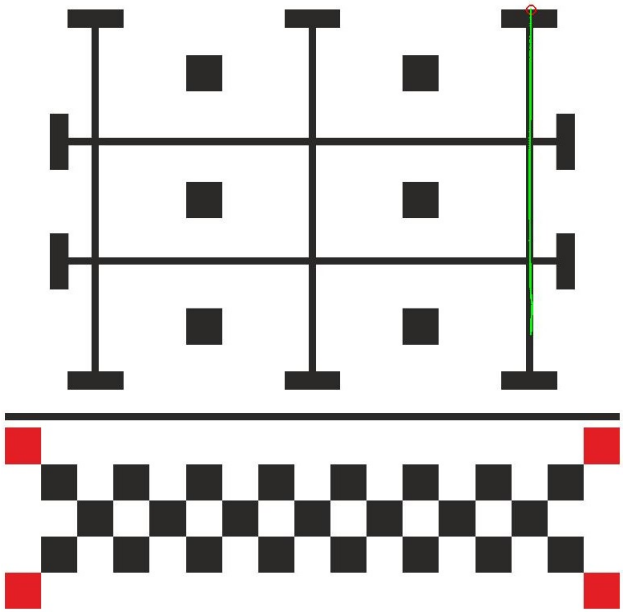

Figure 4.1: *Robot trajectory shown in simulator*

The simulator also has the facility to generate user defined trajectories. It stores random points along the mouse movement with timing information. This feature will be helpful in rapid testing of *curve similarity* code. Figure 4.2 below, shows a generated trajectory. The blue points are the

sampled positions.

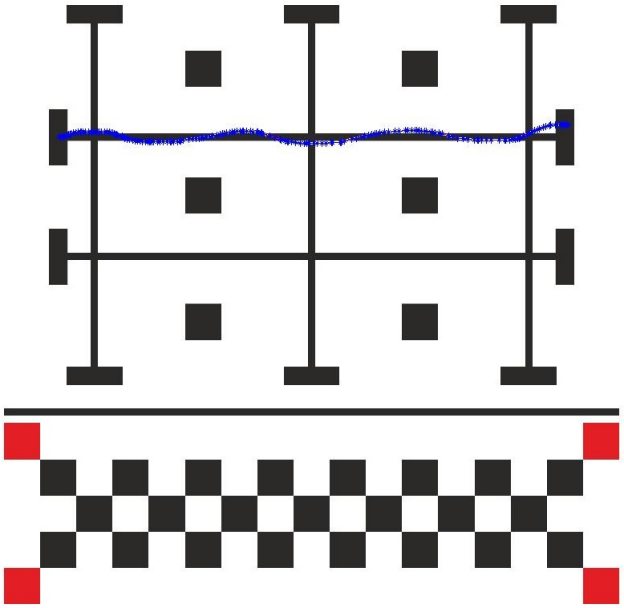

Figure 4.2: *Generated trajectory*

Chapter 5

Conclusion and Future Work

In this report, we described an approach for robot localization on an arena using a single camera. The camera calibration technique we discussed is easy to perform and can be replicated even in non-lab environment. We discussed method to obtain robot's position (X_w, Y_w) on arena from frames of the recorded video using only a single calibrated camera. The position (X_w, Y_w) so obtain were well within targeted error limit. We also developed a 2D simulator which replicates robot's motion according to the trajectory extracted from the recorded video.

For future work, we would like to perform *curve similarity* analysis on extracted robot trajectory and ideal robot trajectory. We will work on collision detection and develop a system to grade themes like *White line following robot with adaptive cruise control*, *Balancing bot* etc. which can be evaluated just by analyzing the trajectory of robot. In future we would like to build an end to end system which can take recorded videos as input and provide grade according to how well the task was performed by the robot.

Acknowledgment

This report is an outcome of my work under the guidance of *Prof. Kavi Arya* and *Prof. Paritosh Pandya*. I would like to thank them for their valuable guidance, motivation, and feedback. For taking out time from their busy schedules to provide helpful insight, whenever I was stuck. It was immensely helpful in gaining the understanding required for writing this report.

Thanks to my colleagues for their valuable insights which helped me throughout the project. Special thanks goes to *Sandip Ghoshal*, *Pooja Choudhary*, *Aurobindo Mondal* for motivating me throughout my work.

Bibliography

[1] Olivier Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT press, 1993.

[2] Karim Ismail, Tarek Sayed, and Nicolas Saunier. A methodology for precise camera calibration for data collection applications in urban traffic scenes. *Canadian Journal of Civil Engineering*, 40(1):57–67, 2013.

[3] Stephen J Maybank and Olivier D Faugeras. A theory of self-calibration of a moving camera. *International Journal of Computer Vision*, 8(2):123–151, 1992.

[4] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[5] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[6] Pei-Chih Wen, Wei-Chih Cheng, Yu-Shuen Wang, Hung-Kuo Chu, Nick C Tang, and Hong-Yuan Mark Liao. Court reconstruction for camera calibration in broadcast basketball videos. *IEEE Transactions on Visualization and Computer Graphics*, 22(5):1517–1526, 2016.

[7] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.