R N O C

A thesis submitted in partial fulfillment of
the requirements for the degree of

Bachelor of Technology
in
Computer Science and Engineering

Rahul Bhotika

Anurag Mittal

Supervisor : **Dr Subhashis Banerjee**

Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

May 1997

CERTIFICATE

This is to certify that the thesis entitled **Robot Navigation with Online Control**, submitted by Rahul Bhotika and Anurag Mittal in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering, at the Indian Institute of Technology, Delhi, is a bonafide record of the project work done by them.

This project was carried out under my supervision and the matter and results embodied in this thesis are original and have not been submitted elsewhere, wholly or in part, for the award of any degree or diploma, or for any other purpose.

Dr Subhashis Banerjee
Department of Computer Science and Engineering
Indian Institute of Technology
Delhi

ACKNOWLEDGMENTS

We are deeply indebted to our supervisor **Dr. Subhashis Banerjee** for the continuous encouragement, guidance and support he has provided to us throughout the course of this project. He has been a great source of inspiration and his advice has been invaluable for us.

We are also deeply indebted to **Mr. Kaushik** without whose help our laboratory work would have been impossible. Thanks are also due to Mr. Sumantra Dutta Roy for his enthusiastic support, to Amit Aggarwal and Lalit Kumar for providing valuable tips, and to Dinesh Prasad and Mohit Talwar for carrying Rahul along for so many days.

Rahul Bhotika **Anurag Mittal**

Abstract

This work demonstrates a method of using information derived from a partially calibrated vision system to guide a mobile vehicle through free space in an environment with static obstacles.

The system used consists of a mobile robot with a pair of cameras mounted on it to grab stereo images of the scene. Correspondences are determined between corners in the two images. Information derived from a one-time offline calibration is used in back projection to recover the 3D structure. This is projected on to ground parallel points and the robot moves through the available space.

The ultimate goal of a navigation project is to provide an automated control for the robot. In this work, we have implemented a smaller subset of the total navigation problem. Our objective is to demonstrate the use of a simpler method, instead of existing elaborate and expensive techniques to build a robust navigation system. Our work provides a user controlled system for navigating through the free space. We have also attempted to implement a small automated version, which navigates through the space between two objects. This work can be used as the basis for a more complete navigation system.

Contents

List of Figures

Part I

Theory

Chapter 1

Introduction

Robot navigation is a well-known problem. Efforts have been on for several years for obtaining information about the environment in which the robot is required to operate to enable it to move through it. Robots, provided with mounted cameras and movement capabilities, can be used for various purposes. Our aim is to provide a robust vision-based navigation system which can actually be used in practice.

1.1 Motivation

One of the methodologies used as the basis for robot navigation is through the process of estimation of the $3D$ of the environment. The dynamic recovery of structure is now sufficiently well-explored for such methods to have been applied to the guidance of autonomous mobile vehicle. The $3D$ structure is computed from two or more views, acquired simultaneously, and this is used to determine $3D$ information about the environment. This requires calibration of the parameters of the cameras used.

Other methodologies have obtained the affine structure of the scene using uncalibrated cameras which is then used to find the projective structure using further manipulations. These methods are complicated and involve elaborate computation.

Our motivation is towards providing a system where a one-time off-line calibration is sufficient. This information from calibration can be used to calculate the $3D$ structure to a high degree of accuracy.

The focus in this work is on getting sufficient information to deduce a structure from which locations of the obstacles can be identified as clusters of points, and the robot can be navigated around them, or through the available free space.

1.2 Basic principles involved

The following steps are salient to our system:

- For structure determination, the system requires a robot head with cameras mounted on it which grab stereo views of the scene and derive the $3D$ structure of the scene from it.
- The $3D$ structure is determined in terms of the corners in the scene objects, by way of matching corners to establish correspondences. We are using standard corner detection routines (*Plessey* corner detector [HORATIO]) and compute correspondences using correlation based algorithms given in [Faugeras92].
- From the $3D$ structure, projections are taken onto the ground parallel plane, for determination of the path that the robot should follow through the obstacles. Clusters of points in the $2D$ scene are identified as single objects. The complete problem of navigation involves path planning algorithms for guiding the robot through the obstacles.

  In this work, we examine this problem on a smaller scale. We have implemented an automatic routine to guide the robot through two objects. This can be used as the basis for a more complete navigation system. In addition, we have also provided a user-controlled routine where a person can see the projected view on the screen and give commands through a mouse to move the robot. This is similar to the local path planning approach, in the sense that we update the scene information after a small motion by the robot, recompute the structure and move again.

1.3 Structure of the Report

This report is organized as follows:

- We begin with some preliminary explanation of the existing camera models and their relevance to our problem. We further explain the camera geometry used and the process of calibration. We also examine different approaches to recovering the $3D$ structure of the scene. **(Chapters 2, 3, 4)**.
- Next, our basic algorithm and the implementation issues involved, are explained. **(Chapters 5, 6)**. We also take a look at some significant results.
- Finally we end with a look at the future prospects in this direction in **(Chapter 7)** as also the restrictions imposed upon the system on the basis of our assumptions and hardware limitations.

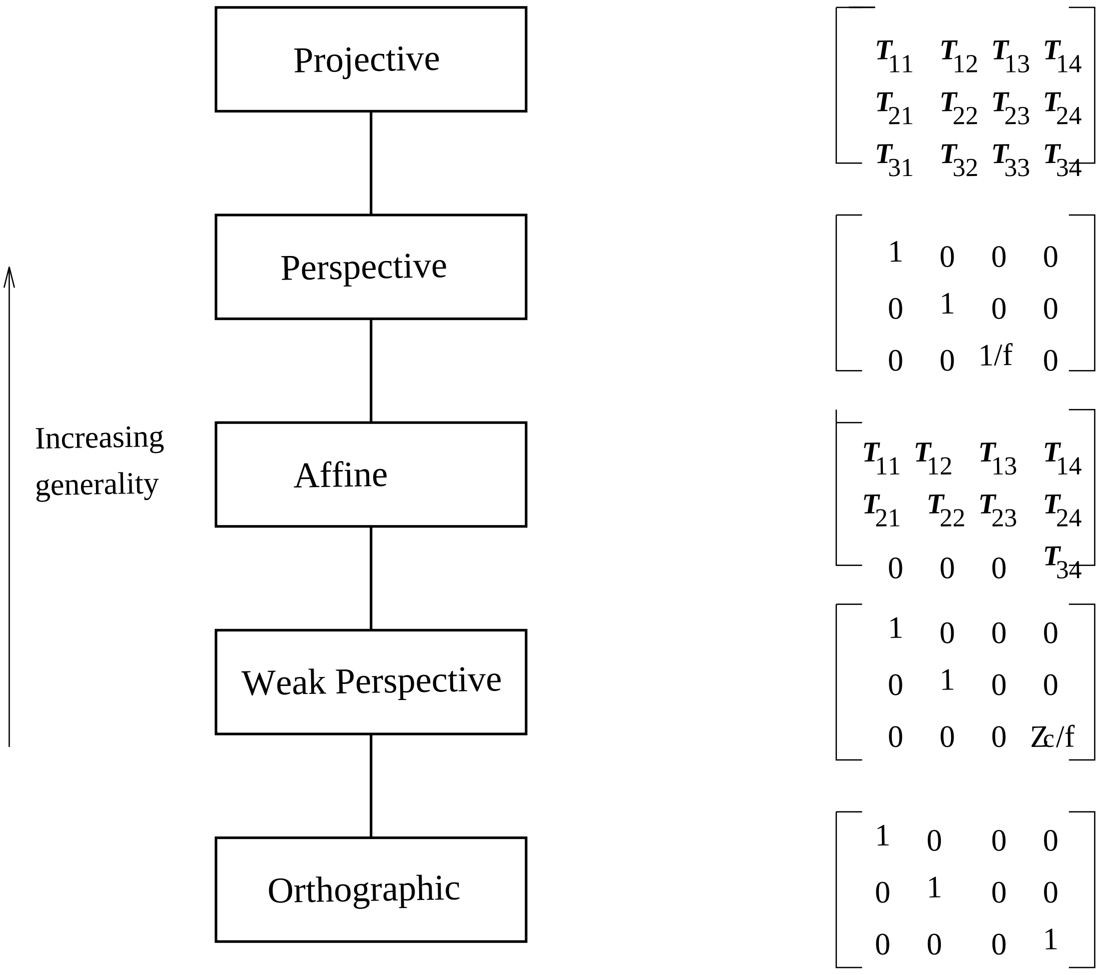
Chapter 2

Camera Models

Selecting the appropriate camera model for a computer vision system is very important and basic to the problem, for it influences not only the complexity of the calculations, but what results may be obtained, and how robust they are for the given working environment.

Our goal was to design an algorithm that can work in the most general possible framework, without any loss of robustness. Therefore, the choice of a suitable camera model was essential, such that the assumptions underlying that model could be satisfied for our application. There are five popular camera models, whose hierarchy is as shown in Figure 2.1.

Starting from the orthographic model to the projective model, each level in the hierarchy reduces the restrictions placed upon its validity. The orthographic model is the least general, while the projective camera makes the least assumptions about its working environment. For our purpose, two models – the affine model, which is a generalization of the orthographic model, and the projective model, appeared to be useful. Extensive work on $3D$ structure recovery using both the models, have been carried out ([Shashua92], [Beardsley94], [Faugeras92]).



Projective
Perspective
Affine
Weak Perspective
Orthographic
Increasing generality
$\begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \end{bmatrix}$
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix}$
$\begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ 0 & 0 & 0 & T_{34} \end{bmatrix}$
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & Zc/f \end{bmatrix}$
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Figure 2.1: Hierarchy of Camera Models.

2.1 The Affine Camera

The generalised T matrix represents a tranformation from $3D$ space to the $2D$ image space. In homogenous coordinates,

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.1}$$

where, $[X, Y, Z, 1]^T$ is a $3D$ point and $[x, y, 1]^T$ is the image point. The matrix T defines the transformation and consists of rotation, translation, scaling and projection. The affine camera can be obtained by constraining the T matrix such that $T_{31} = T_{32} = T_{33} = 0$. thereby reducing the degrees of freedom from eleven to eight.
We assume the mapping of the world coordinates to image coordinates to be then represented by an affine transformation

$$\mathbf{x_i} = \mathbf{MX_i} + \mathbf{t} \tag{2.2}$$

where M is a general 2×3 matrix with elements $M_{ij} = T_{ij}/T_{34}$, while $t = (T_{14}/T_{34}, T_{24}/T_{34})^T$ is a general 2-vector representing the image centre. Importantly, the affine camera preserves parallelism, i.e., lines that are parallel in the scene remain parallel in the image. This is very useful for cases where mapping from one to another image of the scene is required, or there is some motion in the scene that is to be detected. The validity of the affine camera is strictly under the assumption that $T_{31} = T_{32} = T_{33} = 0$ and is seldom satisfied in general. In fact, it assumes that the depth variation in the object is very less compared to the depth of the scene from the camera ([LS93]).

It can be shown that the $\mathbf{t}$ matrix can be eliminated by taking relative coordinates in the scene. These are preserved across views and depend only on an invariant, which is resolvable as what is known as the affine structure of the scene. The ability of removing the translation effects in the image comes from the fact that the affine camera preserves parallelism, an extremely important feature of the camera. In affine geometry, four points that are non-coplanar, can be used as a basis set for describing a scene and once the affine structure is resolved in terms of these basis points, the transformation from one scene to another is completely known [KvD92], [Shashua92], [LS93].

In our case, the affine structure is not applicable. This is because, during navigation, the entire scene and the robot are interacting with each other and depth variations an important consideration. It is the depth of an object that will help the system to ultimately decide whether it should change direction to avoid colliding with an object , or can safely move closer to it.

2.2 The Projective Camera

A projective camera transforms a $3D$ scene point $\mathbf{X} = (X, Y, Z)^T$ into an image point $\mathbf{x} = (x, y)^T$. This is the most general mapping from P^3 to P^2, and is given by:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \end{bmatrix} \begin{bmatrix} X_1 \\ x_2 \\ X_3 \\ X_4 \end{bmatrix} \tag{2.3}$$

where $(x_1, x_2, x_3)^T$ and $(X_1, X_2, X_3, X_4)^T$ are the homogeneous coordinates related to $\mathbf{x}$ and $\mathbf{X}$ by $(x, y) = (x_1/x_3, x_2/x_3)$ and $(X, Y, Z) = (X_1/X_4, X_2/X_4, X_3/X_4)$. This transformation matrix T has eleven degrees of freedom and forms the generalized form of the perspective camera, which is the more familiar pin-hole camera, in which the leftmost sub-matrix is an orthonormal *rotation matrix* and the third row is scaled by the inverse of the focal length. This gives us the familiar equations:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{2.4}$$

Inherent to the perspective model, is the assumption that there is a principal point, or the center of projection, which lies on the intersection of the optical axis and the image plane. This point in general depends on the internal parameters of the camera and so the center of projection also requires calibration [Shashua92]. This is examined later in the chapter on calibration, and for further details, see [Tsai87]. This calibration is essential for certain situations, and cannot be avoided.

The advantage of the projective camera is its generality. It provides us with opportunity to examine the camera parameters obtained and make assumptions about them to suit our needs. We adopt this camera model for our work. We examine the calibration procedure to obtain the internal camera parameters in the next chapter, and methods to obtain the projective and $3D$ structure of the scene from the stereo views.

Chapter 3

Camera Geometry and Calibration

3.1 The Camera Geometry

The general model used by us is the same as the pin-hole camera. There are three coordinate systems to be considered - two in $3D$ space and one in the two dimensional image space. For any transformation from a WCS to the image frame coordinates centered at the center of projection, we require three transformations (Figure 3.1). These are described below.

1. First a transformation is carried out to obtain the scene points' coordinates in terms of the $3D$ coordinate system of the camera from the World Coordinate System (WCS) in which the points are originally specified. This consists of a rotation and a translation and can be represented as:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{3.1}$$

   Here, the R matrix is an orthonormal 3×3 matrix and represents a general $3D$ rotation and T is a translation from the camera centre to the origin of the world coordinate system. The camera center here is the origin of the $3D$ coordinate system of the camera. It is also the center of the camera image frame. The X

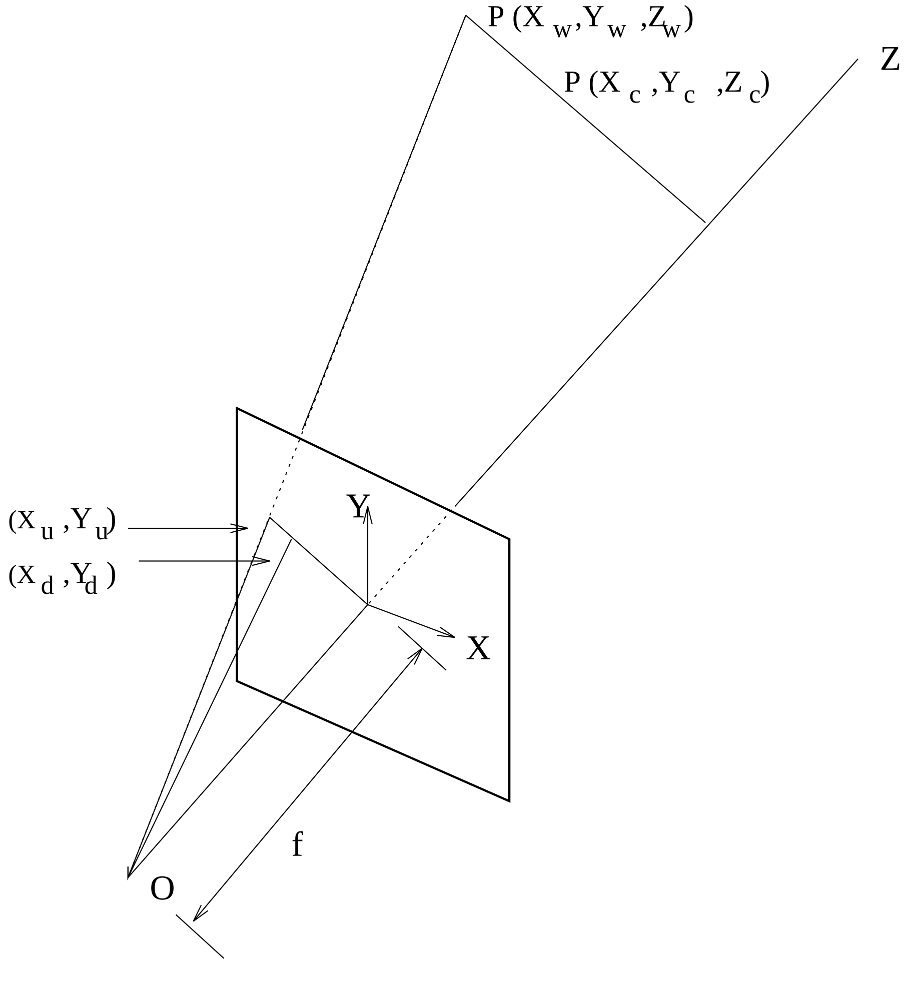


Figure 3.1: The Calibration Camera Model

and Y axes are aligned and the principal axis of the imaging system is the same as the Z axis of the $3D$ coordinate system.

2. Once the coordinates are available in terms of the camera $3D$ system, we obtain, through a perspective transformation, the projection of the points onto the camera image screen located at a distance equal to the focal length of the camera. This transformation is the same as shown before in eqn (2.3).

3. The final transformation is from the camera image system to the computer frame buffer. This involves mapping the image frame origin to the screen center and scaling it in x and y to map from actual image frame sensor points to each pixel. This can be expressed as [Tsai87]:

$$X_f = C_x + \frac{N_f * s_x * x}{N_c * d_x} \tag{3.2}$$

and

$$Y_f = C_y + \frac{N_f * y}{N_c * d_y} \tag{3.3}$$

where (C_x, C_y) are the coordinates of the computer frame center and d_x, d_y are the scalings in x and y from camera frame units to computer pixels. Note that in X_f there is an additional scale factor s_x, which is there to correct the mapping from image to computer frame [Tsai87]. This is due to the conversion from the camera to computer frame.

This gives the complete transformation from a world point to the final pixel as stored in the computer image frame. Another factor, that accounts for *radial distortion*, is also to be incorporated. Distortion in the image occurs during the perspective projection from the 3D coordinate system of the camera onto the camera image plane. We can assume there is only radial distortion, and its effect is that the length of the radial vector from the camera image plane centre to the projected point (X_u, Y_u), the undistorted point, is shortened to give (X_d, Y_d), the distorted point, which lies along the radial vector only. This distortion is quantified by a parameter κ, and is defined by a series. If

$$R = \sqrt{X_d^2 + Y_d^2}, \tag{3.4}$$

then the undistorted and distorted points are related by:

$$X_u = X_d(1 + \kappa_1 R^2 + \kappa_2 R^4 + ...) \tag{3.5}$$

and similarly for Y_d.

The series has higher terms also, but for most applications, the first two terms are sufficient to provide a reasonable approximation to the camera geometry [Tsai87].

3.2 Calibration

For a correct mapping from the world coordinate system to the computer frame buffer, all the parameters involved in the four transformation stages are to be computed accurately. Calibration is a very important feature for vision applications that are based on obtaining metric information from the scene. Once the calibration is done, we can do two things:

- Given a point in the world coordinate system in $3D$ space, its corresponding pixel position can be predicted from the calibrated parameters. This is called forward projection.
- Also, given two or more pixel values corresponding to the same $3D$ point, a *backward projection* can be carried out to obtain the world coordinates by an inverse transformation and triangulation.

3.2.1 Calibration in a Dynamic Environment

Using a calibrated setup has its own limitations, and foremost among them is being bound to the world coordinate system. All the $3D$ points are specified in terms of this system. Usually, for calibration, a standard calibration object is used and the world coordinate system is taken as attached to it. In a scene involving multiple objects, as it is in our case, it is difficult to specify the world coordinates of points not lying on the same object as that to which the $3D$ axes are fixed. Even if this is done and the calibration can proceed, once the scene changes, due to motion in the scene or in the camera, the transformation from the world to camera $3D$ system (in Step 1 above) no longer holds, and the calibration has to be repeated.

In our system, it is neither possible for us to keep a fixed WCS, nor is it feasible to do a recalibration using a changed system after every movement. Thus it is difficult to obtain the $3D$ structure in *Euclidean* parameters, i.e., absolute world coordinates. We are more interested in obtaining a $3D$ structure in terms of some invariant coordinate system, which need not be updated at every step. To this goal, we shall examine several approaches ([PAB94],[Faugeras92],[Shashua92]).

3.2.2 Calibration parameters

Due to the nature of the parameters, they can be classified into

two types, extrinsic and intrinsic parameters. The *extrinsic* parameters are those which depend on the camera geometry, that is, its location with respect to the environment and the world axes system. These are clearly the rotation and translation matrices defined above to map from the world to camera $3D$ coordinates. The *intrinsic* parameters are those which depend on the camera internal characteristics and are defined by the imaging geometry of the camera. The following constitute the internal parameters of a camera:

- **Focal Length:** f, the focal length of the camera, defines the distance of the focal, or the image plane, with respect to the center of projection. There is an assumption here that the image plane is perpendicular to the principal axis of the camera, which generally holds true. This parameter is variable, depending on the adjustments made to the camera setup. For a particular setup , it remains constant. Focal lengths vary from less than 1 cm. to $3-4$ centimeters, depending on the applications and the depth and width of the view desired.

- **Scale factor:** s_x, is used as a correction factor, for mapping the x abscissa from image to computer frame. No correction is needed for the y ordinate as the lines are scanned horizontally and the inter line spacing is fixed for the camera. This parameter also remains constant for a camera. It is generally in the range of 1.01 to 1.1 [IITD96].

- **Camera Centre:** (C_x, C_y) is the pixel position to which the camera image plane centre is mapped. This may be taken as a constant, but only in cases where the level of accuracy desired or obtained is not much and not very important. In case of the accuracy being to the order of micrometers in world coordinates and of milli-pixels in the frame, this needs to be calibrated for better results, and gains a more prominent role in the calibration process. For more detail refer to [Tsai87] and results obtained by Rahul Bhotika and T.R. Vishwanath [IITD96].

- **Number of sensors/pixels**: N_f, N_c are required to obtain the mapping from the discrete camera sensor array to the discrete pixel array of the computer, in the $x-$coordinate. Again, this is a constant for a camera and depends on the hardware. Generally, both are equal to 512.

- **Distortion:** It is assumed that distortion is only in the radial direction and is modeled by the κ series as given above. Normally, the κ value changes with the lighting and the accuracy of the calibration points for a particular setup. However, it is chiefly dependent on the lens and the imaging process. Generally, the value of the first value in this series, i.e., κ_1 , is around 0.1 [IITD96], but varies largely with the camera. It remains more or less constant for a particular camera.

Most of these parameters are calibrated by the calibration procedure that is described below, in brief.

3.3 The Calibration Process

There are many extensive calibration processes developed over the years, that use optimization techniques and error minimization techniques, among others, to obtain the parameters. The technique due to [Tsai87], examines this problem in a different light and simplifies it by the principle of the *Radial Alignment Constraint* (RAC). This principle simply states that there exists parallelism between the line drawn from the principal axis to the $3D$ point (parallel to the image plane) and the line joining the image point and the centre of the image plane. This can be made use of to simplify certain parameters and eliminate most of the internal parameters.

Then the calibration can be carried out in the following two stages:

1. By the RAC, we have:
$$\frac{X_c}{Y_c} = \frac{X_u}{Y_u} = \frac{X_d}{Y_d} \tag{3.6}$$
This enables us to work initially without the distortion and focal length. With this dissociation, the external parameters, except the translation along the principal axis, are determined. This is done by a linear least squares' solution and requires something like 60 or more calibration points. The scale factor s_x is also obtained during this minimization.

2. In the second stage, the focal length and T_z are first initialized under the assumption that the distortion is zero. This is again done through a linear least squares' minimization. Then with this as the initial point, a non-linear minimization based on a standard technique, like the steepest descent, is carried out, to give the final values. The Camera center is not calibrated and is assumed to be at (256,256), the center of the screen.

Chapter 4

The Fundamental Matrix and 3D Structure Recovery

4.1 Projective Geometry

In the general case, the pin-hole camera performs a linear projective transformation, rather than the simple perspective transformation. A general projective linear transformation is given by

$$\mathbf{q} = [s.x, s.y, s]^T = \mathbf{AGM} \tag{4.1}$$

where $\mathbf{M} = [X, Y, Z, 1]^T$ is the $3D$ point whose image point is $\mathbf{q}$, $\mathbf{A}$ is a 3×3 transformation matrix accounting for camera sampling and optical characteristics and $\mathbf{G}$ is a 3×4 displacement matrix accounting for the external parameters - camera position and orientation.

$\mathbf{A}$ is characterized by the camera internal parameters - focal length, Camera Center and the horizontal scale factor. In the case that only the pixel coordinates are known and the camera is uncalibrated both $\mathbf{A}$ and $\mathbf{G}$ are unknown. For a calibrated camera both are known and the internal parameters are generally constant for different orientations also.

Given two stereo views of the same scene a projective construction of the environment is possible [Faugeras92]. This structure is ambiguous as it is in terms of a projective basis of $3D$ points whose actual $3D$ coordinates are unknown. In the case when the transformation to actual coordinates is known for the basis, the ambiguity is removed and we get a Euclidean Structure.

4.2 Epipolar Geometry and the Fundamental Matrix

The geometry of epipolar lines [Luong,Faugeras96, LS93] constrains the way a feature moves between views: given a point in one image, the corresponding point in the second image must lie on a particular line. For a given point $\mathbf{q}$ in the first image, the projective representation $\mathbf{l'_q}$ of its epipolar line in the second image is given by (see Figure 4.1):

$$l'_q = Fq \tag{4.2}$$

Since the point $\mathbf{q}'$ corresponding to $\mathbf{q}$ belongs to the line $\mathbf{l'_q}$ by definition, it follows that

$$\mathbf{q'^TFq = 0} \tag{4.3}$$

We call the 3×3 matrix F which describes the correspondences between the two images as the *Fundamental matrix.* If the internal parameters of a camera are known, this Fundamental matrix is changed to an Essential matrix, as the matrix $\mathbf{A}$ that is defined above can be incorporated into it. Thus F relates two images, of the scene, generally taken from the same camera. In fact, if there are two cameras, and the scene is constant, then the images taken from them can also be taken to follow a similar correspondence and a Fundamental matrix relating them can also be formulated.

The Fundamental matrix relates the images using only pixel values. This is very basic to the problem of obtaining three-dimensional information in the real world. In our case also, during navigation, we work only with the images taken at that time

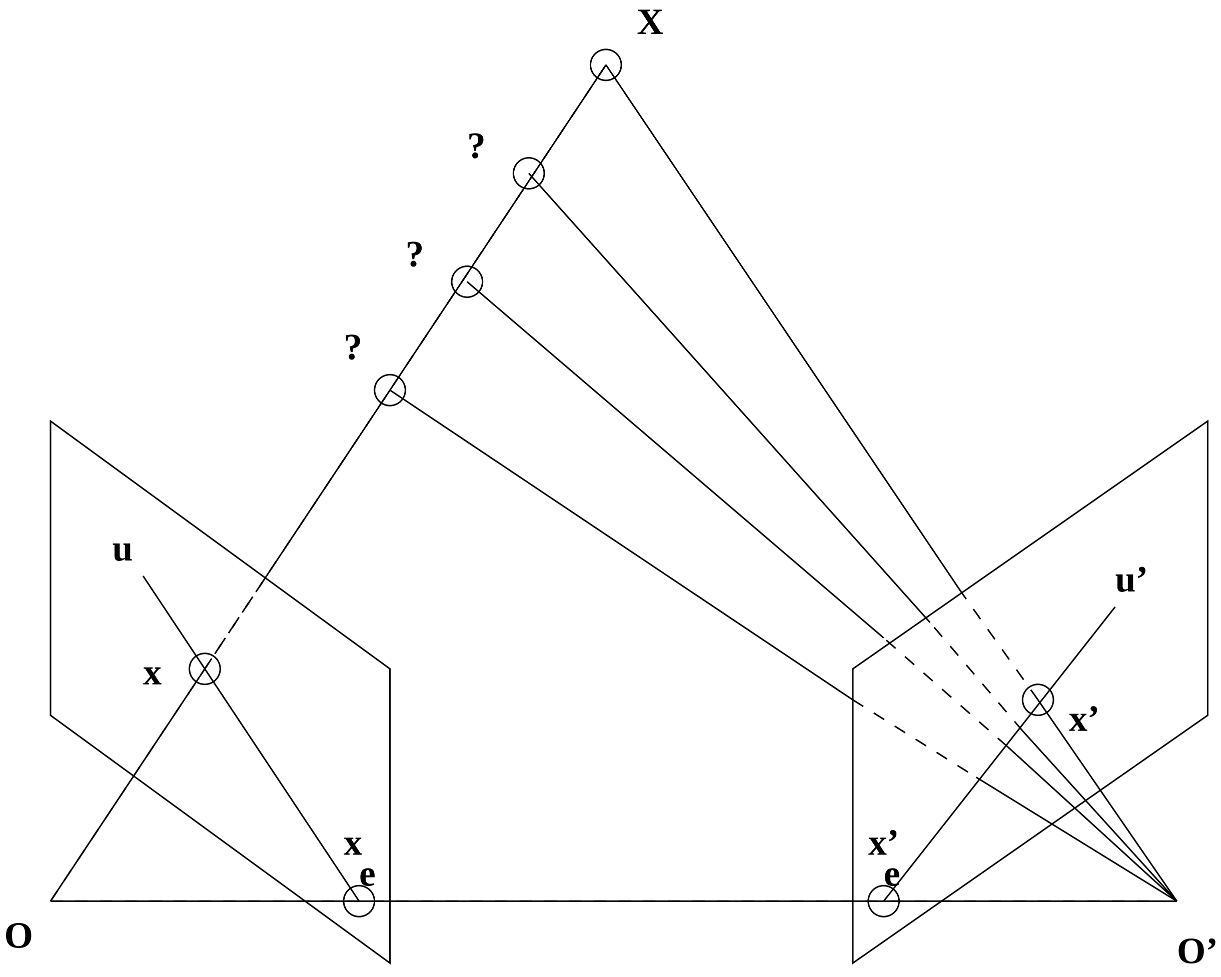


Figure 4.1: Perspective Epipolar geometry: The point $\mathbf{x}'$ corresponding to $\mathbf{x}$ must lie on the epipolar line $\mathbf{u}'$. Points $\mathbf{x_e}$ and $\mathbf{x}'_\mathbf{e}$ are the epipoles through which all epipolar lines must pass.

and have, in general, no information about the $3D$ world. The Fundamental matrix is essential to our case, where the $3D$ world information is not to be obtained from repeated calibration. The internal parameters of the cameras, which are constant, and can be determined for the camera setup, can be used in conjunction with the Fundamental matrix to obtain directions in $3D$ space from pixel measurements alone [Deriche94, Luong,Faugeras96].

Consider that we have two cameras and the optical centers of the two are our two viewpoints. We have for each the generalized projection matrix from $3D$ space to $2D$ space, which contains the transformations described in chapter 3, given by:

$$m = \tilde{P}M \tag{4.4}$$

and

$$m' = \tilde{P}'M \tag{4.5}$$

The Fundamental matrix F, can be related to these two projection matrices. The epipole in the second image is the projection of the optical centre onto the second camera screen. We can thus determine the epipolar line of a point m of the first image,in the second image, and is given by the projection of the ray (C, M) where, C, is the optical centre of the first camera and M is the $3D$ point whose image is m. The optical center is projected to the epipole e' and instead of the point M, we project the point at ∞ along the line (C, M).

It can be shown [Faugeras92] that given at least eight correspondences obtained between the two images of a scene with uncalibrated stereo setup, it is possible to construct for any pair of corresponding points in the images, its $3D$ point. This point is obtained in terms of the unknown projective transformation that is determined from the initial correspondences. In a calibrated setup, completely Euclidean structure can be actually deduced from the above formulation [Faugeras92].

4.3 3D Structure Recovery

This forms the most important part of the system that we propose to build. There are various possibilities and methodologies to choose from, here. As stated earlier, the effort is towards using the information available from calibration, but not to go to the extent of carrying out a very careful calibration, which may have to be repeated. We examine various approaches [Faugeras92, Shashua92, Beardsley94] and outline our own approach based on them. We are considering only the recovery of projective and completely Euclidean structures as per our choice of camera model, and the discussion assumes use of a stereo rig.

4.3.1 Recovery of Projective Structure

[Faugeras92] gives a robust algorithm for estimating the $3D$ structure of the environment in terms of its projective structure. The projective representation of the environment thus reconstructed, is defined upto an arbitrary projective transformation. Let us say there are two images and correspondences between some points in the two images is established. The $3D$ reconstruction makes use of these to get the structure. However, the extrinsic parameters of the camera are unknown and so a complete Euclidean structure cannot be computed. Further let us assume, for the time being, that the intrinsic parameters are unknown too. Using the general projective transformation from P^3 to P^2 (Eqn 4.4), and assuming there are at least five point matches, the matrix $\tilde{P}$, representing the perspective transformation can be deduced upto two unknown parameters and an unknown transformation defined by five basis points, for whom the correspondences are known [Faugeras92].

These two unknown parameters can be determined from the epipolar geometry existing between the stereo pair, which as shown earlier can be deduced in terms of the epipoles, and the Fundamental Matrix F. Now given any other correspondence, its three-dimensional point can be reconstructed in terms of our basis points. In this sense, the structure computed is non-metric. However, if the unknown transformation can be resolved, then the $3D$ structure is completely known in a Euclidean space.

Similar work has been done by Shashua [Shashua92]. The relative structure of the $3D$ scene is recovered from two images in terms of an invariant. He also proceeds without any prior knowledge of calibration. No assumptions are made about the camera geometry either, and the projective to orthographic cases are treated alike.

4.3.2 Recovering Euclidean Structure

Beardsley et al [PAB94, Beardsley94] have given a method of navigating using non-metric structure. It uses Faugeras' approach as above to obtain a projective structure of the scene from stereo images. This is then transformed to an affine structure mapped to a plane at ∞ using arbitrary rotations of both pan and elevation of the stereo head to get new images. A projection of the $3D$ structure along with the camera position, onto a ground parallel plane, enables the robot to move through the free space. Minimal use of calibration is made. This method demands an elaborate and expensive setup, where an active stereo head whose pan and elevation can be accurately controlled, is mounted on the mobile robot.

However, accuracy is not central to the robot navigation problem and an approximately accurate structure is sufficient to obtain information to plan a path through the free space with updates of the scene using fresh views. This forms the basis for our algorithm. We outline in the next section how we have used information available from calibration without forsaking accuracy entirely to achieve the goal of robust navigation and without going through the complex motions as described above.

[Faugeras92] describes a robust method to determine the epipolar geometry using correspondence detection, outlier rejection, and linear and non-linear methods of computation of the epipolar geometry. We have used parts of his work, as suitable to our work, to obtain matches between the corners of the the stereo pair.

Part II

The Proposed Algorithm and Its Implementation

Chapter 5

The Algorithm

Here, we present a description of the final algorithm, giving an explanation of the major steps and the issues involved. So far, we have taken a general view of existing theories and methods that work toward achieving specific goals of our project. Our work has been to modify some of these and incorporate others so as to come up with an algorithm that works in a simple, yet general framework. We work with a stereo setup, with the two cameras statically mounted on the mobile robot head.

5.1 Solution Proposed

An alternative to the approach in [Beardsley94] is to incorporate the information available from off-line calibration done prior to navigation. It is known that if the camera intrinsic parameters as well as the rotation and translation are exactly known, then the Euclidean $3D$ structure of the scene can be estimated exactly. However this requires an online calibration and is both expensive in terms of time and difficult to achieve in a real scenario, which is dynamic in the sense that the robot moves through it.

We calibrate our cameras off-line *once* and make use of only the *internal parameters* of the two cameras and the rotational and translational transformation between the two, and not the camera geometry with respect to a world $3D$ system of coordinates. During navigation, the conditions and the configuration are the same as for the calibration, and hence, these parameters remain constant. The epipolar geometry also remains the same, as the cameras are fixed relative to each other and act as a rigid unit.

This information can be utilized effectively to give us the $3D$ structure of the environment. This structure is not resolvable in terms of a world or a fixed system. However, it can be determined in terms of a $3D$ coordinate system fixed with respect to the robot (and the cameras). The entire scene is resolved relative to the robot. This suffices for navigation purposes as all the robot needs to know is the position of the obstacles, relative to itself, to avoid collisions.

However, the final task in any navigation system is to move from one point to another specified point. For this a hand-eye coordination must be maintained. We have not concentrated on this aspect of the problem in this work. If the initial position is known, we can get the position after a move by adding an increment corresponding to that move, to the last position. In such a scenario, the robot operates in a calibrated space and its world coordinates can be specified. However, the accuracy of such a method would depend to a great extent on the accuracy of the robot motion.

5.2 Major Processing Steps

We now examine the algorithm in more detail. The algorithm has been classified broadly in the following steps:

1. **Off-line Calibration** of the two cameras (including calculation of the epipolar transformation between the two cameras).
2. **Matching Correspondences** using the epipolar lines.
3. **Recovery of the $3D$ structure** using back projection and information from Step (1) above.

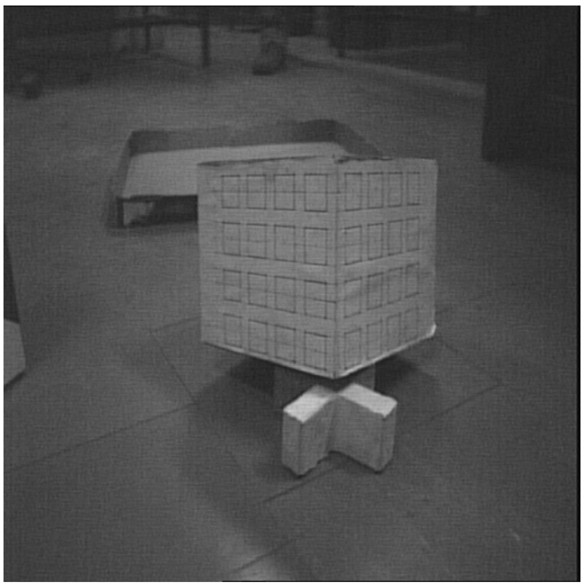


Figure 5.1: Calibration object as seen from left camera

4. **Formation of clusters** by clustering the points obtained to identify objects after projection of the $3D$ structure on to a ground parallel plane.

5. **Navigation** through the free space between two objects by (a) going to a suitable point on the perpendicular bisector of the line joining the centroids of the two clusters and (b) moving in a straight line along the perpendicular bisector, to pass through the obstacles.

We now take a more detailed look at each of these steps.

5.2.1 Off-line Calibration

In the stereo setup, calibration is used to obtain the internal parameters as well as the transformation between the camera coordinate systems of the two cameras. The cameras are calibrated for a focal length approximately equal to the one to be used during the navigation. The calibration is done using an implementation of Tsai's technique based on the *Radial Alignment Constraint* [Tsai87].

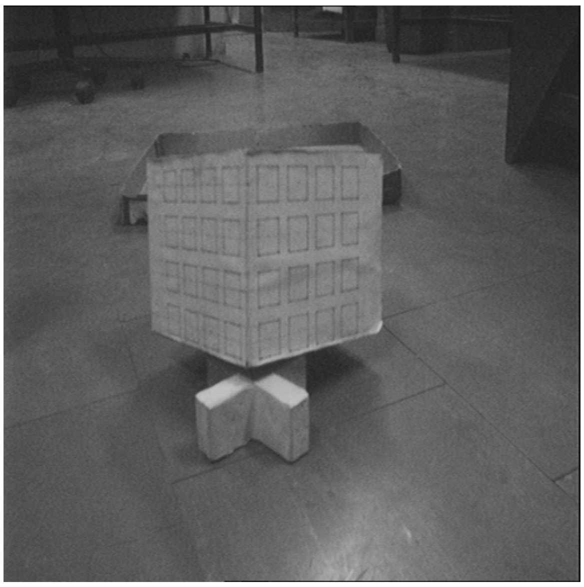


Figure 5.2: Calibration object as seen from right camera

The rigid transformation (**R, T**) between the two camera coordinate systems, gives us the epipolar geometry between the two cameras. This can be calculated during calibration and remains constant during navigation as the cameras are fixed relative to each other and move or rotate only as a unit. We take a stereo pair of images of the calibration object from the two cameras. Each camera is calibrated independently first to relate it to a world coordinate system that is fixed to the calibration object.

If R_1, T_1 and R_2, T_2 describe the transformation from the world coordinate system to the camera $3D$ system, we have :

$$X_1 = R_1 X + T_1$$
$$X_2 = R_2 X + T_2$$

where X_1, X_2, X are the $3D$ coordinates of a point in Camera 1 (C_1), Camera 2 (C_2) and the World respectively, and R_1, R_2, T_1, T_2 are the corresponding *rotation* and *translation* matrices describing the transformation from World to Camera $3D$ coordinate systems.

Then we have a transformation from C_2 $3D$ coordinates to C_1 $3D$ coordinates as:

$$X_1 = RX_2 + T \tag{5.1}$$

where

$$R = R_1 R_2^T \tag{5.2}$$

and

$$T = T_1 - RT_2 \tag{5.3}$$

Thus given a point in the camera $3D$ coordinates of C_2, we can get its $3D$ coordinates in the C_1 coordinate system. We examine later how this helps us during back projection using triangulation.

Secondly , we estimate F, the *Fundamental matrix*, also during the offline calibration. We have, through the calibration data set, a good set of correspondences obtained manually between the two camera images of the calibration object. These are used to get F by a linear least squares solution using the following equation:

$$\begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = 0 \tag{5.4}$$

where $[x_2, y_2, 1]^T$ and $[x_1, y_1, 1]$ are the pixel coordinates of corresponding points in the two images and F gives the Fundamental matrix. Other methods of obtaining the Fundamental matrix, through an unguided matching, are described later.

5.2.2 Matching Correspondences Using the Epipolar Line

We use a correlation based matching routine for the computing correspondences. The matching is done using corners detected in the two images. The set of correspondences is obtained by imposing a high correlation value criterion and matching of the corner strengths. The corner strength gives us an idea of the gradient along the maximum gradient direction. The matching is restricted to a band about the epipolar line (guided matching).

Determining the F-matrix

There are two approaches that we tried for estimating the geometry of the epipolar lines between the stereo pair:

1. We determine the **F**-matrix from the offline calibration as described above.

2. Alternatively, we determine the F matrix from the two scenes themselves by unguided matching using a correlation window. This is done by checking for corner matches in a window around a particular pixel in the second image. By getting corner matches as described in the next section, we determine the F-matrix using a linear least squares method on the correspondences.

   Strict criteria have to be imposed here to get robust F even if the number of correspondences are less. The focus is on getting F for matching correctly later, during navigation. The solution is similar to that used in [Deriche94].

   A supplement to this approach is outlier rejection after the initial unguided matching, by doing a guided matching using the F matrix and then iteratively improving F. Also, non-linear methods may be employed to estimate F, reflecting its rank two property. A linear method only assumes F_{33} to be 1.0 and minimizes eqn 4.2 to get F. The results we have obtained in matching correspondences based on the F matrix estimated only from linear criterion, were found to be good enough for our application and there was no need to improve F further.

Determining correspondences

We use the epipolar geometry (i.e. the matrix $\mathbf{F}$) to do a guided matching for correspondences. Eqn 4.2 says that the correspondence in the right image of point$\mathbf{q}'$ lies on the corresponding epipolar line. Transposing this equation yields the symmetric relation from the second image to the first image. We match corners in the first image, with a possible matching partner, restricting ourselves to a band about the epipolar line $l_q = F.x_1$ of an image point x_1 in the second image. In matching, we use the following criteria for matching corners:

1. The correlation score must be higher than a given threshold.
2. The strengths of the two corners as returned by the Plessey corner detector in the two images must not differ by more than a threshold value.

Using these two criteria, a reasonably good set of matches for the corners is established. Too strict a criteria for matching results in a loss of corners, while a loose one, gives us a poor F and consequently false matches.

5.2.3 Determination of $3D$ structure using back projection

Once the correspondences are obtained, we obtain a $3D$ structure by *triangulation* (see Figure 5.3). First, all image points are converted to a single $3D$ coordinate system, namely that of C_1. Each image point is located at the image plane. Therefore its camera $3D$ coordinates are:

$$\mathbf{x_i} = [u_i, v_i, f_i]^T \tag{5.5}$$

where (u_i, v_i) is the image point and f_i is the focal length of camera C_i. The image point is in the camera image plane and the camera internal parameters are used to map from the pixel image back to the camera image. With the choice of the uniform coordinate system as that of C_1, the following is used to obtain the $3D$ structure.

Using the information from off-line calibration , we transform every point in the $3D$ camera coordinate system of C_2 to C_1. The center of projection of the second camera in its native coordinates, given by $\mathbf{c_2} = [0, 0, 0]^T$ is also transformed to C_1 coordinates to give

$$c_2^1 = R.0 + T = T \tag{5.6}$$

A ray joining the image point and the center of projection defines a ray along which the actual $3D$ point lies. Then using the image points and the centers of projection we get two rays from corresponding points in each camera and generate the $3D$ point with respect to the coordinate system of C_1 by a least squares solution for the intersection point of the two rays from 4 equations in 3 variables.

Given two points (x_1, y_1, z_1) and (x_2, y_2, z_2) in $3D$ space, the line passing through them is given by:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1} \tag{5.7}$$

Similarly, the other ray, for the second camera, passing through (x_3, y_3, z_3) and (x_4, y_4, z_4) is represented as:

$$\frac{x - x_3}{x_4 - x_3} = \frac{y - y_3}{y_4 - y_3} = \frac{z - z_3}{z_4 - z_3} \tag{5.8}$$

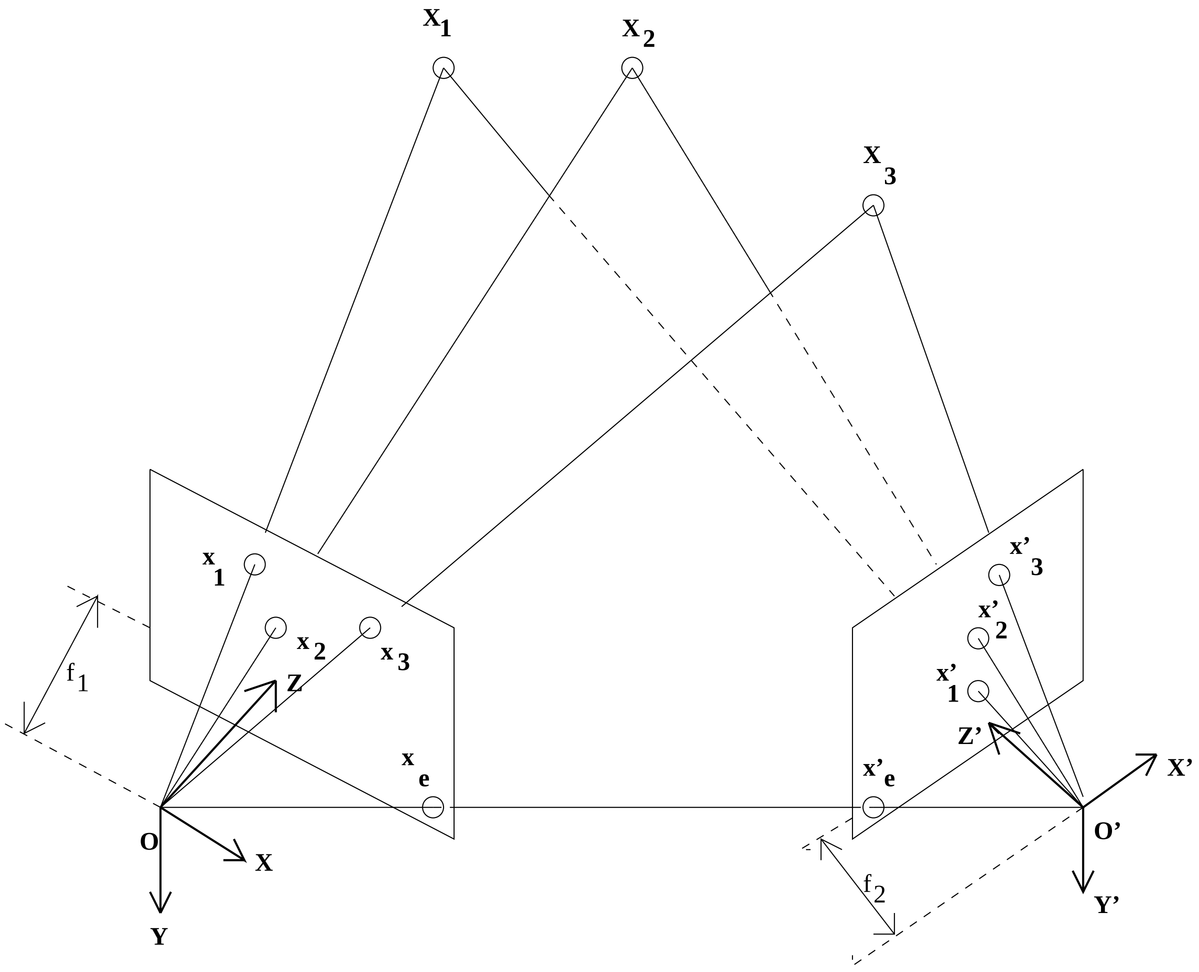


Figure 5.3: Back Projection: Given correspondence $\mathbf{x_1}$ and $\mathbf{x'_1}$ in the two images, we can determine the $3D$ point $\mathbf{X_1}$ if we have the intrinsic parameters of the two cameras and the transformation between them.

These expressions can be simplified to give the following four equations:

$$(y_2 - y_1)x - (x_2 - x_1)y + y_1x_2 - x_1y_2 = 0 \tag{5.9}$$

$$(z_2 - z_1)y - (y_2 - y_1)z + z_1y_2 - y_1z_2 = 0 \tag{5.10}$$

$$(y_4 - y_3)x - (x_4 - x_3)y + y_3x_4 - x_3y_4 = 0 \tag{5.11}$$

$$(z_4 - z_3)y - (y_4 - y_3)z + z_3y_4 - y_3z_4 = 0 \tag{5.12}$$

These are solved for the intersection point (x, y, z). Here, the two rays may not actually intersect, and so the least squares solution will find the midpoint of the perpendicular drawn at the points on the two lines where the lines come closest. Thus from the correspondences, we get back a $3D$ structure of the scene in terms of the first camera. Navigation can be carried out with respect to this structure as it is fixed to one of the cameras, which in turn, is fixed to the robot.

This structure is projected on to $2D$ planes for identifying clusters of points, i.e., the objects. The navigation is caried out in a $2D$ plane that is called the ground-parallel plane. For this projection, it is sufficient to determine the orientation of the optical axis of the first camera, C_1, (in whose coordinate system the $3D$ structure has been computed). This is known to us approximately from the geometry of the setup and may be further refined during calibration. Once the projection is done on to the ground parallel plane, we have the necessary data for the next stage, that is, navigation.

5.2.4 Formation of Clusters

A cluster is a group of closely spaced points (back-projected corners) in the projected $2D$ ground plane. The algorithm we use takes a seed point and determines all the points which are within a given distance from the seed and add them to the same cluster. Now, for each of these points in the cluster, all the points not yet in any cluster are tested again to see whether they are in the same cluster. This way, we find all the points in a cluster recursively. This is repeated till we get all the clusters and there are no more points to be added.

5.2.5 Navigation

We have tackled the navigation part of the problem in two ways. The first method is human controlled, where the user looks at the $2D$ projection and specifies the commands for motion through a mouse. After each motion command is executed, a fresh pair of stereo images may be asked for by the user to get a new structure.

In general, there will be several situations where for a number of reasons the view that is observable to the robot through its stereo images may be insufficient to give a complete picture of the environment. For example, the view angle may be such that some of the obstacles are not visible, even though they are in the robot's path. If we assume from this incomplete picture that the path is clear for the robot to proceed, then there may be a collision. Therefore, in general, the user has to be careful and take small steps and re-estimate the structure.

The other approach is to provide an automated system. This is the final goal of navigation. We have shown an example automated navigation for a particular case of two clusters, and the robot is required to move through them. Here, the robot calculates a path for itself through the obstacles. The system identifies a cluster of points as an object and identifies the location of the centroids of these clusters in the scene. From this, navigation is carried out in two phases. First, an attempt is made to identify a line of motion along which the robot can proceed without any problem.

This line may generally be taken as the perpendicular bisector of the line joining the centroids of the two clusters. During the first phase, the robot is made to move safely to a suitable position on this perpendicular line. In the second phase, the robot moves along the line in free space. The steps taken by the robot are small and re-estimation of the path is done through updates of the structure.

This sums up the basic algorithm. Implementation details and related issues are discussed in the next chapter. The algorithm is simple, and fast. It provides a robust approach towards building a sophisticated and easily adaptable navigation system.

Chapter 6

Implementation and Results

Before we move on to the implementation, it is desired that a clear view of the requirements that the algorithm entails, be looked at. The available hardware and the system configuration are a major factor at the implementation level. Based on the algorithm, certain requirements were drawn out. Subsequent implementation and testing led to modifications and refinements that were not foreseen during the design of the algorithm as outlined in the previous chapter.

6.1 Hardware and Software Requirements

6.1.1 Guidance Vehicle

Essentially, this is the mobile robot. Control is to be provided for motion of the vehicle by specifying direction and distance. Therefore, it should have the capability of turning as well as moving in a straight line. That is, the body moves in a straight line but decides on the direction before.

For this purpose, we are using the mobile robot, or the **Mobot** that is available in the laboratory. Initially it was designed to have a single camera mounted on it. It is undesirable to accurately calculate the epipolar geometry every time we take a stereo pair of images from one camera. Therefore, a new mount was added to the robot to allow two cameras to be mounted statically and fixed with respect to each other, on to the robot head.

The relative positioning of the two cameras affects the epipolar geometry. Also, there is a shift between the two images. This shift has to be reduced as much as possible for a good back projection, as well as for a better correlation based-matching. After testing for various positions, the ideal separation of the two cameras was established to be somewhere around 7-10 cm.

6.1.2 Camera characteristics

We have taken images for different camera positions. From the results obtained, we came to the following conclusions:

- The two cameras have to be similar in the respect that their focusing range and wide angle are similar, so that the views obtained do not differ significantly in object size as well as the range of view. Results show that this is necessary for proper correlation based correspondence.
- Camera lens used should be such that it has a sufficiently large wide angle, so that a larger region can be covered.
- The lighting conditions should not be disparate for the two cameras and the aperture of the two have to be similar. This is necessary for removing noise and better matching. Uniform lighting is also desirable for reducing the distortion in the images.

6.1.3 System Setup

The system setup is divided into two parts. See Figure 6.1.

- **The SUN Ultra SPARC:** The image grabbing and computation for determining the structure and its projection, are obtained on this machine. The machine

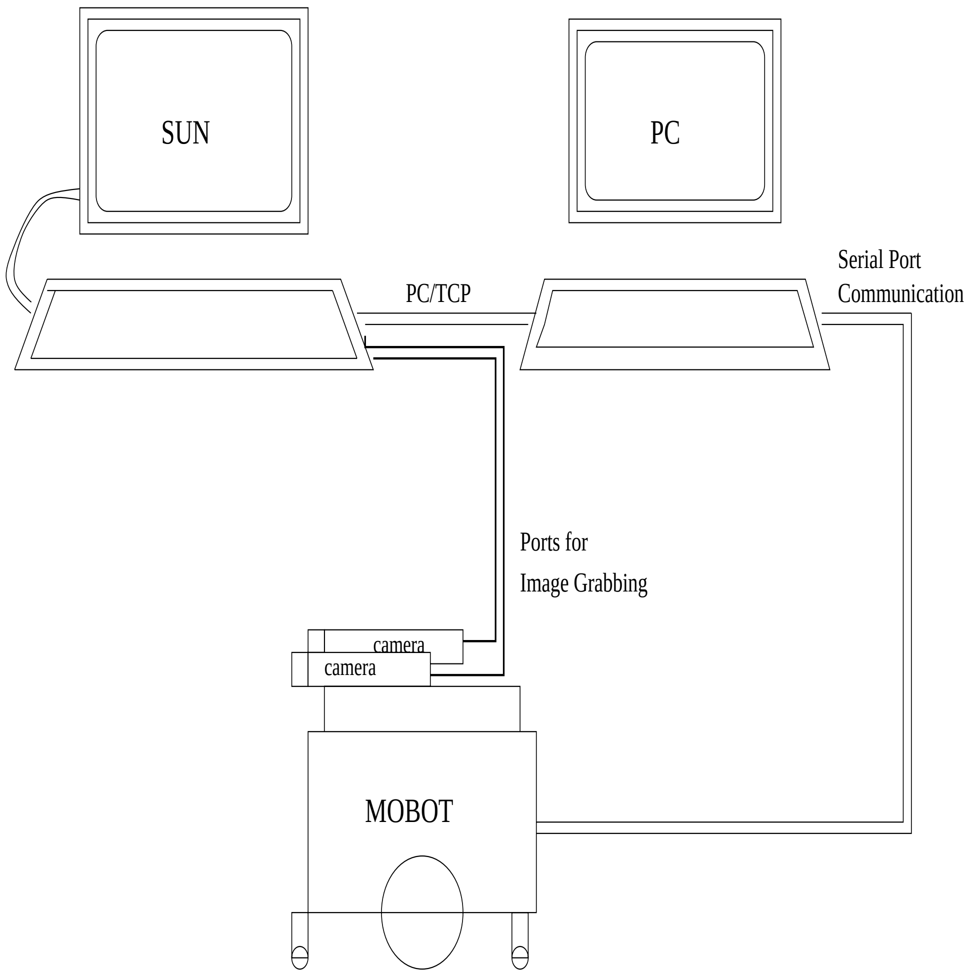


Figure 6.1: The System Setup

has four channels (or ports) to grab images. We are using two of these ports for connecting the two cameras. These ports are multiplexed and can be accessed only one at a time. So, we have to grab the two images sequentially one after another from the two ports by changing the device attribute named PORT.) The entire program is run here, and it also provides the interface for the user.

- **The PC interface:** A PC is used as an interface to the robot. The robot is controlled through a driver that receives motion commands from the PC. The PC is connected to the SUN machine through TCP network protocol. The PC passes the commands on to the robot through its serial port. The robot may be controlled through a linux-based device driver or a DOS-based one. We have implemented a DOS-based driver. (See Appendix A)

The two parts are linked through a TCP socket on the SUN side and a PC/TCP socket on the PC side. In brief, grabbing of images, computation of structure, display of 3D scene, path planning and generation of motion commands etc. are done on the SUN and the PC is used as an interface to run the robot which can be controlled only through a serial port which is not available on SUN, because it does not allow add-on devices.

6.2 Implementation

In this section, we examine each routine as implemented by us and present the results obtained. The entire code was developed in a modular fashion, with the testing of each routine done separately, before it was incorporated in the whole system. The application has been developed using the HORATIO library (for the user interface, standard image processing routines like the *Plessey* corner detector and 3D structure display etc.) and the XIL (SUN) library [XIL] for image grabbing through the inbuilt hardware routines (See Appendix B). Extensive documentation of the HORATIO library is available online as well in hard copy in the laboratory [HORATIO].

6.2.1 Image Grabbing

The Image Grabbing is done using the XIL library available on the SUN-SPARC. We are grabbing the two stereo images by using two of the ports available on the SUN. The ports are multiplexed and images have to be grabbed one at a time by changing the *PORT* attribute of the image. Refer to appendix B for the image-grabbing routine used. The XIL library uses routines to utilize the inbuilt hardware for image grabbing [XIL].

6.2.2 Matching correspondences

Corner Detection

The standard *Plessey* corner detector is applied to identify corners in the two images. Here, we set the appropriate Gaussian mask to smoothen the image so that we get the right number of corners. For example, if we use very less value of standard deviation of the Gaussian mask, the image is not smoothened properly and we get corners on the floor and even small come as corners which cause problems during navigation. We have used a standard deviation of around 5 for normal objects that we used to get corners only on the objects and not on the floor etc. The size of the Gaussian mask used has to be about 3 times its standard deviation for good results.

Determining the Fundamental Matrix

As discussed in the previous chapter, there are two approaches that one can use to determine the F matrix. We found that the F matrix determined from the calibration is not very good for matching in the whole space. This is because the correspondence points during calibration are specified manually and hence are not very accurate. Secondly, all the points are within a small $3D$ region.

The other approach is to use unguided matching. The matching is totally *unguided* in the sense that it is does not use the epipolar geometry information. Using the correspondences, we obtain the Fundamental matrix F by an unguided matching. We obtain this by keeping a large number of objects with sharp corners at various distances to cover the entire view space of the cameras. These yield a good set of initial matches and therefore a good F matrix. This F matrix is further improved by performing a guided matching on the same scene. This entire procedure is carried out off-line and there is no further need to refine the F matrix during navigation.

6.2.3 Off-line Calibration

We take a stereo pair of images of a calibration object with the cameras placed in the same relative orientation as they will be during the navigation. For testing we have taken images of the calibration object that consists of a cube with the calibration points marked on it. The cameras were kept at various relative distances and angles. The optimum value was discovered to be around 8 cm. apart with some tilt to focus upon the scene in front. This tilt was needed to get objects close together into the view frame.

Based on these results, the mount for the cameras was built on specification. (See Figure 6.2 and 6.3) . It has an adjustable hinge to set the tilt of the cameras with respect to the ground and also for the relative angle between the two cameras. The tilt value is taken at somewhere between 20 to 30 degrees with the horizontal. For the projection onto the ground parallel plane, a transformation must be first applied to

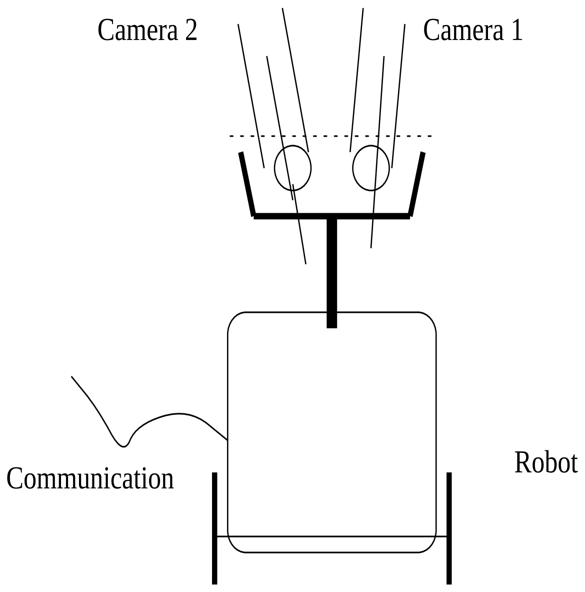


Figure 6.2: Robot-Front View

get the $3D$ structure aligned with the ground parallel plane. Then it can simply be projected downward.

The calibration routine was available to us [IITD96]. This routine is used to get the extrinsic and intrinsic parameters for each camera with respect to the global $3D$ coordinate system fixed to the calibration object. As outlined above, we calculate the transformation from camera C_2 to C_1 coordinates using their rotation and transformation matrices. For the calibration object, we also manually obtain a set of correspondences. The F matrix is calculated by a Least Squares fit using these correspondences. We have tried using this F matrix for our guided matching. However, the results obtained were not encouraging. The reason for this may be one of the following:

1. The values for the correspondences are generated manually, during calibration and do not have sub-pixel accuracy. This is good enough for the calibration procedure, where the error of upto 2-3 pixels can be tolerated. However for the contents of the F matrix, which are sometimes of the order of 10^{-3}, this level of

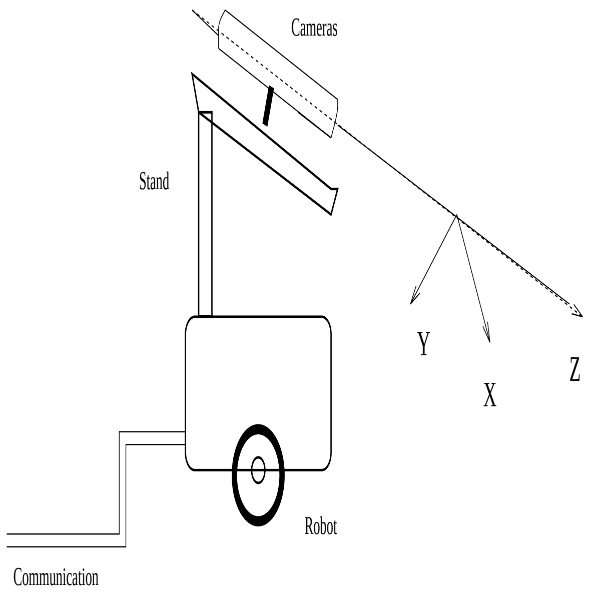


Figure 6.3: Robot-Side View

accuracy is not good enough.

2. Also, the calibration points being close together, and lying on two perpendicular planes, do not give adequate idea of the entire field of view. We have found that for corners which are at the distance similar to what the calibration object was during the calibration, we get robust matches, but for objects at a different distance, we get false or no matches.

6.2.4 Back Projection

The back projection routine uses the transformation between the cameras to do a triangulation. For the calibration images, we have the transformation from the world $3D$ coordinates to camera 1 (C_1) $3D$ coordinates. As the back projected structure is obtained in terms of C_1 coordinates only, we determine the correctness of the obtained structure. We have calculated the average root mean square error between the actual and back projected coordinates. The errors have come out to be in the range of $0.1 - 0.4$ cm. which is very good considering a forward projection error of two pixels.

6.2.5 Interfacing the Robot

The SUN Ultra Sparc communicates with a PC that is linked to the mobile robot (MOBOT) through a serial port. The communication between the SUN and the PC is based on the PC/TCP protocol. We have used BSD sockets for this purpose. The main program on the SUN machine, acts as a client process and connects to the server running on the PC. The motion commands are generated on the SUN machine and passed on to the PC. The server on receiving the command string, translates it into suitable commands for the mobot and writes them to the serial port. The server program also ensures proper communication with the Mobot by initializing it to proper Baud rates and parity bit etc. Please refer to the Appendices C and A for a more detailed explanataion of the interfacing of the Mobot and its motion comands.

6.2.6 Navigation

As discussed earlier, we have implemented two methods of navigation.

1. The first method is to control the robot manually. On the click of a mouse button, images are grabbed, correspondences are determined, $3D$ structure is recovered and clusters of objects are identified. Then, the user can specify the robot motion by giving the appropriate commands through the mouse. The commands available to the robot are move-forward, move-backward, rotate-left, rotate-right and stop. After motion, the user can again give the appropriate commands for either image-grabbing and processing to refresh the images or more movement commands.

2. The second method is an automatic version where the goal is to pass through two objects. Here, we grab the images and find the two objects by clustering nearby points. The goal is accomplished in two phases. In the first phase, we try to go to a suitable point on the perpendicular bisector of the line joining the centroid of the two clusters. This is accomplished by appropriate rotation and move-forward of the robot. In the second phase, we try to go straight through the obstacles by moving along the perpendicular line.

6.3 Results

We present some data on the results obtained here. The testing ws carried out initially on the AGV images included in the HORATIO library. These were used to verify the correctness of our correspondence routines and the calculation of epipolar geometry. Then actual images were taken and the program was run off-line. The final implementation was online and the results obtained are given below.

6.3.1 The Fundamental Matrix

We give here values of the Fundamental matrix for AGV, test and final images:

For AGV images, the matrix is obtained through only unguided matching:

$$F_{ung} = \begin{bmatrix} 0.0000240 & -0.000535 & 0.045813 \\ 0.0005715 & 0.0000194 & -0.089955 \\ -0.0547374 & 0.0805798 & 1.000000 \end{bmatrix} \tag{6.1}$$

For the test images, the matrix is obtained from off-line calibration:

$$F_{calb} = \begin{bmatrix} 0.000003 & -0.000078 & 0.015203 \\ 0.000080 & 0.000008 & -0.027293 \\ -0.017005 & 0.021334 & 1.000000 \end{bmatrix} \tag{6.2}$$

For the final navigation, we calculated F from both calibration and through unguided and then guided matching. The values obtained are:

$$F_{calb} = \begin{bmatrix} 0.0000110751 & -0.0000100878 & -0.0035248935 \\ 0.0000087509 & 0.0000028498 & 0.0017727935 \\ -0.0015968762 & -0.0041731132 & 1.00000000 \end{bmatrix} \tag{6.3}$$

$$F_{guid} = \begin{bmatrix} 0.0000005543 & -0.0000149637 & 0.0295496603 \\ 0.0000331725 & -0.0000003639 & -0.0188252918 \\ -0.0339422599 & 0.0158080660 & 1.0000000000 \end{bmatrix} \tag{6.4}$$

It is clearly seen that there is a difference in the two matrices obtained from calibration and from off-line guided matching. F_{guid} is much more robust and gives accurate epipolar lines.

6.3.2 Epipolar Geometry and Back Projection

The two cameras were placed $10 - 12cm.$ apart and their axis were almost aligned, with the optical axes making around 10 degrees of angle with each other. The rotation and translation matrices obtained were as follows:

$$R_{epi} = \begin{bmatrix} 0.997799 & 0.032258 & -0.050401 \\ -0.042368 & 0.995577 & 0.088490 \\ 0.053973 & -0.086745 & 0.994752 \end{bmatrix} \tag{6.5}$$

and

$$T_{epi} = \begin{bmatrix} 11.650555 \\ -1.314000 \\ -5.542301 \end{bmatrix} \tag{6.6}$$

These values agree excellently with the actual measured values between the two cameras. The back projection done on the basis of these was checked out for the calibration object. There we have the back projected coordinates and the transformation from world to C_1 coordinates. These were compared and the average error over 80 correspondences was less than $0.3cm$. For actual images, the back projection was verified by measuring actual distances between corners of the objects and compared with the obtained values. They were found to be correct within an error of $0.4cm.$.

6.3.3 Navigation

Here we present some results from an automated run. The scene consists of two clusters of objects. One of them blocks a straight path. The robot is shown to navigate in two phases. During phase I, the navigation system identifies the clusters, finds an axis of motion that is perpendicular to the line joining the clusters and move to align itself with it (see Figure 6.6. The images from the left (Figure 6.3.3) and right (figure 6.5 cameras with the obtained correspondences are shown. The $3D$ structure projected to the ground plane is shown in Figure 6.5

Phase I

The following information is generated about the structure to determine the motion:

Cluster 1 at location $x_1 = 25.388421, y_1 = 134.921660$
Cluster 2 at location $x_2 = -15.300971, y_2 = 78.991673$
Distance between two clusters = $69.164948cm$.
Center of Cluster $X_c = 4.910808, Y_c = 103.907726$

The following motion commands are generated for the robot to align itself with the axis normal to the line joining the clusters and passing through (X_c, Y_c):

1. Rotating right by angle 21.955544 degrees.

2. Moving forward by amount 87.134391 cm.

3. Rotating left by angle 103.510417 degrees.

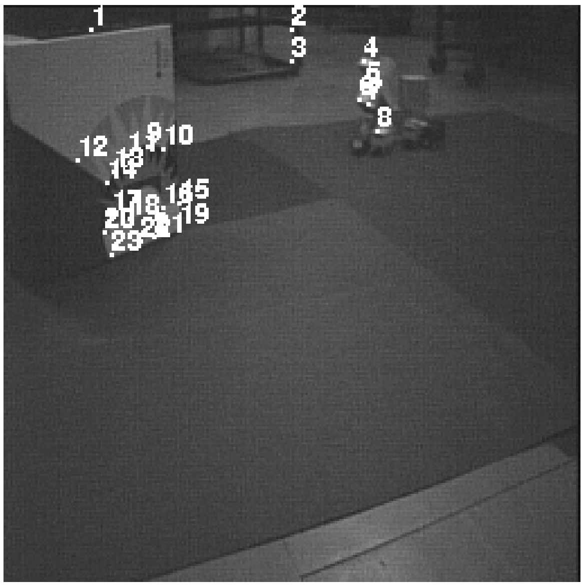


Figure 6.4: Phase I - Left View

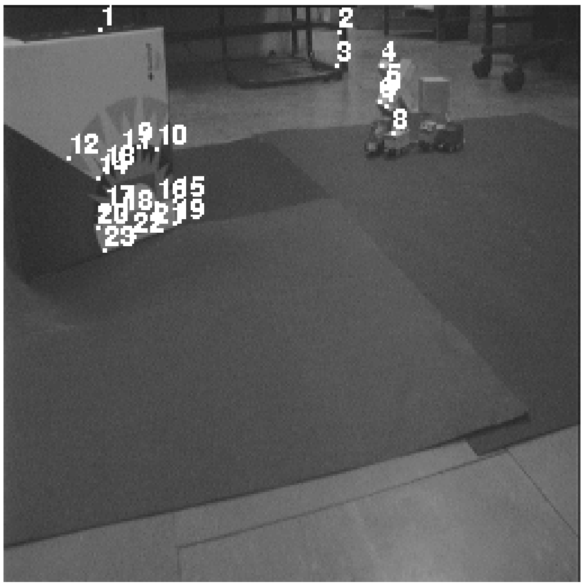


Figure 6.5: Phase I - Right View

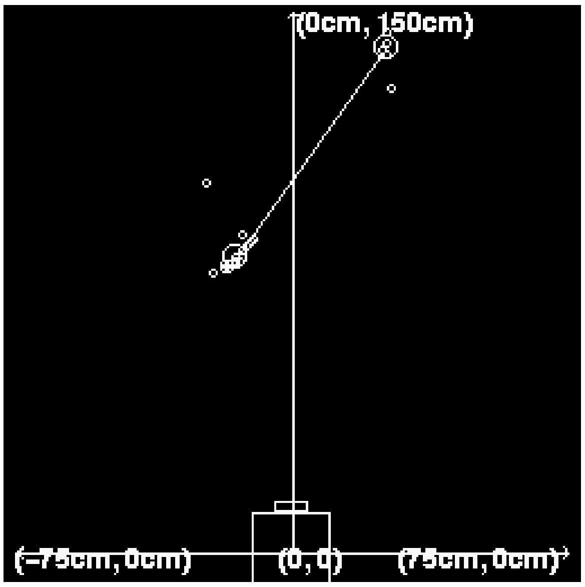


Figure 6.6: Phase I - Projected Structure

Phase II

Now, images are taken again, in every stage, to get a new estimate of the scene. Phase II involves making small corrective moves, and then steps of 10 cm. to guide itself through the two clusters.

This is accomplished in the following steps:

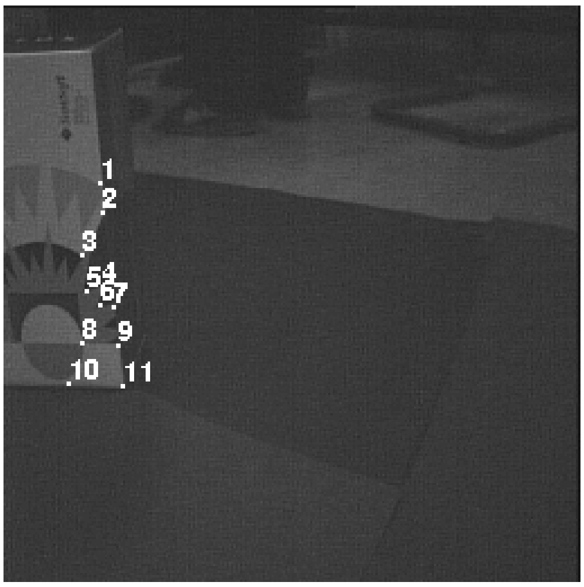


Figure 6.7: Phase II, Step 1 - Left View

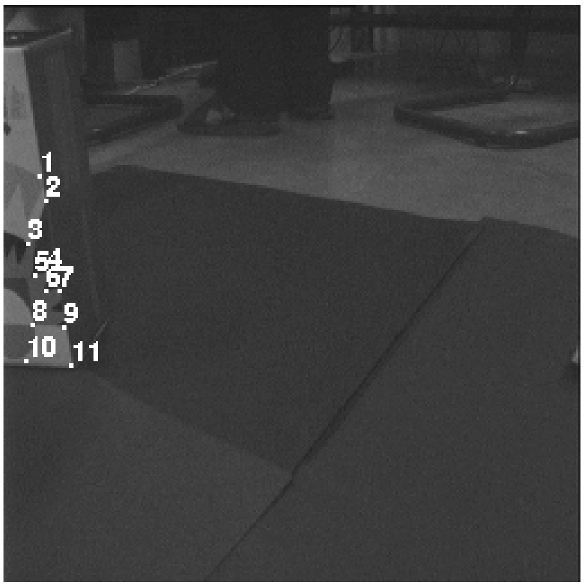


Figure 6.8: Phase II, Step 2 - Right View

1. **Step 1:** Only the left cluster is visible here. A small rotation to the right is required to align with the direction of straight motion (see Figures 6.3.3, 6.8). The information generated is:
   Found cluster 1 at location $x = -14.98$
   $y = 51.158729$ (Figure 6.9).

   The motion command generated is:
   Rotating right by angle 10 degrees.

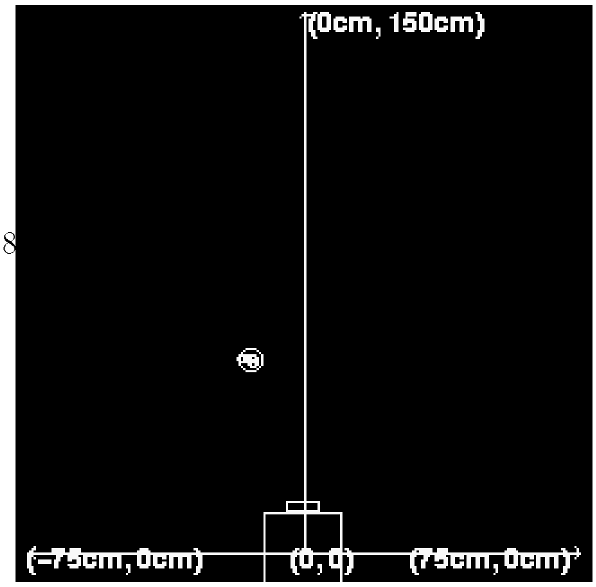


Figure 6.9: Phase II, Step 1 - Projected Structure

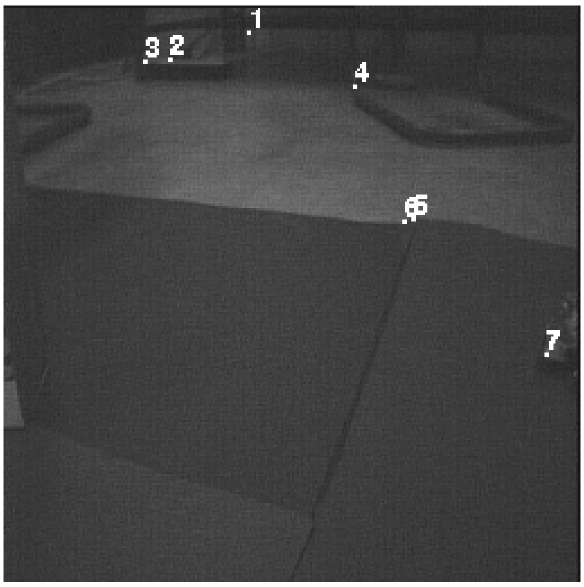


Figure 6.10: Phase II, Step 2 - Left View

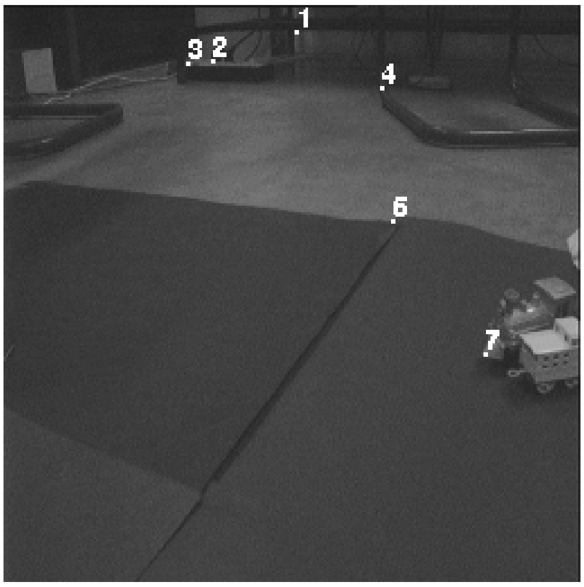


Figure 6.11: Phase II, Step 2 - Right View

2. **Step 2:** The robot is sufficiently closer to the objects, and no cluster is visible, that obstruct it (see Figures 1, 6.11, 6.12). The information generated is:
   Unable to find proper clusters.
   and the subsequent motion command is:
   Moving forward by amount 10 cm.

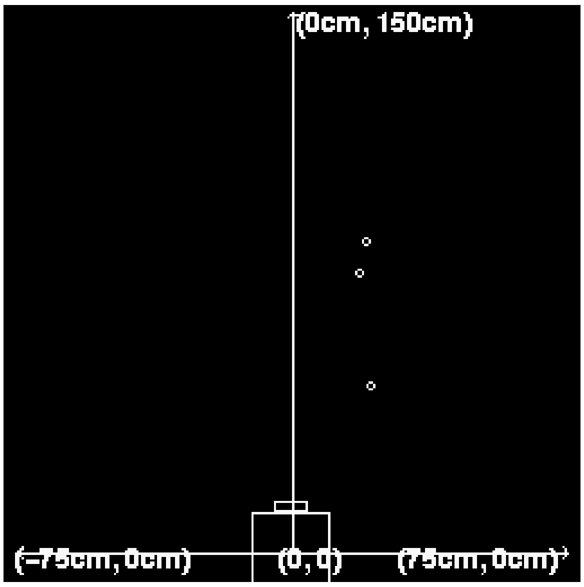


Figure 6.12: Phase II, Step 2 - Projected Structure

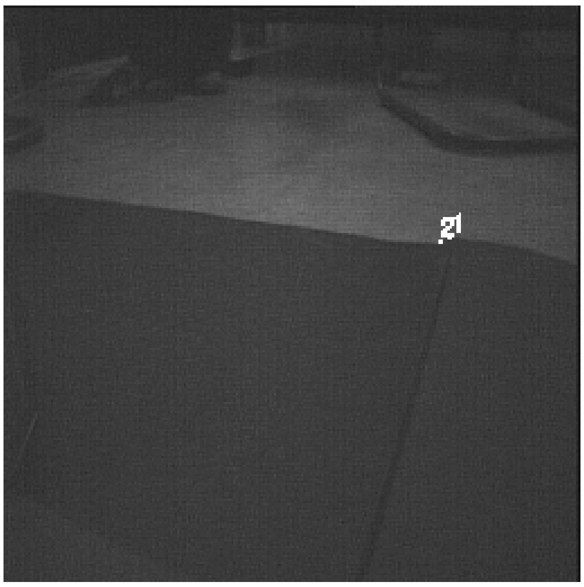


Figure 6.13: Phase II, Step 3 - Left View

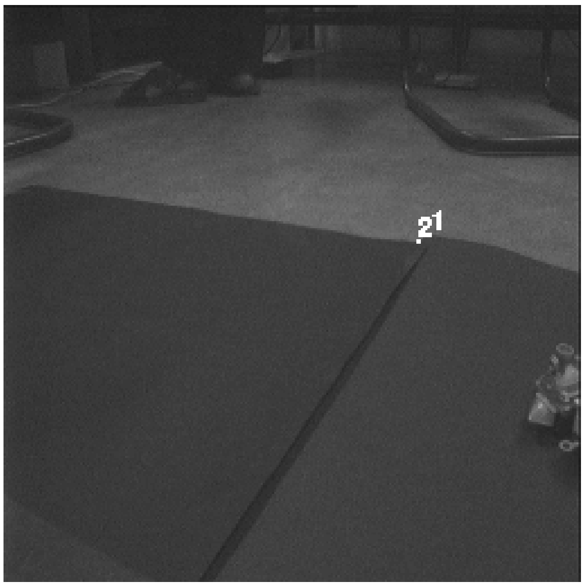


Figure 6.14: Phase II, Step 3 - Right View

3. **Step 3:** Again no clusters are obstructing its path (see Figures 2, 6.14, 6.15). We have:
   Unable to find proper clusters.
   and the motion is:
   Moving forward by amount 10 cm.

The last step occurs again and again until the robot has crossed the obstacles.

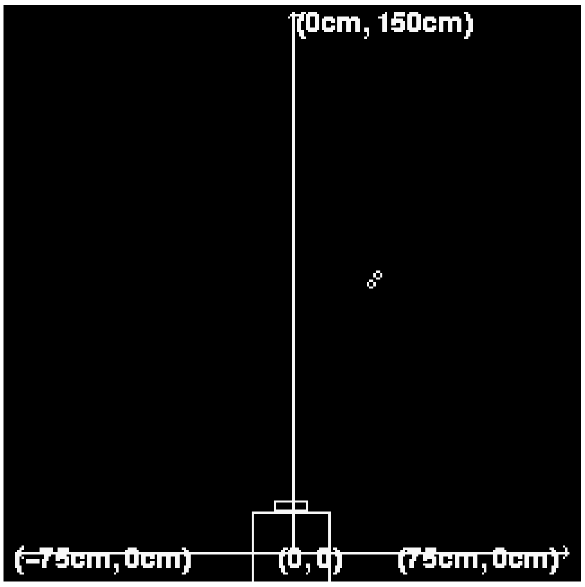


Figure 6.15: Phase II, Step 3 - Projected Structure

Chapter 7

Conclusion

Our work has demonstrated that a robust navigation system can be built without employing elaborate procedures for either accurate calibration or for obtaining the $3D$ Euclidean structure. The robot head also is simple, with statically mounted cameras. Using off-line information of the camera internal parameters, the epipolar geometry and the Fundamental matrix, we provide a simple basis for a robust navigation system.

We have implemented navigation based on user-controlled robot motion. We have also provided an example automated version for a specific case. This can form the basis of a completely automated navigation system.

There were certain limitations, which caused some hindrance. The mobile robot being used is not very precise in its motion and an automated version suffers because of this. A very tight feedback control has to be provided for updating the structure as the robot moves.

Also, wide angle lens for the camera are required. This is very important because as the robot comes closer to the obstacles, the angle that they subtend increases and they go out of the view.

Another suggested addition to the structure estimation is possible use of edge

detection in the scene to form a more complete view of the environment.
Applications of such a robot navigation system are manifold. Our work provides a prototype for building simple systems that can be used in various environments to replace human beings and also in hazardous environments where human beings may be restricted. The robot may be used for transporting payload, or carrying out simple tasks which require working in a cluttered environment.

Appendix A

PC/TCP Module and Mobot Control

The PC/TCP and robot navigation program (combined) running on the PC. The mobot program accompanying the mobot manual was in **GWbasic** and it had to be converted to C. We give here a short listing of the program to explain this part of the interface.

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <pctcp/types.h>
#include <pctcp/pctcp.h>
#include <pctcp/sockets.h>
#include <pctcp/error.h>
#include <pctcp/options.h>
#include <bios.h>
#include <dos.h>
#include <conio.h>
#include <stdlib.h>
#define BUFLEN  20
#define SETTINGS 0x5a
```

```
/* settings are according to the parameters of the robot serial
   communication i.e. 300 baud rate, even parity, 7 bits data
   transfer and 1 stop bits */

main()
{
  in_name  addr;
  int nd,len,num,total,len1,send;
  struct addr a;
  char ch, buf[BUFLEN];
  long int lp;

  addr = nm_prs_addr("202.141.65.70");

/* this the network address of SUN machine (nilaya in our case) */

  a.fhost = addr;
  a.fsocket = 5911;
  a.lsocket = 0;
  a.protocol = STREAM;

  bioscom(0, SETTINGS, 0);

/* initialize the PC serial port according to the robot serial
   communication requirements */

  printf("tcp no = %d \n",SOCK_TCP_FTP);
  nd = net_connect(-1,STREAM,&a);

/* this tries to connect to the SUN socket */
```

```
if (nd == -1 )
{
  pneterror("netconnect refused ");
  exit(1);
}
else printf ("success \n");

while (1)
{
  len = net_read(nd,buf,BUFLEN,(struct addr *) 0,0);

  /* read the character string from the socket
     buf[0] contains the character indicating the direction of
     motion, i.e F(orward), B(ackward), L(eft rotation), R(rotation)
     and the remaining string contains the amount of motion.
  */

  if (len  < 0)
     pneterror("net_read");
  else
  {
    printf("received %s", buf);
    num = atoi(&buf[1]);

  /* now num contains the amount of motion as integer */

  /* The robot can accept motion command only from 0-127 at a time.
      So, if the amount of motion is more then we have to send
      multiple commands for motion.  So, this loop was required
      for multiple commands to robot.
```

```
*/

while (! (num <= 0))
{
  if (num <= 127) send = num;
  else send = 127;

  bioscom(1, buf[0],0);
  /* send F,B,L or R to the robot for indicating the direction
     of motion */

  ch = (char) send;
  bioscom(1,ch, 0);

  /* send the character from 0-127 indicating the amount of motion */
  printf("%d %c\n", send, ch);
  for (lp = 0; lp < 800000; lp++) {}

   /* this delay is required as the robot can accept the next
      command only after some time */

  num -= 127;
}

num = bioscom(2, ch, 0);
ch = (char) num;

 /* we read the character sent by the robot which determines the
    status of the transmission.  The character can be G(ood),
    F(ull), AND B(ad) */
```

```
     printf("Read %c\n", ch);
     printf("wrote all the data\n\n") ;
   }
  }
  len  = net_releaseall();
  if (len == -1)
  {
    printf("error in releasing \n");
    exit(1);
  }
}
```

Alternatively, instead of the bioscom function, we may write directly to the COM 1 port. An assembly code can be included in the program, which initialises the port and then writes the desired characters on to it. It is based on calling an interrupt at the bios level of the PC (**INT 14h**).

Appendix B

The XIL Image Grabbing Library

This library is used to capture images from the SUN ports. The captured images are then directly converted into the horation image type for further processing using h oratio image processing routines and our own processing routines(e.g. matching, 3D structure etc.).

Here, we are including the commented routine used to grab an image. To get faster grabbing, this routine was modified and in the final program, all the i nitializations are done once only in the beginning of our program and only the setting of the port attribute, grabbing of the image and rescaling to 512 X 512 is done each time the image is grabbed. For further details and commands, one can see the on-line help available on the SUN Ultra Sparc. We use two ports available at the SUN, each connected to one of the two cameras. For grabbing, only one of the images can be grabbed at a time. So, we switch between the two ports.

```
#include <xil/xil.h>
/* header file needed for xil */

extern Display *xdisplay;
extern Window xwindow;
extern XEvent event;
extern XVisualInfo vinfo;
extern int display_depth;

int Xilcapture(Image *image, int port)
{

      XilSystemState state;
      XilImage rtvc_image, scaled_image, image1;
      XilDevice device;
      char* devname = "/dev/rtvc0";
      int status;
      float xscale, yscale;
      unsigned int width, height, nbands;
      XilDataType datatype;

      int max_buffers = 4;
      int h,w;
      Xil_unsigned8* scanline;
      XilMemoryStorage storage;

      /* initialization of xil state is required before any xil operations */
      state = xil_open();
      if (state == NULL) {
        fprintf(stderr,"Unable to open xil library \n");
         exit(1);
      }

        /* a device handle has to be associated with the state and
           the image grabbing device i.e. SUNWrtvc */
        if (!(device = xil_device_create(state, "SUNWrtvc"))) {
```

```
    fprintf(stderr, "Unable to create a device object\n");
    xil_close(state);
    exit(1);
  }

    /* now the attributes of the device are to be set */

    xil_device_set_value(device, "DEVICE_NAME", (void *) devname);
    /* the DEVICE_NAME for our image grabbing is /dev/rtvc0 */

    xil_device_set_value(device, "PORT_V", (void *) port);
    /* set the port supplied by the parameter to the
       Xilcapture routine */

    xil_device_set_value(device, "MAX_BUFFERS", (void *) max_buffers);

    if (!(rtvc_image = xil_create_from_device(state,
                         "SUNWrtvc", (void *) device))){
      fprintf(stderr, "failed to open SUNWrtvc device\n");
      xil_close(state);
      exit(1);
 }

 /* now rtvc_image is the xil image (type XilImage) associated with
    the image grabbing device */

    xil_device_destroy(device);
    /* the device handle was required only to create the rtvc_image
        with required attributes */

   {
      int format;

       xil_get_device_attribute(rtvc_image, "FORMAT_V", (void **) &format);
       if (format == 0) {
          fprintf(stderr, "Unknown video format, exiting.\n");
          xil_close(state);
          exit(1);
       }
    }
      /* check if the camera is ready */

/* uptil this point can be given as initialization once */
/* the remaining part has to be done each time the grabbing is done */



       status = xil_set_device_attribute(rtvc_image, "PORT_V", (void *) port);
      /* set the port number from where to grab the image */

       if (status == XIL_FAILURE)
       fprintf(stderr,"Failed to set the SunVideo Port attribute \n");

       xil_get_info(rtvc_image, &width, &height, &nbands, &datatype);
       /* get the attributes of the rtvc_image */

       xscale = 512.0/((float) width);
       yscale = 512.0/((float) height);

       scaled_image = xil_create(state, 512, 512, nbands, datatype);
       /* create a scaled_image of size 512 X 512 */

        xil_scale(rtvc_image, scaled_image, "bilinear", xscale, yscale);
```

```
/* here we capture the rtvc_image and scale to scaled_image
    using bilinear interpolation.  xscale and yscale scales
     in x and y directions.  */

image1 = xil_create_child(scaled_image, 0, 0, 512, 512, 0, 1);
 /* now, convert scaled_image(24-bit 512x512) to an 8-bit 512x512
    image (image1) for our use */

 /* image1 has to be exported for getting the pixel values */
 status = xil_export(image1);
 if (status == XIL_FAILURE)
   fprintf(stderr,"Failed to export \n");

  status = xil_get_memory_storage(image1, &storage);
  if (status == FALSE){
     exit(1);
  }

 scanline = storage.byte.data;

 image->width = 512;
 image->height = 512;
 image->type = U_Char;

 /* here we read in the pixel values of image1 and put them in
    image (supplied as argument to Xilcapture) of horatio image
     type */
 for (h = 0; h< 512; h++){
 Xil_unsigned8* row = scanline;
 for (w=0; w< 512; w++){
      (image->array.uc)[h][w] = *row;
```

```
                row += storage.byte.pixel_stride;
            }

            scanline += storage.byte.scanline_stride;
        }

        /* image1 is imported to save memory */
        xil_import(image1, TRUE);

        xil_close(state);
        /* close the state.  This step can be done only once while quitting
           the program */

        return(1);
    }
```

Appendix C

Mobot Transmission Protocol

The mobot we are using is of PRAVAK Cybernetics(P) Ltd. make and uses the RS-232 transmission protocol. The details of the protocol are as follows:

C.1 Transmission Settings

Baud Rate	300
Parity	Even
No. of bits	7 bits + 1 Parity bit
No. of Stop bits	1 (High)
Start bit	1 (low)
Lines Used	T_xD, R_xD, Gnd

C.2 Software Protocol

C.2.1 From IBM PC

- To Transmit Command, send character: F(orward), B(ackward), L(eft), R(ight), S(top).
- To Transmit Motion Units: Character (1) to Character (127).

Two bytes must be transmitted always

- first byte is command.
- second byte is the number of steps (Motion Units).

In case of stop command, the character “S” may be transmitted twice.

C.2.2 From Mobot

After every byte received by the MOBOT, the MOBOT transmits one of the following characters:

- G - good reception.
- B - bad reception.
- F - Buffer full.

C.2.3 Example

To ask the MOBOT to move forward by 85 (decimal) steps, transmit the following bytes:
1-1000110 - Byte for “F” command.
0-1010101 - character corresponding to 85 decimal.

Bibliography

[LS93] Larry S. Shapiro, Andrew Zisserman and J. Michael Brady What can one see with an Affine Camera ? May 1993, Oxford University Tech. Report.

[KvD92] Jan Koenderink and Andrea van Doorn. Affine Structure from Motion. In *Journal of the Optical Society of America.*, 1992.

[HORATIO] P. F. McLauchlan. HORATIO : Libraries for Vision Applications. Technical report, Robotics Research Group, University of Oxford, WP3/OXFORD/930112/HORATIO, 1993.

[PAB94] P.A. Beardsley, A.p. Zisserman and D.W. Murray Navigation Using Affine Structure and Motion. In Proc. 3rd European Conference on Computer Vision. Springer-Verlag, 1994

[Beardsley94] P.A. Beardsley, I.D. Reid, A. Zisserman and D.W. Murray. Active Visual Navigation Using Non-Metric Structure. 29/11/94.

[Deriche94] R. Deriche, Z. Zhang, Q.T. Luong and O. Faugeras. Robust Recovery of the Epipolar Geometry for an uncalibrated Stereo Rig. In Proc. 2nd European Conference on Computer Vision, Springer-Verlag, 1994.

[Faugeras92] O.D. Faugeras. What can be seen in three Dimensions with an Uncalibrated Stereo Rig? In Proc. 2nd European Conference on Computer Vision, pages 321-334. Springer-Verlag, 1992

[Maybank, Faugeras92] O.D. Faugeras, Q.T. Luong and S.J. Maybank. Camera Self-calibration: Theory and Experiments. In Proc. 2nd European Conference on Computer Vision, Springer-Verlag, 1992.

[Longuet-Higgins] A Computer Algorithm for Reconstructing a Scene from Two Projections. *Nature* Magazine, 1981.

[Luong,Faugeras96] Q.T. Luong, O.D. Faugeras. The Fundamental Matrix: Theory and Algorithms, and Stability Analysis. International Journal of Computer Vision, 17, 1996

[Faugeras94] O. Faugeras and L. Robert. What can Two Images tell us About a Third One? 1994.

[Shashua92] Amnon Shashua. Projective Structure from Two Uncalibrated Images. AI Lab, Massachussetts Institute of Technology, September 1992.

[Tsai87] Roger Tsai. A Versatile camera Calibration Technique. Conference on Vision and Pattern Recognition, 1986.

[IITD96] Rahul Bhotika and T.R. Vishwanath. Experiments in Camera Calibration and Shape From Shading. Mini Project Report, Spring 1996, Deptt. of Computer Science and Engineering, IIT Delhi, 1996.

[XIL] XIL Programmer's Reference Guide. SUN Microsystems' Inc. Pasadena, USA, Feb. 1996.