2-9-2010

Structure and camera calibration from active correspondence of lines and intersection points

Chandra Sekhar Gatla

Chandra Sekhar Gatla
Candidate

Mechanical Engineering
Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Dr. Ronald Lumia , Chairperson

Dr. Gregory P. Starr

Dr. John Russell

Dr. Rafael Fierro

STRUCTURE AND CAMERA CALIBRATION FROM ACTIVE CORRESPONDENCE OF LINES AND INTERSECTION POINTS

BY

CHANDRA SEKHAR GATLA

B.E., Mechanical Engineering, Osmania College of Engineering,
Hyderabad, India, May 2000,
M.S., Mechanical Engineering, University of New Mexico,
Albuquerque, New Mexico, May 2002,

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Engineering

The University of New Mexico
Albuquerque, New Mexico

Dec, 2009

DEDICATION

This dissertation is dedicated to my parents and my family members for their constant support.

ACKNOWLEDGMENTS

I heartily acknowledge and thank Dr. Lumia, who has been my advisor and my committee chair for the past years. I am grateful to him for teaching me about organizing, planning and doing my research effectively. I would like to thank him for the countless suggestions that he gave me. He was always willing to help in every possible way whenever I had some problem.

I wish to acknowledge Dr. Starr for the knowledge I gained from his various courses that I have taken from time to time. I am very thankful to him for being on my committee.

I would like to thank Dr. Smith for his teaching and suggestions that in many ways have helped me understand computer vision. He has taught me about various aspects about computer vision especially about active contour models.

I would like to thank Dr. Rafael Fierro and Dr. John Russell for being on my committee, giving me various suggestions and helping me.

I would like to thank Dr. David Vick who has taught me a great deal about programming, using and controlling the robots which helped me complete my research successfully. He has been a good friend to me and constantly encouraged my work and was always ready to help at all times whenever I needed.

I would like to thank Colin Selleck for his help in calibrating Biclops. I have learned a lot about computer vision, cameras, hardware and many other things from him.

Last but not least I thank Dr. John Wood for providing me the necessary financial support without which this work would not have been possible.

Finally I thank all my family members, teachers and friends, who have constantly encouraged and supported me in all my life.

STRUCTURE AND CAMERA CALIBRATION FROM ACTIVE CORRESPONDENCE OF LINES AND INTERSECTION POINTS

BY

CHANDRA SEKHAR GATLA

ABSTRACT OF DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Engineering

The University of New Mexico
Albuquerque, New Mexico

Dec, 2009

STRUCTURE AND CAMERA CALIBRATION FROM ACTIVE CORRESPONDENCE OF LINES AND INTERSECTION POINTS

By

Chandra Sekhar Gatla

B.E., Mechanical Engineering, May 2000

M. S., Mechanical Engineering, May 2002

Ph.D., Engineering, Dec 2009

ABSTRACT

Many applications, e.g., motion planning, virtual reality, CAD, vehicle navigation, object recognition, photogrammetry, remote sensing, etc., all require a geometrical representation of the three dimensional structure of a scene. In this dissertation we study the problem of determining the 3D structure of a scene given images of it from various views.

We built Biclops a two camera directed vision system. Each camera is mounted on a Pan-Tilt Unit which can be independently controlled. Biclops is built as a tool so that it can be picked up by an industrial robot. This eye in hand system is used to find the structure of the scene. Two robots are mounted on a Robot Transport Unit (RTU) on either end. We use lines in the images and find their intersection points and use them to make the necessary correspondence between the points in different views. Lines and point features and their corresponding entities are used to determine the 3D structure of the scene. The 3D structure is transferred to the other robot space for it to access the objects in the scene.

The accuracy of 3D structure of the scene found is dependent on the accuracy of the various parameters of the underlined equipment. We go a step further to improve the accuracy of the parameters of various equipments used. We developed a new method called ViCKi (virtual close loop kinematic method) to calibrate the industrial robots and the RTU equipment using a laser pointer. We also calibrate the pan tilt units of the Biclops using the laser pointer tool. Various experiments are done to show the accuracy of the calibration methods.

This calibrated equipment is used to find the 3D structure of a scene. Various experiments are done to prove the concept that using lines and intersection points are better than using the traditional corner features. The robot uses this 3D structure to pick up the objects.

We also used the lines and intersection points to find a track painted on the ground. A robotic mobile platform is used and a vision system is built. The vision system takes pictures of the track and finds its 3D location. The RMP plans it trajectory to follow the track without losing the track out of sight.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1. Introduction

1.1. Introduction

Many applications, e.g., motion planning, virtual reality, CAD, vehicle navigation, object recognition, photogrammetry, remote sensing, etc., all require a geometrical representation of the three dimensional structure of a scene. Inference of 3D structure of objects in a scene from its 2D projections is a long studied problem. One of the important methods is to determine the 3D shape of visible objects in a static scene from images acquired by two or more cameras or a single camera at multiple view points. The images obtained from numerous viewpoints are processed for various primitives such as points, lines, curves, planar entities, etc., which are the input to the system. For any of these primitives, the main problem boils down to the correspondence between those primitives in the images. Either human intervention is needed or extensive search techniques which include epipolar constraints, correlation, optical flow, etc., are developed to find these correspondences. All these search techniques are susceptible to image noise. Various algorithms are developed using point correspondences to determine the structure [1], [2], [3]. Points are very difficult features to detect in an image and are more susceptible to noise, unlike lines and curves which are easier to detect and are less susceptible to noise. When a point is detected in an image there is no means to correct or to minimize the error in the detection process. When, if an edge is detected in an image and using a least squares error minimization a curve is fit to the edge thus we have some means to minimize the errors due to the image acquisition. For points we have two degrees of freedom, *i.e.*, its position, which can only be adjusted or corrected using very localized image patch, whereas for an edge the two degrees of freedom are its position

and orientation which can use the information from a larger area of the image making up this line, thus giving us a greater means to reduce the errors. This is the reason that many people have started using lines or curves or combinations of them as primitives [2]-[9].

1.2. CCD Cameras

Most modern cameras are *Charge-coupled-device* (or *CCD)* cameras. A CCD sensor uses a rectangular grid of electron sites laid over a thin silicon wafer to collect the amount of light energy reaching each of them. When a photon strikes the electron site, an electron-hole pair is generated. The electrons are captured by applying a positive electrical potential to the corresponding gate. The electrons generated at each site are collected over a fixed period of time producing a digital image.

1.3. Digital Image

A real image is a continuous 2D picture for processing the image it has to be digitized somehow. CCD and other digital cameras now produce digital images directly. A digital image is a 2D grid representation of the image by using various properties of the particular elements of the grid called the pixels. The images pixels can be composed of any properties ranging from intensity, color red, blue and green values, infrared intensity, x-ray intensity, etc. For our purposes the images are either gray scale (intensity level from black to white) or color (red, green, blue). A digital image of width (W) and height (H) can be represented as

$$I(x,y)\forall x \in [0,W), y \in [0,H) \qquad \textbf{(1.1)}$$

1.4. *Pinhole Camera Model*

A pinhole camera is a device where one side of a box has a pin hole and the opposite side is a translucent plate. This pinhole camera will project an inverted image on the translucent plate, of the light source in front of the box. Since the pinhole camera produces an inverted image it is convenient to consider a virtual image on a plane lying in front of the pinhole, at the same distance from pinhole as the actual image plane as shown in Figure 1.1 where the image is straight.

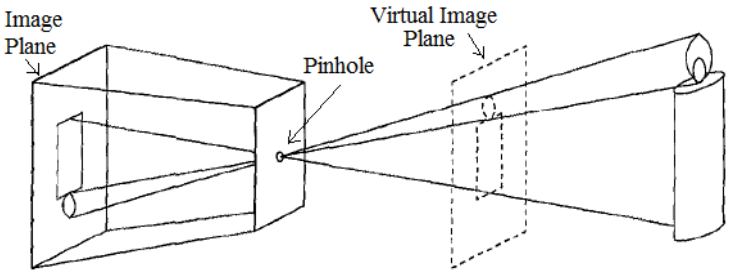

Figure 1.1. A pinhole camera projecting an object onto the image plane.

1.5. *Perspective Projection*

Assuming the pin hole is very small one ray of light would pass through each point in the image plane of the plate, the pinhole, and some scene point. In reality, the pinhole will have a finite size, and each point in the image plane will collect light from a cone of rays with a finite solid angle. Imaging geometry will not strictly apply in pinhole cameras or real cameras.

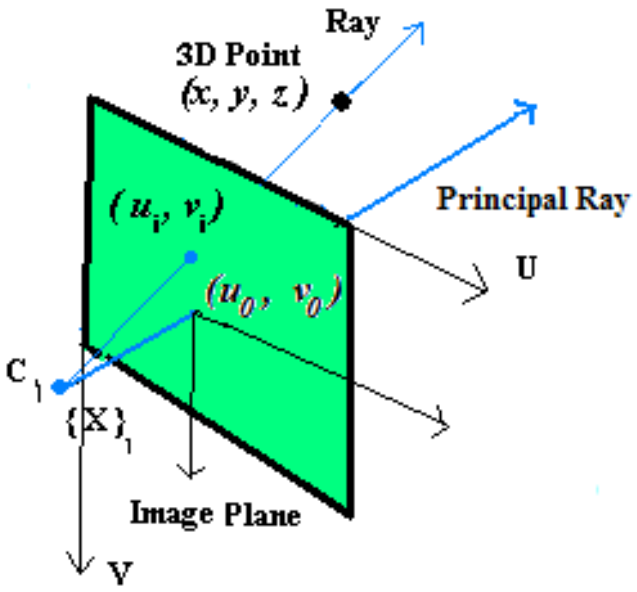

Figure 1.2. Perspective Projection.

In real cameras the projection is also complicated by the lens, aperture, etc. Despite all this the pinhole perspective projection model is mathematically convenient and often provides an acceptable approximation of the imaging process.

Consider a coordinate system in pixels with the origin at the top left corner of the image, and the u-axis along the horizontal rows from left to right and the v-axis along vertical columns from top to bottom as shown in Figure 1.2. Consider a 3D coordinate system fixed at the camera focal point (C), with x and y axis parallel to the u and v axis respectively and z axis coming out of the camera. Let an orthogonal ray passing through the optical center intersect the image plane at a point (u_0, v_0) in pixel units. This point is called the "*Principal Point*". The orthogonal distance between the optical center and the image plane is the "focal length" (f). Since we are measuring the distances in the image as pixels we need to know the scale factors of the two axis u and v. Let these scale factors be s_x and s_y.

Consider a 3D point (x, y, z) projected into the image as a point (u, v). Based on similar triangles we get the two following equations

$$\frac{(u-u_0)}{\left(f/s_x\right)} = \frac{x_c}{z_c} \Leftrightarrow \frac{(u-u_0)}{f_x} = \frac{x_c}{z_c} \qquad \textbf{(1.2)}$$

$$\frac{(v-v_0)}{\left(f/s_y\right)} = \frac{y_c}{z_c} \Leftrightarrow \frac{(v-v_0)}{f_y} = \frac{y_c}{z_c} \qquad \textbf{(1.3)}$$

The parameters 'f_x', 'f_y', 'u_0' and 'v_0' are specific to this camera and do not depend on the position or orientation of the camera. They only depend on the camera lens and focus. These parameters are often referred to as "internal parameters" in contrast to "external

parameters" of a camera which are position and orientation of the camera coordinate system with respect to a global coordinate system.

1.6. Goal

The goals of this dissertation are to:

- **determine the 3D structure of a scene** (a geometric representation of all the objects in a scene) using active automatic correspondence of lines and their intersection points in the images.
- **improve the accuracy of the 3D structure** of scenes obtained by improving the camera calibration parameters, also going further by modeling the underlying equipment used and calibrating them.

1.7. Problem Formulation

Given a set of images of objects in a static scene taken by a calibrated camera (on a stationary or mobile robot) from various viewpoints, the aim is to determine the three dimensional structure of objects, *i.e.*, to determine the 3D location of all the edges and corners of visible objects. By determining the 3D structure of all objects, the scene is reconstructed. For example consider a box in the field of view of the cameras. Images of this box are taken from various view points and the 3D locations of its edges are determined. Another objective is to model the underlying robot camera systems and calibrate their parameters to improve the accuracy. For example consider a hand in eye system. The 3D structure obtained using this system is solely dependent on the accuracy of the robot and the camera calibration parameters. We develop new algorithms to calibrate the robot, the RTU, pan- tilt mechanisms and the cameras to improve the accuracy of the parameters.

1.8. Contribution

1.8.1. Industrial Robot and RTU Calibration

We developed a new calibration procedure to calibrate industrial robots and other ancillary mechanisms. The proposed method uses a laser pointer tool on the robot's end-effector to aim at a fixed location on a distant object. By projecting the laser pointer onto a distant object, the resolution of observations is improved, increasing accuracy of measurements of the joint angles required for accurate calibration of the robot. The calibration procedures to calibrate industrial robots and RTU are described in detail in Chapter 4 and Chapter 5 respectively.

1.8.2. Pan-Tilt cameras Calibration

The robot and RTU calibration procedures improved the accuracy of their parameters allowing us to compute a more accurate position of the PTU origin. This alone is not sufficient for accurate position of the cameras on the PTUs. The PTUs have their own mechanisms which also need to be modeled for accurate position of the cameras with respect to the robots base coordinate system. A new calibration procedure is developed with complete pant-tilt model without any assumptions. The calibration procedure uses a single unknown 3D point in space. The robot motion and the pan-tilt motion are used to acquire various images of this point. The calibration is carried out using the acquired data. The details of the procedure are presented in Chapter 6.

1.8.3. Automatic correspondence

All the methods that use lines segments as primitives either assume

1. correspondence is given or

2. the sequence of images is taken by a camera from viewpoints very close to each other such that the images are nearly similar to perform a region matching.

A new method is proposed which does not make any of the above assumptions and uses the IDI points (indirectly determined intersection points) as the guide to determine the corresponding line segments automatically. No real image points are used so the errors in determining the (intersection) points are dependent on the errors in the lines which are easier to extract and are less susceptible to noise. The correspondence between the IDI points is made using the epipolar constraint.

1.9. Outline of Dissertation

The dissertation is arranged in chapters each describing one of the major aspects of the goal.

1.9.1. Literature Review

Chapter 2 describes in detail about the various technologies and their state of the art, used in this dissertation.

1.9.2. Work-Cell

Chapter 3 describes the work-cell in detail. It consists of two Staubli robots mounted on a rail transport unit.

1.9.3. Robot Calibration

The cameras that we use are mounted on Staubli robots. These robots are repeatable but not very accurate. Since the PTUs are mounted on the robot and accuracy

of their position depends on the accuracy of the robots parameter, so they need to be as accurate as possible. In Chapter 4 a new method called ViCKi is developed to calibrate the robot system. Robots are calibrated using this method and experiments are done to show the improvement in the accuracy of the system.

1.9.4. Robot transport unit (RTU) Calibration

The two Staubli RX-130 robots are mounted on a Robot transport unit (RTU) such that each one of them can be independently controlled to move along the track. Since the cameras are mounted on a robot which is mounted on the RTU, the accuracy of the RTU is important. The accuracy of RTU is also important in transferring the 3D information (objects 3D points, line positions etc) from one robot to the other. In Chapter 5 RTU calibration methods are presented. Experiments show the improvement in the accuracy.

1.9.5. Pan-Tilt Camera Calibration

The camera system we used to acquire images of the workspace is called Biclops. Biclops is a custom-made, dual-camera motorized vision system. It consists of two FireWire color cameras, each attached to a pan-tilt unit (PTU). The PTUs are attached to a bracket, which is connected to a tool base. Biclops has four degrees of freedom, *i.e.*, one pan axis and one tilt axis for each camera. The PTUs can be programmed to move the cameras to aim at any desired location in the workspace. Chapter 6 presents a new method to calibrate Biclops. Experiments show the enhanced accuracy.

1.9.6. Indirectly Determined Intersection (IDI) Points

Chapter 7 describes various image processing routines and algorithms to find the image features. It describes IDI points and their extraction from images. Various experiments, which prove IDI point extraction is less error prone than other image points, are presented.

1.9.7. 3D Structure Using IDI Points

Chapter 8 describes in detail the underlined theory of 3D structure from IDI points. Experiments to compute the 3D structure and calibration are also presented. The theory is applied to different applications involving either the Staubli robots or the Segway robot.

Chapter 2. Previous work

2.1. Three-dimensional Structure

2.1.1. 3D Structure using Points

Determination of three dimensional structure (3D position and orientation of all the objects in the scene) from motion and motion parameters is a long studied problem. One of the many approaches is to use point correspondences within various images obtained from a single camera or multiple cameras. Faugeras and Mourrain [1], [2] studied the geometric and algebraic relations and constraints between the corresponding points in number of images. These relations are of three types bilinear, trilinear and quadrilinear arising when we consider two, three and four images, respectively, among the N images. The bilinear relations are the well-known epipolar constraints. They also show that two trilinear relations imply the bilinear ones, the quadrilinear relations are in the ideal generated by the bilinearities and trilinearities and do not bring in new information. Faugeras and Mourrain [2] showed how the perspective projection equation can be suitably generalized and that in the case of three images there exist two independent trilinear relations between the coordinates of the images of a 3D line. For projective reconstruction from images, there exist a minimum number of entities that are required to compute a solution to the structure without ambiguities. For three views and a projective reconstruction Oskarsson, Zisserman and Astrom [3] derived the minimal number of entities that are required for a combination of points and lines. For three images, the minimal cases for combinations of points and lines are: “6 points,” “4 points and 3 lines,” “2 points and 6 lines” and “9 lines.” Thus knowing the minimum number of entities the 3D structure can be determined without any ambiguities.

2.1.2. 3D Structure using Lines

Making point correspondence is a difficult process and usually involves human intervention or high level input. Also the process suffers from image noise. Many people have started to use the correspondence of lines and curves to determine the structure. Since a line has 2D of freedom when a correspondence between two lines is made we are only concerned with the match of the direction (thus the actual position of individual points on these lines need not correspond, giving them an extra degree of freedom to move along the line). Bartoli and Sturm [4] addressed the problem of camera motion and structure reconstruction from line correspondences across multiple views, from initialization to final bundle adjustment. Based on Plucker coordinates [4] to represent the lines, they proposed a maximum likelihood algorithm, relying on linearizing the Plucker constraint and on a Plucker correction procedure to compute the closest Plucker coordinates to a given 6-vector. However the correspondence of the lines in three images is assumed to be given. Hartley [5] developed a practical and rapid algorithm for projective reconstruction of a scene consisting of a set of lines seen in three or more images with uncalibrated cameras.

Quan and Kanade [7] investigated the properties of projection of lines by affine cameras (hypothetical cameras with affine transformations) and proposed a linear algorithm for affine structure from line correspondences. The affine structure is a good approximation to the real one when the depth of objects is small. They introduced a one-dimensional projective camera which converted the problem of "3D affine reconstruction of line directions" into a "2D projective reconstruction of points." They also proposed a line based factorization method to handle redundant views. Quan and Kanade [8]

extended this factorization method to lines and developed a multi-step factorization method. Instead of one step factorization for points, a multi-step factorization method is developed for lines based on the decomposition of the shape and motion into substructures. Each of these substructures is then linearly solved by factorizing the appropriate measurement matrices.

Taylor and Kriegman [9] developed a new method to determine the three dimensional structure of a scene composed of straight line segments using the image data obtained from a moving camera. The algorithm is formulated in terms of an objective function which measures the total squared distance in the image plane between the observed and projected edge segments. The objective function is then minimized with respective the line parameters and the camera position. Berthilsson and Åström [10] proposed an algorithm for reconstructing a general 3D curve from a number of 2D images taken by uncalibrated cameras. This algorithm is based on aligning the subspaces by using orthogonal projections and maximizing some of the largest Eigenvalues of the sum of these projections. No point correspondences except the end points are assumed.

2.1.3. 3D Structure using Curves

Kahl and Heyden [11] showed how to use corresponding conics to compute the fundamental/essential matrix and to reconstruct the scene. Kaminski, Michael Fryers [12] showed how one can compute, without any knowledge on the camera, the homography induced by a single planar curve. They derived the extended Kruppa's equations that describe the epipolar constraint of two projections of a general algebraic curve. They also established the minimal number of algebraic curves required for a solution of the epipolar geometry as a function of the degree and genus. Papadopoulo and Faugeras [13]

discussed the problem of determining the structure and motion of rigid curve. They used long monocular sequences of images of the curve and computed the derivatives that are defined on the spatio-temporal surface generated by the curve. For general 3D rigid curves, there is exactly one constraint for each image point that relates these derivatives to the kinematic screw and its first order time derivative.

2.1.4. Analysis of 3D Structure Methods

Determining structure using line and curve correspondence has eased some of the problems with point correspondences. In detecting lines, the image noise is minimized in one degree of freedom, thus the equations of 2D lines do not suffer as much as points from image noise. Though the lines are easier to detect, the correspondence problem still remains. Most of the time human intervention is needed. Though human intervention is required less frequently for line correspondences than points, it is still undesirable. Some automatic means to make the line correspondences is needed.

2.2. Calibration of Cameras

Various applications like photogrammetry, remote sensing, motion planning, virtual reality, CAD, vehicle navigation, object recognition, etc., all require calibrated cameras. Thus, determining the camera calibration parameters is necessary. The calibration parameters can be determined using the point, line or curve correspondences in various images of objects in known locations. Luong and Faugeras [14] analyzed in detail the geometry of a pair of cameras and introduced the fundamental matrix which has all the relevant information to establish correspondences between features in two images. The fundamental matrix is the result of the epipolar constraints. Knowing a minimum of corresponding features this fundamental matrix can be computed numerically. Faugeras,

Luong, and Maybank [15] developed a method to calibrate cameras using just point correspondences in sequences of images without knowing the motion of the cameras. Luong and Faugeras [16] addressed the problem of determining the motion of the cameras and structure, using an uncalibrated moving camera. They showed that point correspondences between three images and the fundamental matrices computed from these point correspondences are sufficient to recover the internal parameters of the camera (calibration). They showed those point correspondences are sufficient to recover the motion parameters and to compute the perspective projection matrices which enable to reconstruct the 3D structure up to similarity.

2.3. Calibration of Pan-Tilt Cameras

Machine vision camera systems need quick, simple, easy and repeatable and accurate calibration methods. Many approaches to camera calibration exist. Some of these methods use a set of calibration points with known world coordinates. These world points can be obtained by either using a calibration object [17] in a known location or points marked in the workspace whose locations are measured. For example, a planar object with feature points clearly marked in a grid can be placed at a known location and moved by a known motion. Given this set of feature points with known world coordinates (X_i, Y_i, Z_i) and their projected locations in an image plane (u_i, v_i), the external and internal parameters are found which will best map the world points to their image points by determining the parameters which minimize the mean square distance between the observed and computed positions of a feature on the image plane.

Other methods use geometric properties to calibrate cameras. These methods calibrate some of the internal camera parameters using invariant characteristics of

geometric objects and their images. These methods do not require the position of the object relative to the camera. The aspect ratio is found using the image of a sphere [18]. Spheres are also used to locate the principal point [19]. The vanishing points [20] of parallel lines drawn on the faces of a cube are used to compute the principal point and focal length.

Some methods use only feature coordinates in the image plane to calibrate. These methods are called self calibration methods [21], [22], [23], [24] because they do not require known calibration points. It requires camera motion to take multiple images. Faugeras *et al*. [21] developed a method where a motion sequence of a single camera moving in an unconstrained manner can be used to calculate the internal camera parameters. This method does not require known world coordinates of the calibration points. It requires only feature correspondences from a set of images where the camera has undergone pure rotation. In this method, the internal parameters of the camera are determined including radial lens distortion.

Calibrating a camera mounted on a pan-tilt mechanism involves the added complexity of finding the location of the pan and tilt axes of rotation. Most of the existing methods of calibrating pan-tilt cameras have assumed PTUs with orthogonal axes, or have assumed a relatively simple geometric model of motion, in which the axes of rotation are orthogonal and aligned with the camera imaging optics [25], [26], [27]. While this simplification works well over small volumes, accuracy suffers in a larger workspace, *i.e*., the camera model does not predict well the projection of a 3D feature point. In [28] a more complete model of pan-tilt cameras was employed, making the calibration suitable for use with low cost pan-tilt mechanisms. This method uses an

existing tracking system consisting of stationary calibrated cameras. An LED point feature in the workspace is tracked using the fixed cameras to build a virtual calibration object that is then used to calibrate the pan-tilt cameras.

Most of the existing methods require either a known set of calibrated world points or a large number of corresponding image features. Either of these requirements makes the calibration process complicated and slow because the data gathering process must be supervised for correctness. Even a few correspondence errors will reduce considerably the accuracy of the resulting model. This chapter describes a simple method to calibrate cameras and their PTUs using a single unknown stationary world point as the calibration point. Note that the 3D location of the point is unknown at the start of the process, and is determined during the calibration procedure. The data acquisition process is very much simplified by using a single unknown stationary 3D point (with reasonable initial guess); the extra effort to acquire multiple points is eliminated. Furthermore, with a single point to detect, a faulty correspondence of a feature between two cameras can never occur. Consequently, it is possible to use a very general model for pan-tilt camera motion and minimize human effort, since feature correspondence is unnecessary, in the calibration process.

2.4. Calibration of Industrial Robots

There has been considerable research in the field of robotic calibration. A brief review is presented in [29]-[33]. Existing techniques can be classified into open-loop and closed-loop approaches. Open-loop methods involve measuring the end-effector pose which requires special equipment (such as theodolites, inclinometers, ball-bar, and coordinate measuring machines [34]). The process of obtaining these measurements is

time consuming and must be repeated for high precision systems. The resolution of measurements near the end-effector is limited by the equipment used.

Closed-loop methods [35]-[41], on the other hand, use the joint angle measurements already in the robot, and thereby can be considered self-calibrating. These methods impose some constraints on the end-effector and the joint readings alone are used to calibrate the robot using kinematic closed-loop equations. Some researchers in the past have used linear constraints on the end-effector positions allowing the end-effector to slide along a line, e.g., Newman *et al.* [35] used a laser line. Ikits *et al.* [36] and Zhuang *et al.* [37] imposed plane constraints on the end-effector positions. Using a plane constraint is problematic because it is difficult to be certain that the end effector is exactly on the surface; neither above it nor indenting it.

Bennet *et al.* [38] considered manipulators as mobile closed kinematic chains. It is difficult to move a physically closed kinematic chain from one position to other while maintaining the physical constraints. Hence it is difficult to gather accurate joint readings. Meggiolaro *et al.* [39] used a single endpoint contact constraint, equivalent to a ball joint, to calibrate the robot. The robot moves to different configurations that satisfy the contact constraint. This method needs a physical contact point, and suffers from the same problems as the plane constraint methods.

Gatla et al. developed a new method called “virtual closed kinematic chain” (ViCKi) [66]. Unlike previous closed-loop methods, this approach does not require any physical constraints. A laser tool is attached to the end-effector. This laser tool aims at an arbitrary but fixed point on a distant object, thus creating a virtual closed kinematic chain. This procedure can be used to collect the joint readings for various positions of the end

effector that aim at that fixed location. The procedure capitalizes on the constraint that the laser line must pass through the fixed point for all robot joint configurations. With this redundant joint information, calibration is accomplished. The main advantage of this method is that the distant laser point is very sensitive to the joint values, *i.e.*, it magnifies the error (a very fine adjustment in the joint angle configuration is needed to aim at a particular point), which facilitates acquiring more accurate joint values for the calibration.

2.5. RTU Calibration

Gatla *et al*. [66], developed a calibration approach called "virtual closed kinematic chain" (ViCKi) to calibrate the industrial robots which is later applied to calibrate the RTU calibration. Unlike previous closed-loop methods, this approach does not require any physical constraints. We use this method to independently calibrate two industrial robots on a RTU. The method is modified to determine the transformation of one robot with respect to the other by calibrating the RTU parameters.

Chapter 3. Work-cell

The workcell consists of two Staubli RX-130 robots mounted on a robot transport unit (RTU) that moves the robots along a track. The various tools used include a Barrett Hand, Probe, Laser Pointer and Biclops (dual-camera directed vision system). The workcell also include a Robotic Mobile Platform (RMP) manufactured by Segway.

3.1. Staubli Robot

The Staubli RX-130 robot [65] as shown in Figure 3.1 has six rotary degrees of freedom. The shoulder, arm, elbow, and forearm, are controlled by axes one, two, three and four, respectively. The wrist is controlled by joint axis five and tool flange by joint axis six. The rotations of the forearm and wrist together give spherical degrees (roll, pitch and yaw) of freedom to the tool. This

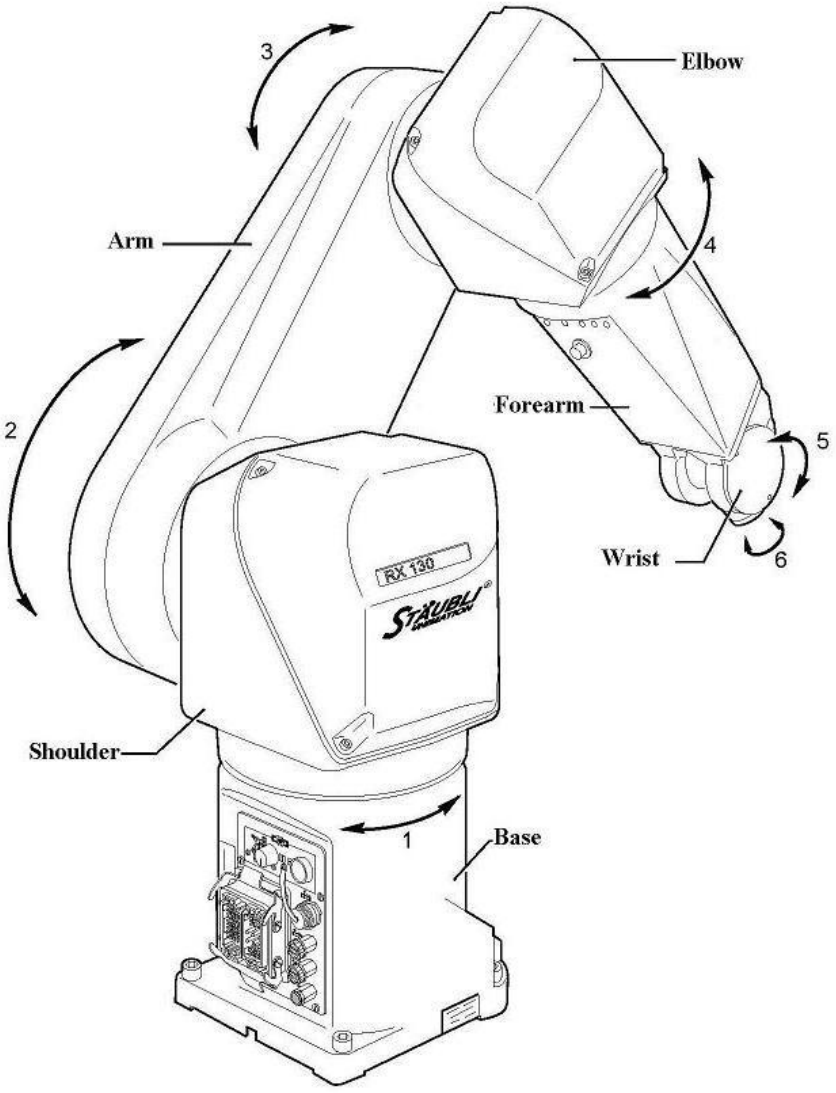

Figure 3.1. Staubli RX-130.

configuration sets axis 1 perpendicular to axis 2. Axes 2 and 3 are parallel. Axis 4 is perpendicular to 3 and 5. The last three joint axes intersect at a common point. This robot is similar kinematically to the well known Unimation Puma 560.

3.1.1. Staubli Model

A model of the robot shown in Figure 3.1 is built with coordinate system definitions according to Craig's modified Denavit-Hartenberg (DH) [60], [61] and Hayati (HR) [62] conventions combined. The coordinate systems are shown in Figure 3.2. We adopt the use of the following notation: $C\theta_{12}$ for $\cos(\theta_1+\theta_2)$, $S\theta_{12}$ for $\sin(\theta_1+\theta_2)$ and ${}^{A}T_{B}$ for transformation matrix which transforms points described in frame B to points in frame A.

The transformation matrix from frame 'i-1' to frame 'i' with DH parameters (α_i, a_i θ_i and d_i) is given by $T=R_x(\alpha_i)T_x(a_i)R_z(\theta_i)T_z(d_i)$ computed by

$$ {}^{i-1}T_i = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & a_i \\ C\alpha_i S\theta_i & C\alpha_i C\theta_i & -S\alpha_i & -d_i S\alpha_i \\ S\alpha_i S\theta_i & S\alpha_i C\theta_i & C\alpha_i & d_i C\alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.1)} $$

where α_i is the angle between Z_{i-1} and Z_i measured about X_i and a_i is the distance between Z_{i-1} and Z_i measured along X_i, θ_i is the angle between X_{i-1} and X_i measured about Z_i and d_i is the distance between X_{i-1} and X_i measured along Z_i.

The transformation matrix from frame 'i-1' to frame 'i' with Hayati parameters (α_i, a_i θ_i and β_i) is given by $T=R_x(\alpha_i)T_x(a_i)R_y(\beta_i)R_z(\theta_i)$ computed by

$$ {}^{i-1}T_i = \begin{bmatrix} C\beta_i C\theta_i & -C\beta_i S\theta_i & S\beta_i & a_i \\ S\alpha_i S\beta_i C\theta_i + C\alpha_i S\theta_i & -S\alpha_i S\beta_i S\theta_i + C\alpha_i C\theta_i & -S\alpha_i C\beta_i & 0 \\ -C\alpha_i S\beta_i C\theta_i + S\alpha_i S\theta & C\alpha_i S\beta_i S\theta_i + S\alpha_i C\theta & C\alpha_i C\beta_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.2)} $$

where α_i is the angle between Z_{i-1} and Z_i measured about X_i and a_i is the distance between Z_{i-1} and Z_i measured along X_i, β_i is the angle of rotation about intermediate Y_i axis and θ_i is the angle of rotation about Z_i.

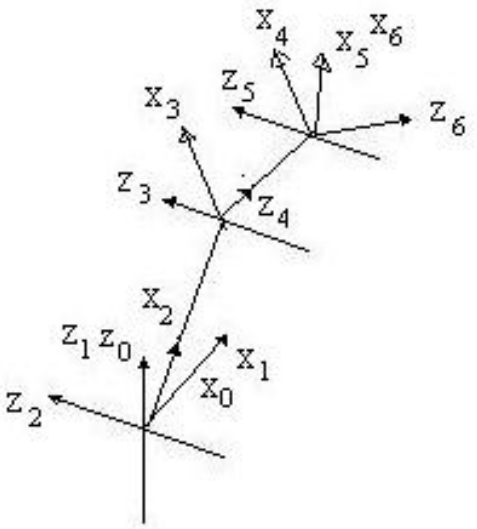

Figure 3.2. Staubli Coordinate Frames.

Table 3.1. Staubli DH Parameters

i	α_i rad	a_i mm	θ_i rad	d_i mm	β_i rad
1	x	x	0*	x	x
2	$-\frac{\pi}{2}$	0	0	0	x
3	0	625	$-\pi$	x	0
4	$\frac{\pi}{2}$	0	0	625	x
5	$\frac{\pi}{2}$	0	0	0	x
6	$-\frac{\pi}{2}$	0	π*	110*	x

*- FIXED AT SPECIFIED VALUE
X- PARAMETER NOT USED

It is well known that the DH parameters have a singularity when neighboring joint axes are parallel. When this singularity exists, the HR transformation is used. Therefore, each transformation matrix is either DH or HR with four parameters, where the fourth parameter is either ‘d’ or ‘β’. For the Staubli robot joints 2 and 3 are parallel. Consequently we use HR for this transformation and DH for all other transformations. Since the base coordinate frame ‘0’ is arbitrary it is chosen to coincide with the frame ‘1’ when the reading of joint 1 is zero. This reduces the transformations from frame ‘0’ to ‘1’ to just rotation about Z_0 by reading of joint 1 angle. Hence the joint offset is fixed at zero

indicated by '*' in the Table 3.1, the other three DH parameters for this transformations are not used and denoted as 'X' in Table 3.1.

The coordinate system of the end-effector (frame 6) is also arbitrary. It is chosen such that the x-axis of this frame coincides with x-axis of frame 5 when joint 6 read 180º. Also, for the Staubli RX-130, d_6 is set to a fixed value of 110mm.

Table 3.1 lists the DH/HR parameters of the complete robot model, where four parameters for each transformation are required. Either 'd' or 'β' is used for the fourth parameter depending on whether the transformation is DH or HR. The parameter that does not apply in each row is marked as 'X'. The 'θ_i' listed in the table are the joint offsets. These offsets are added to the actual readings of the joints to compute the transformation matrices. Hence the total number of independent calibration parameters required to characterize the robot alone is 24 – 4 (base) - 2(end effector) =18 parameters.

3.2. Robot Transport Unit (RTU)

The two Staubli RX-130 robots are mounted on a robot transport unit (RTU) on each end. The robots can be independently controlled to move along the track as shown in Figure 3.3. Each of the robots when at their home positions are at their extreme end of the RTU which correspond to the RTU position '0'. Each of the robots has an independent RTU home coordinate frame that is attached to the RTU.

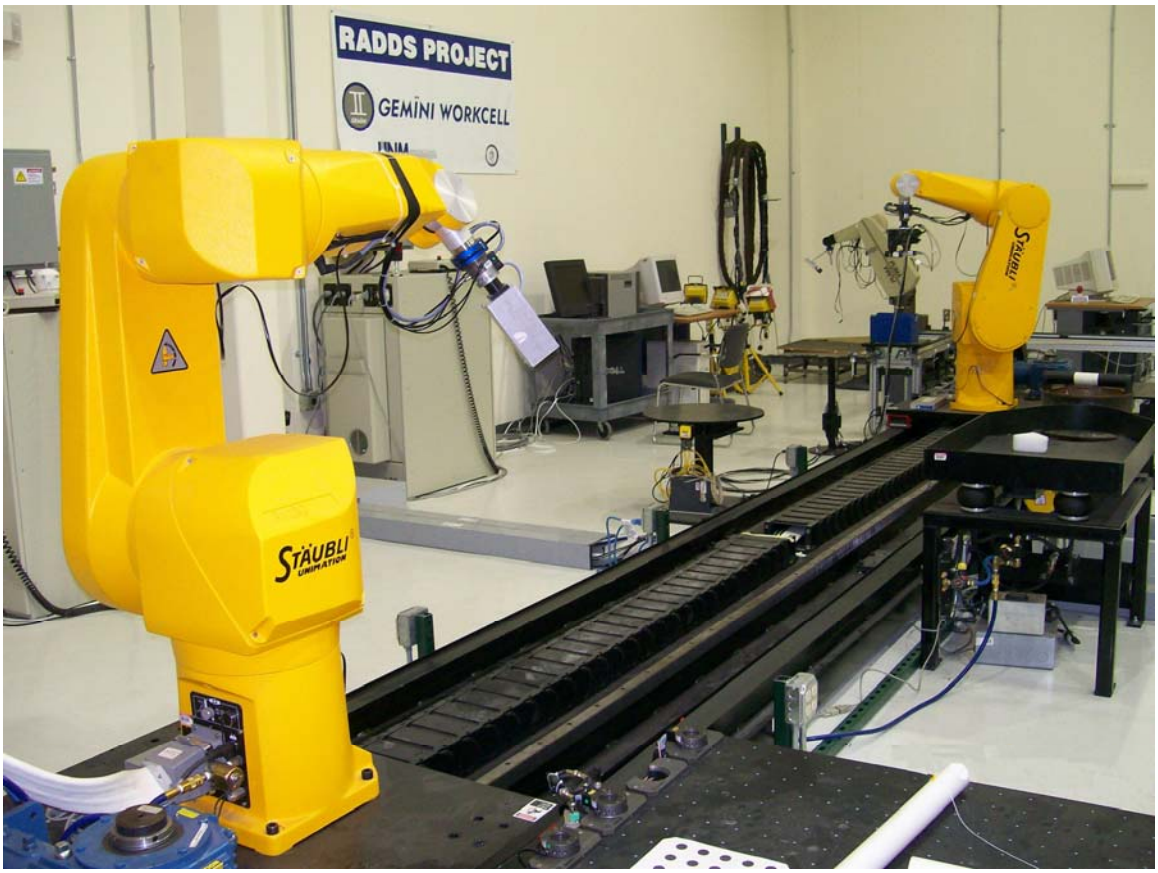

Figure 3.3. Two Staubli Rx130 robots mounted on RTU.

3.2.1. RTU Model

The motion of the robot on the RTU is linear which is modeled as a straight line motion. The matrix transforming the coordinate systems from the Robot base coordinate system to the RTU coordinate system is given by

$$ {}^{R}T_{B} = \begin{bmatrix} C\Phi_l C\Phi_m & S\Phi_l C\Phi_m & -S\Phi_m & D \\ -S\Phi_l & C\Phi_l & 0 & 0 \\ C\Phi_l S\Phi_m & S\Phi_l S\Phi_m & C\Phi_m & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.3)} $$

where B is the base coordinate system of the robot and *R* is the base RTU coordinate system for this particular robot. RTU base coordinate system for each robot is chosen such that the *x*-axis is parallel to the RTU direction and the origin coincides with the origin of robot's base coordinate system when the robot's RTU position is zero as shown

in Figure 3.4. Φ_l and Φ_m are the angles of rotation about z and then y to align the robot base x-axis along the RTU direction and D is the RTU position of the robot. We use the notation ${}^{R1}T_{B1}$ for transformation matrix of robot1 with respect to its base RTU coordinate system and ${}^{R2}T_{B2}$ for transformation matrix of robot2 with respect to its base RTU coordinate system. For each robot by knowing the parameters of the RTU direction in the base coordinate system of the robot we can compute the transformations from one robot RTU position to another RTU position.

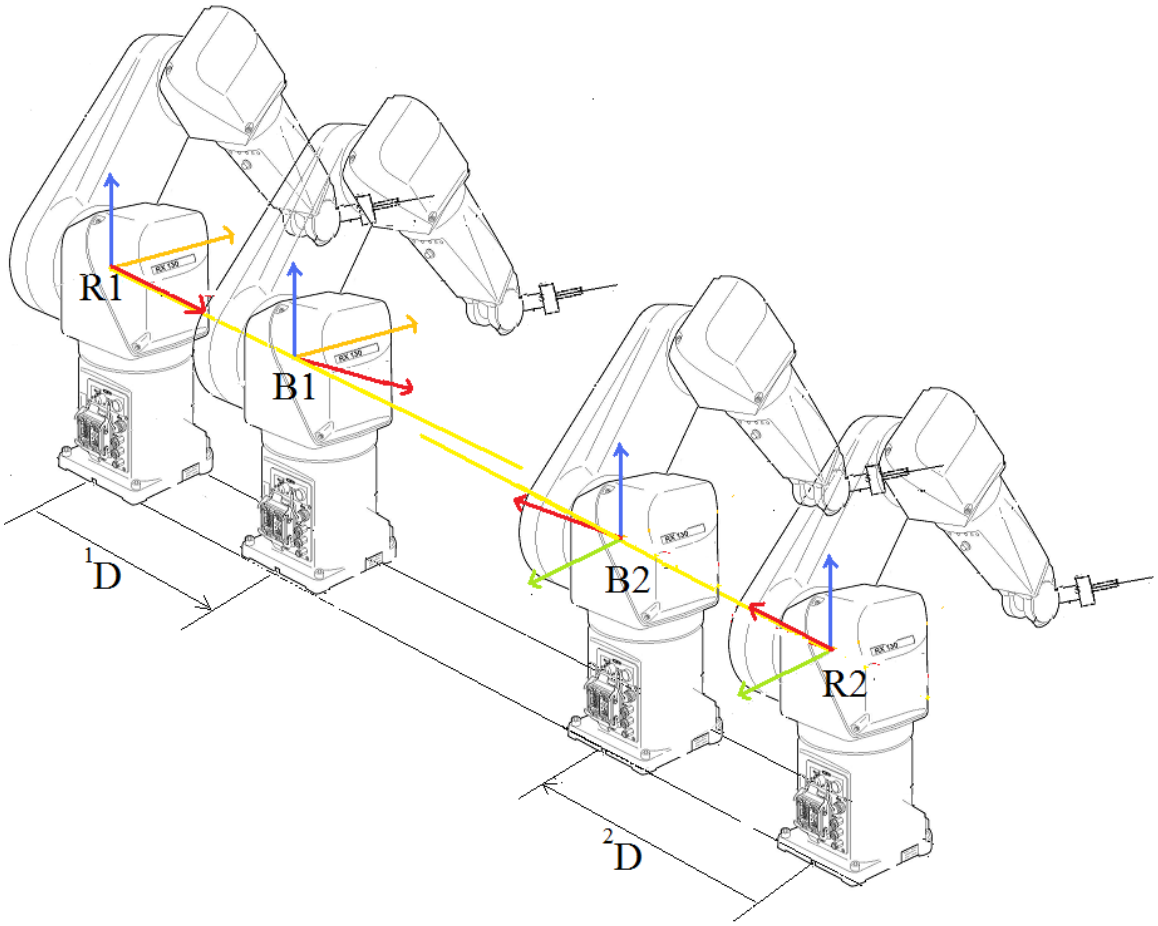

Figure 3.4. Coordinate systems for the RTU and robots.

3.2.2. Robots' RTU Base Transformation

Since each robot is on either side of the robot and we choose independent coordinate systems for the base RTU locations (robot's RTU position = 0) we still need a transformation ${}^{R1}T_{R2}$ between them. Using this transformation matrix, the transformation matrix from one robot base coordinate system at any RTU position to other robot's base coordinate system is given by

$$ {}^{B1}T_{B2} = \left({}^{R1}T_{B1} \right)^{-1} {}^{R1}T_{R2} \, {}^{R2}T_{B2} \qquad \textbf{(3.4)} $$

The transformation in above equation ${}^{R1}T_{R2}$ is the transformation which needs to be modeled. There are four independent parameters to transform these two base RTU coordinate systems from one other considering the constraints that the RTU direction is same in both systems, *i.e.*, their *x*-axes are parallel. We chose the following sequence of transformations to transform from one coordinate system to other. T= Tr (Ψ_x, Ψ_y, Ψ_z)Rx(Ψ_α)Rz(π) computed by

$$ {}^{R1}T_{R2} = \begin{bmatrix} -1 & 0 & 0 & \Psi_x \\ 0 & -C\Psi_\alpha & -S\Psi_\alpha & \Psi_y \\ 0 & -S\Psi_\alpha & C\Psi_\alpha & \Psi_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \textbf{(3.5)} $$

where Ψ_x, Ψ_y, Ψ_z are the translation parameters Ψ_α is the twist angle about RTU direction. These are the parameters of the RTU robots system to be determined. There is a rotation of 180° since the two *x*-axes are parallel but they are in opposite directions due to their definitions.

3.3. Laser Pointer Tool

A laser pointer tool consists of a laser pointer (with pivot to adjust orientation) connected to a tool base. The tool can be picked up by the robot by connecting to the toolbase. The laser tool's pivot is adjusted to align its orientation approximately with the z-axis of the end-effector and it is locked. The robot picks up this tool by connecting to the tool base, as shown in Figure 3.5.

Figure 3.5. Staubli RX-130 robot carrying a laser pointer tool.

3.3.1. Laser Tool Rough Alignment

The laser tool consists of a laser pointer (with pivot to adjust orientation) connected to a tool base. The laser tool's pivot is adjusted to align its orientation approximately with the z-axis of the end-effector and it is fixed rigidly. To increase the accuracy of the laser alignment it is aimed at a constant location and the robot end effector is moved along the z-axis. The change in the laser spot position is noticed and the laser is adjusted to minimize this error as shown in the Figure 3.6. Rotating the laser tool about the z-axis makes a circle. The smaller the radius of this circle the better the tool is aligned. Thus moving the laser forward and backward and rotating it about the z-axis the pivot is adjusted to reduce, but not eliminate, the alignment error.

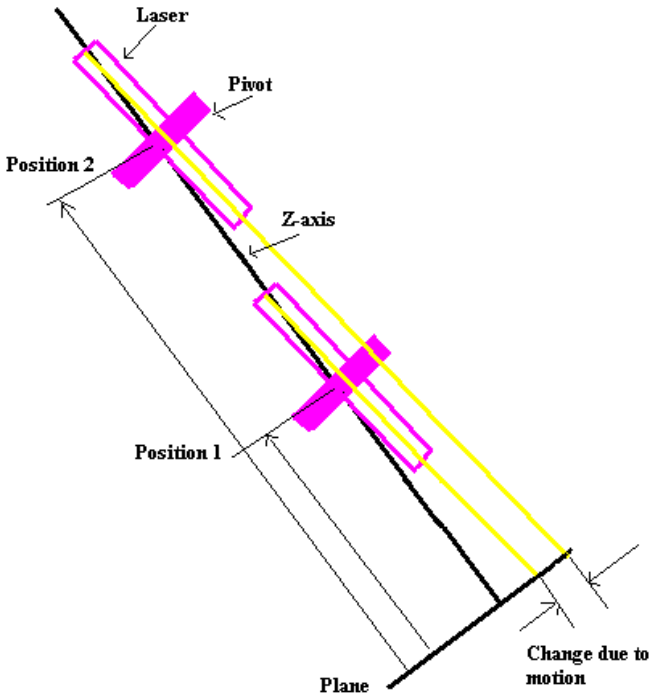

Figure 3.6. Laser Rough Alignment.

3.3.2. Aiming Laser Tool

The robot can aim the laser tool at some location. There are infinite configurations to aim the laser at a desired location, but a convenient location (either present location or any location reachable) for the robot end effector is chosen to aim the laser. Given the position of the robot end effector and the location of the 3D point to aim at, a simple

trigonometric calculation as shown below gives us the required orientation for the laser tool. This approach is described in the following equations.

$$\begin{Bmatrix} d_x \\ d_y \\ d_z \end{Bmatrix} = \begin{Bmatrix} E_x \\ E_y \\ E_z \end{Bmatrix} - \begin{Bmatrix} P_x \\ P_y \\ P_z \end{Bmatrix} \quad \textbf{(3.6)}$$

$$yaw = a\tan 2\left(d_y, d_x\right) \quad \textbf{(3.7)}$$

$$pitch = a\tan 2\left(\sqrt{d_x^2 + d_y^2}, d_z\right) \quad \textbf{(3.8)}$$

where (d_x, d_y, d_z) is the direction of laser pointer that is required to aim the laser. It is the difference between the target point (P_x, P_y, P_z) and the location of the end effector (E_x, E_y, E_z). "atan2" gives the arc tangent angle in correct quadrant. "yaw" and "pitch" are the orientation angles needed to aim the laser at the desired point. Thus knowing the position E and the orientation of the laser (d) the laser can be aimed at the desired location.

3.3.3. Laser Tool Model

Once the laser tool is attached to the robot, the robot can then aim the laser tool at a desired location. The aimed location is accurate only if the whole system, *i.e.,* robot and laser tool models, have accurate parameters. The laser tool is not perfectly aligned with the z-axis of the end effector (Z_6) and the misalignment needs to be modeled by a transformation matrix. A coordinate system ($O_7X_7Y_7Z_7$) is chosen for the laser tool such that the *z*-axis coincides with the laser line. Both the orientation of *x*-*y* axes and the origin along the laser line are arbitrary. Four independent parameters are required to describe a line (laser) in 3D space with respect to the end effector coordinate system. Since the laser

z-axis (Z_7) is closely aligned with the robot end effector z-axis (Z_6) we choose the four required parameters as, two rotations about x and y (These angles of rotations are close to zero.) and two coordinates of translation in x-y plane (also close to zero). We choose the x-y axis and the origin of laser coordinate system to coincide with the previous coordinate system when the four parameters are zeros. Hence the initial guess of all four parameters is zero. The transformation matrix is given by $T= R_x(\theta_x)R_y(\theta_y)T_x(p_x)T_y(p_y)$ computed by

$$ {}^{6}T_{7}=\begin{bmatrix} C\theta_y & 0 & S\theta_y & p_xC\theta_y \\ S\theta_xS\theta_y & C\theta_x & -S\theta_xC\theta_y & p_xS\theta_xS\theta_y+p_yC\theta_x \\ -C\theta_xS\theta_y & S\theta_x & C\theta_xC\theta_y & -p_xC\theta_xS\theta_y+p_yS\theta_x \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.9)} $$

When a laser tool is picked up by a robot then the total number of parameters for the model of the system (robot with the laser tool) are 18(robot) +4(laser) =22.

3.4. Biclops Tool

A directed vision system tool called Biclops consists of two cameras each of which is mounted on a pan and tilt unit (PTU). The cameras used are A602fc color fire-wire (IEEE1394) cameras manufactured by Basler. Each of these cameras is attached to a pan-tilt unit (PTU) from Directed Perception. The PTU camera units are attached to a bracket on either side and a tool base is connected to the middle of this bracket as shown in Figure 3.7.

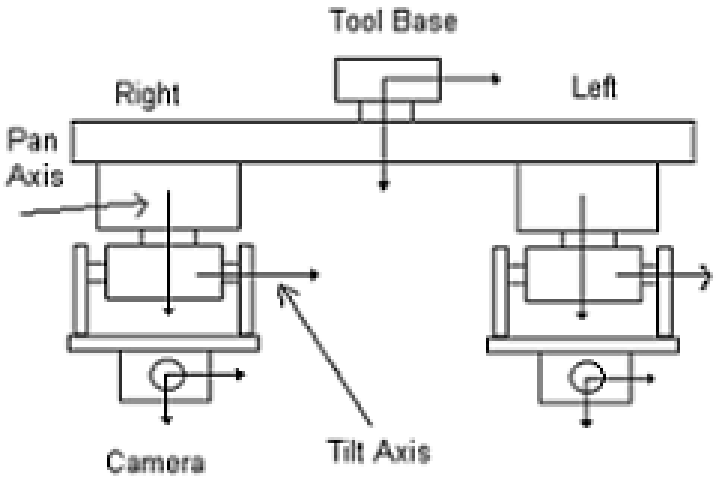

Figure 3.7. Biclops with Pan and Tilt Axis shown.

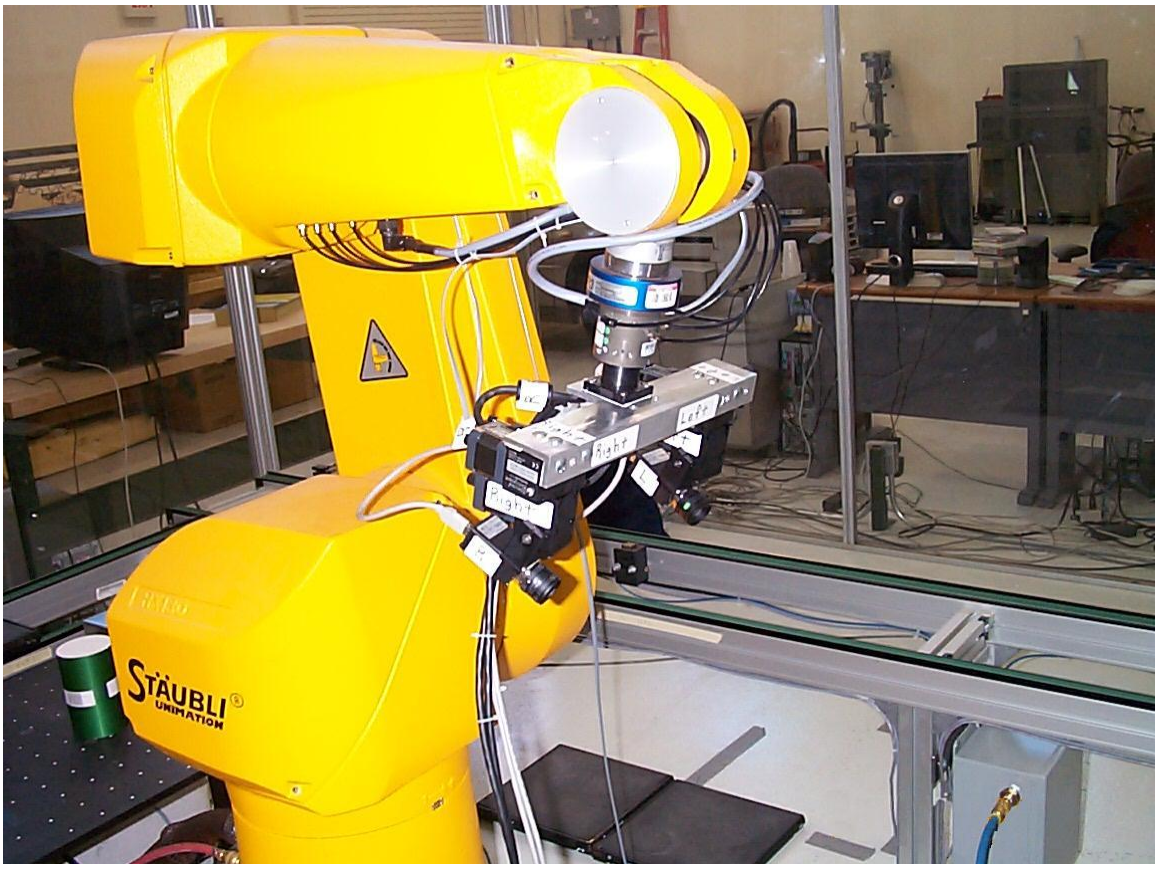

Figure 3.8. Staubli RX-130 robot carrying Biclops.

The robot picks up Biclops by connecting to the tool base as shown in Figure 3.8. Each of the PTU camera units have two (pan and tilt) degrees of freedom (DOF). The cameras are connected to a PC using IEEE1394 (firewire) cables and the PTUs are connected to the PC using serial cables. The various features of Basler A602fc cameras that can be controlled from the PC include area of interest (AOI), brightness, gain, exposure time and color format. Each PTU can be commanded by the PC independently to aim its camera at any desired location in the workspace.

Biclops is built using standard PTUs and a metal frame. Though the dimensions of all the pieces are "known" through design, their accuracy cannot be assumed. Hence we need to model Biclops to determine the dimensions empirically. The Biclops is

modeled with DH parameters and the various coordinate systems in the Biclops system are described below.

3.4.1. Biclops Model

The coordinate system at which the robot holds Biclops is the robot's tool control frame (TCF) which is indicated as frame E in Figure 3.9. We have defined D-H parameters for the two joints of each PTU. A coordinate frame is chosen on the CCD of the camera at the top left corner with the z axis out of the camera towards the objects, the x axis along the horizontal right direction, and y axis along the vertical down direction as shown in Figure 3.10. All the coordinate frames are shown in Figure 3.9. Figure 3.10 shows the image plane coordinate systems with principal point (image center) (u_0, v_0), focal length (f), and the projection of a 3D point p(x, y, z) to (u, v) in camera coordinates. Although pan and tilt axes are shown

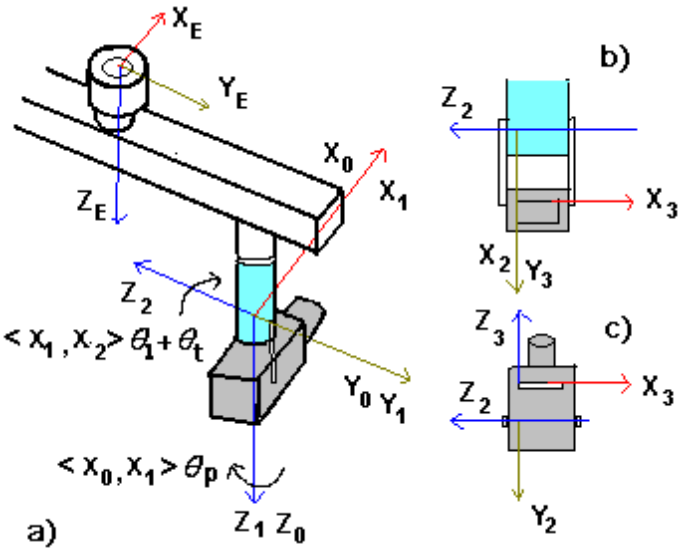

Figure 3.9. Biclops coordinate systems.

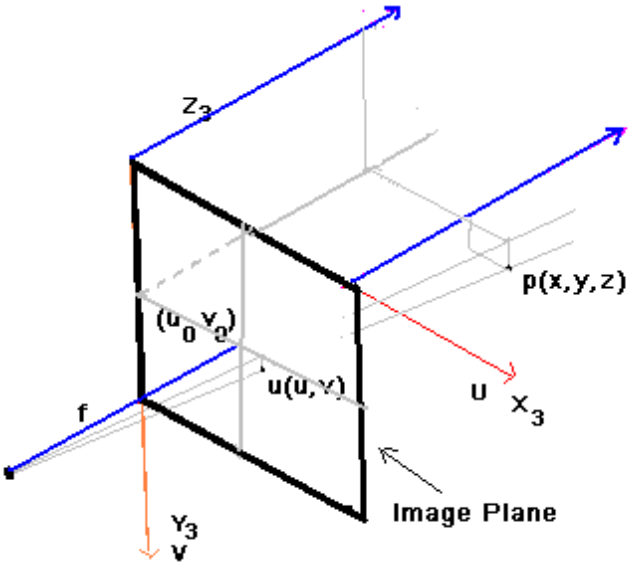

Figure 3.10. Image plane coordinate systems.

intersecting and orthogonal in the Figure 3.9 for clarity and simplicity, the model does not assume this. The separation and the angle between the pan and tilt axes are included in the D-H parameters that will be determined through the calibration procedure.

The transformation matrices used to convert points from one coordinate frame to the next are given by the following equations. We use notation $C\theta_{12}$ for $\cos(\theta_1+\theta_2)$, $S\theta_{12}$ for $\sin(\theta_1+\theta_2)$ and ${}^{A}T_{B}$ for transformation matrix which transforms points described in frame B to points in frame A.

The transformation matrix from the base of the robot to the TCF frame is given by

$$ {}^{B}T_{E} = \begin{bmatrix} C\alpha C\beta C\gamma - S\alpha S\gamma & -C\alpha C\beta S\gamma - S\alpha C\gamma & C\alpha S\beta & x \\ S\alpha C\beta C\gamma + C\alpha S\gamma & -S\alpha C\beta S\gamma + C\alpha C\gamma & S\alpha S\beta & y \\ -S\beta C\gamma & S\beta S\gamma & C\beta & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.10)} $$

where *(x, y, z)* is the position and *α, β, γ* are the yaw, pitch and roll rotation angles, respectively, of the TCF in robot base coordinate system. The transformation matrix from the TCF frame to frame 0 is given by

$$ {}^{E}T_{0} = \begin{bmatrix} C\theta_y C\theta_z & -C\theta_y S\theta_z & S\theta_y & t_x \\ S\theta_x S\theta_y C\theta_z + C\theta_x S\theta_z & -S\theta_x S\theta_y S\theta_z + C\theta_x C\theta_z & -S\theta_x C\theta_y & t_y \\ -C\theta_x S\theta_y C\theta_z + S\theta_x S\theta_z & C\theta_x S\theta_y S\theta_z + S\theta_x C\theta_z & C\theta_x C\theta_y & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.11)} $$

where t_x, t_y, and t_z are the translation and θ_x, θ_y, and θ_z are x, y, z rotation parameters, respectively, from the TCF frame to frame 0. The other transformation matrices using D-H parameters are given by (3.12), (3.13) and (3.14),

$$ {}^{0}T_{1} = \begin{bmatrix} C\theta_p & -S\theta_p & 0 & 0 \\ S\theta_p & C\theta_p & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.12)} $$

$$ {}^{1}T_{2} = \begin{bmatrix} C\theta_{1t} & -S\theta_{1t} & 0 & a_1 \\ C\alpha_1 S\theta_{1t} & C\alpha_1 C\theta_{1t} & -S\alpha_1 & -d_1 S\alpha_1 \\ S\alpha_1 S\theta_{1t} & S\alpha_1 C\theta_{1t} & C\alpha_1 & d_1 C\alpha_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.13)} $$

where θ_p is the pan angle and θ_t is the tilt angle. The D-H parameters from frame 1 to frame 2 are $\theta_1+\theta_t$, α_1, a_1 and d_1. The angle between X_1 and X_2 is $\theta_1+\theta_t$. The angle between Z_1 and Z_2 is α_1, the separation between Z_1 and Z_2 along X_1 is a_1, and the separation between X_1 and X_2 along Z_2 is d_1.

$$ {}^{2}T_{3} = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & a_2 \\ C\alpha_2 S\theta_2 & C\alpha_2 C\theta_2 & -S\alpha_2 & -d_2 S\alpha_2 \\ S\alpha_2 S\theta_2 & S\alpha_2 C\theta_2 & C\alpha_2 & d_2 C\alpha_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.14)} $$

where θ_2, α_2,, a_2 and d_2 are D-H parameters from frame $OX_2Y_2Z_2$ to frame $OX_3Y_3Z_3$. The angle between X_2 and X_3 is θ_2. The angle between Z_2 and Z_3 is α_2. The separation between Z_2 and Z_3 along X_2 is a_2. The separation between X_2 and X_3 along Z_3 is d_2.

3.5. Barrett Hand

The Barrett Hand as shown in Figure 3.11 has 3-fingers, each finger has clutches which allow the finger to lock once it has encountered certain amount of force, thus enclosing the objects completely. Two of the three fingers can be commanded to rotate in plane (called spread of fingers) 180° symmetrically. By adjusting the spread angle of the fingers, it can grasp a wide variety of object shapes and sizes.

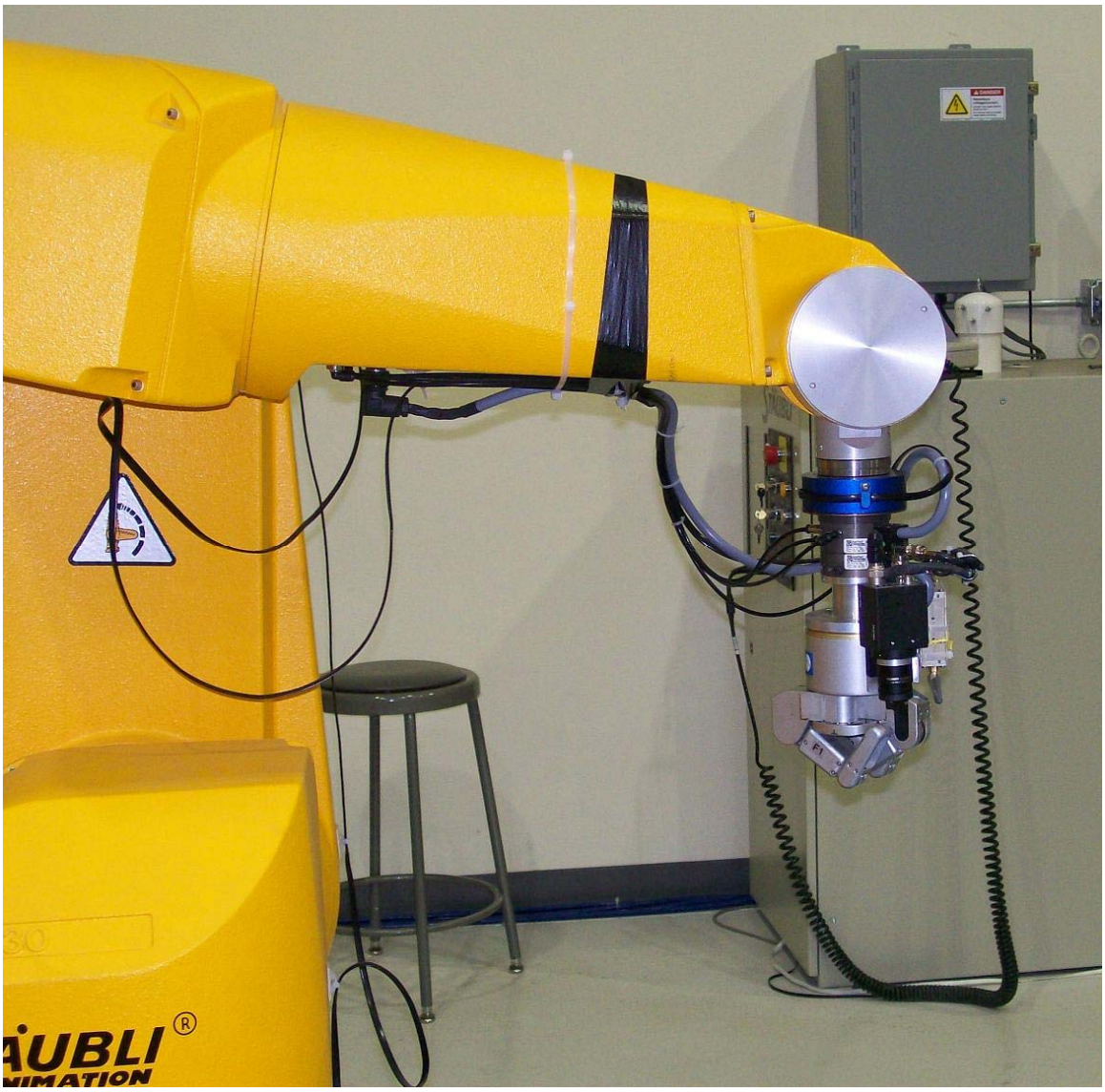

Figure 3.11. Staubli Carrying Barret Hand.

3.6. Probe Tool

A probe is a metallic pointer tool that can be attached to the robot using a tool changer. This tool is shown in Figure 3.12 is used usually to touch a point to determine its 3D location with respect to the robot base coordinate system. The probe is used in to find the 3D location of various points in the environment.

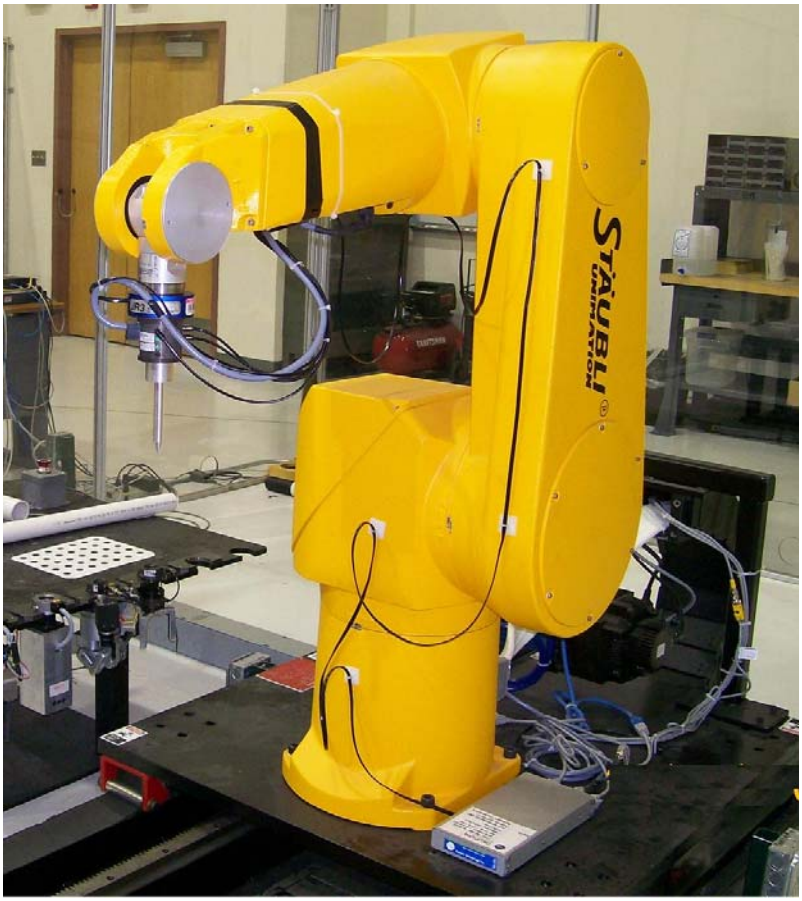

Figure 3.12. Staubli Carrying Probe Tool.

3.7. Robotic Mobile Platform (RMP)

The Segway RMP is based on the commercial Segway Human Transport. It is dynamically stable and balances itself to keep from falling over. The RMP accepts software commands and return state data on a dual channel CAN bus. The RMP is an inverted pendulum, and balances by moving forward and backward to keep itself under the center of mass above it.

The RMP consists of a steel structure for mounting various sensors computers etc. It is also fitted with ballast plates which provide mass to the top of the structure thus making it more stable when in balance mode. The RMP is a modification of the human transporter (HT); it retains the handlebar for its key and display. Two E-stops are provided for emergency use. The E-stops are attached to cords. When either cord is

pulled, the RMP's power is immediately shut off. However, this also stops the active balancing and the RMP will fall over.

Figure 3.13. Segway Robotic Mobile Platform

An emergency landing system has been designed such that when the RMP falls over it lands on this system and the equipment mounted on the RMP will not damaged. This system consists of two angular arms on each side of the RMP with spring loaded wheels at the ends of these arms. If the RMP ever exceeds 40 degrees, the control system will not be able to recover its balance and it will fall. The arms are designed at an angle so that they only touch the ground when the RMP leans forward or backward more than a critical angle (35 degrees). Thus when the RMP auto balance is turned off and when the RMP is falling it will rest on these wheels. The wheels are also equipped with the

emergency shutoff switches (which are using the E-Stops) whenever they press against something, they turn the system off thus not allowing it to go into an unstable mode.

The RMP is connected by a CAN card to a laptop computer which is secured to the top of the RMP. This laptop is the main server for the RMP. A Biclops directed vision system, described in Section 4.5 and modified for the RMP, is shown in Figure 3.14. A third PTU is used to control the direction of a laser pointer.

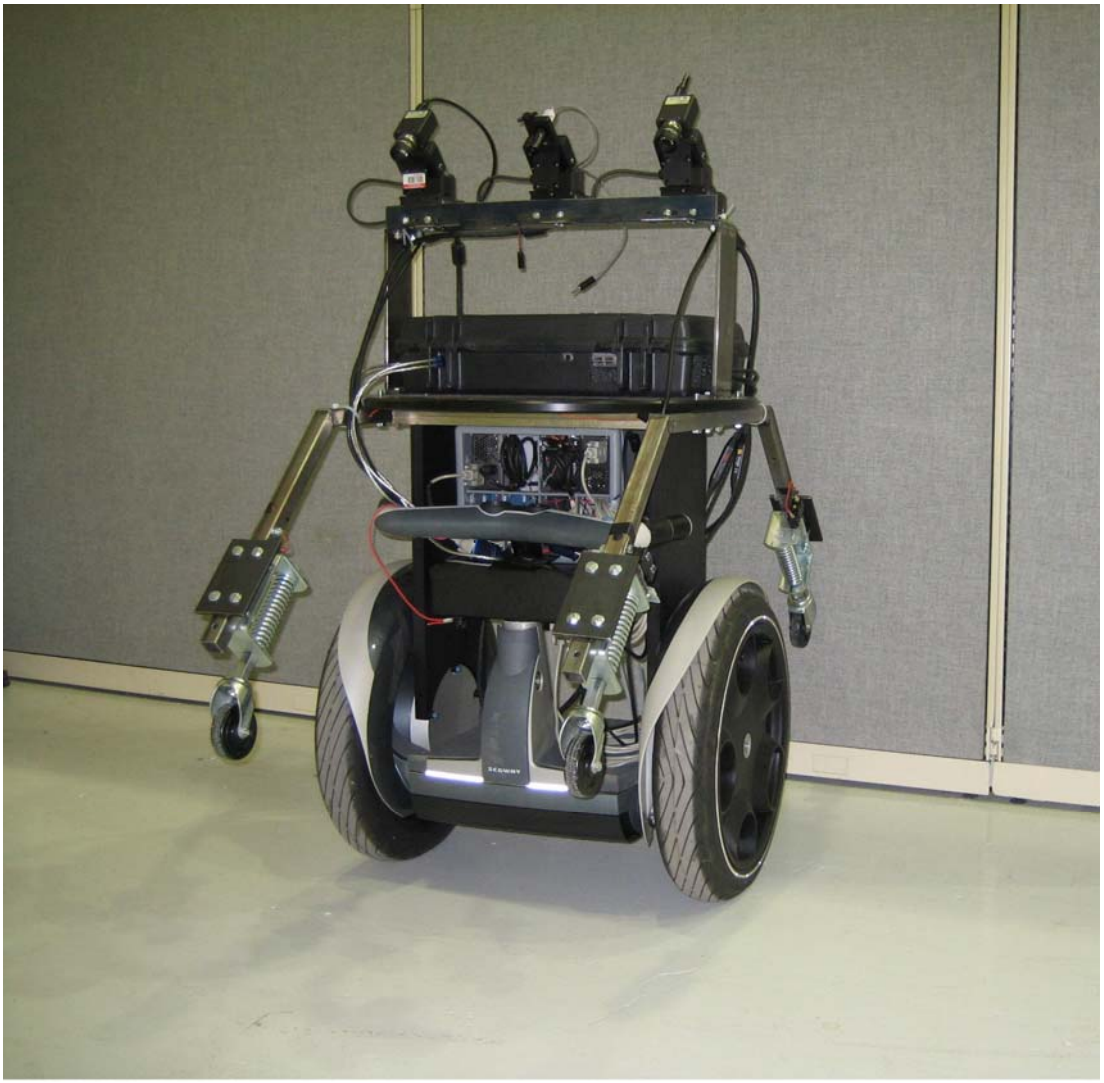

Figure 3.14. Segway Robotic Mobile Platform with Biclops Directed Vision System.

Chapter 4. Robot Calibration

4.1. Introduction

It is well known that industrial robots are highly repeatable but not very accurate. Accuracy has not been deemed necessary by industry. Since most industrial applications are programmed by teach pendant to produce a sequence of points, replay of these points relied strictly on repeatability; accuracy simply does not matter. Most of the robotic applications that capitalize on repeatability, e.g., pick and place, have already been done. For more advanced applications, such as sensor based assembly, accuracy plays a significant role. Consequently, a simple, fast, and accurate robot geometric model computed through a calibration process is needed. The goal of this chapter is to describe a new approach to improve the accuracy of the 3D structure of the scene determined by the cameras fixed on the robot using PTUs. This goal is partially reached by calibrating the robots on which these PTU cameras are mounted.

The main source of error in positioning and orientation of the robot is due to the inaccuracy in the parameters used to compute the position and orientation. The position and orientation of a robot can be represented by the forward kinematics using Denavit-Hartenberg (DH) parameters [60], [61] for each link of the robot which depend on the dimensions of the links. Robot accuracy ultimately depends on the accuracy of the DH parameters. Some variation comes from the manufacturing process, primarily due to machining inaccuracy. Other variation comes from the assembly process, where the precise position and orientation of links and joints is not perfectly repeatable. Most manufacturers of robots do not focus on accuracy because, if accuracy is achieved by higher tolerance in machining, the cost of robot increases dramatically, adversely affecting the company's sales potential.

Since it is economically unattractive to achieve accuracy by machining to higher tolerances, a software calibration approach to identify the DH parameter values is needed to advance the state of the art in robotics. After a calibration procedure in the robot factory, each robot controller can be updated, e.g., by writing non-volatile memory with the correct robot-specific DH parameter values instead of the standard design values. If this procedure is relatively rapid, all robots will leave the factory as accurate mechanisms that cost little more than they cost today. Also, for applications with higher accuracy demands like sensor based assembly, robotic surgery, *etc*., the complete set of the DH parameters can be used to compute the numerical inverse kinematics. This chapter describes a DH parameter calibration approach that fits this need. The Staubli robot and the laser tool both are calibrated.

Table 4.1. Industrial Parameters

Parameters	Initial values
α_2 (deg)	-90
a_2 (mm)	0
θ_2 (deg)	0
d_2 (mm)	0
α_3 (deg)	0
a_3 (mm)	625
θ_3 (deg)	-180
β_3 (deg)	0
α_4 (deg)	-90
a_4 (mm)	0
θ_4 (deg)	0
d_4 (mm)	625
α_5 (deg)	90
a_5 (mm)	0
θ_5 (deg)	0
d_5 (mm)	0
α_6 (deg)	-90
a_6 (mm)	0
$l\theta_x$ (deg)	0
$l\theta_y$ (deg)	0
p_x (mm)	0
p_y (mm)	0

4.2. Staubli Robot & Laser Pointer Model

The aim of this chapter is to calibrate the two robots independently. The Staubli Robots are modeled as described in section 3.1.1. The robots parameters equivalents to the industrial parameters are shown in Table 4.1. These parameters are not accurate enough so we need to calibrate the robot to find these parameters. For the sake of calibration we used a laser tool as shown in Figure 4.1. The laser tool is attached to the robot and the whole system, *i.e*., the robot and the laser pointer tool consisting of the 22 parameters is calibrated.

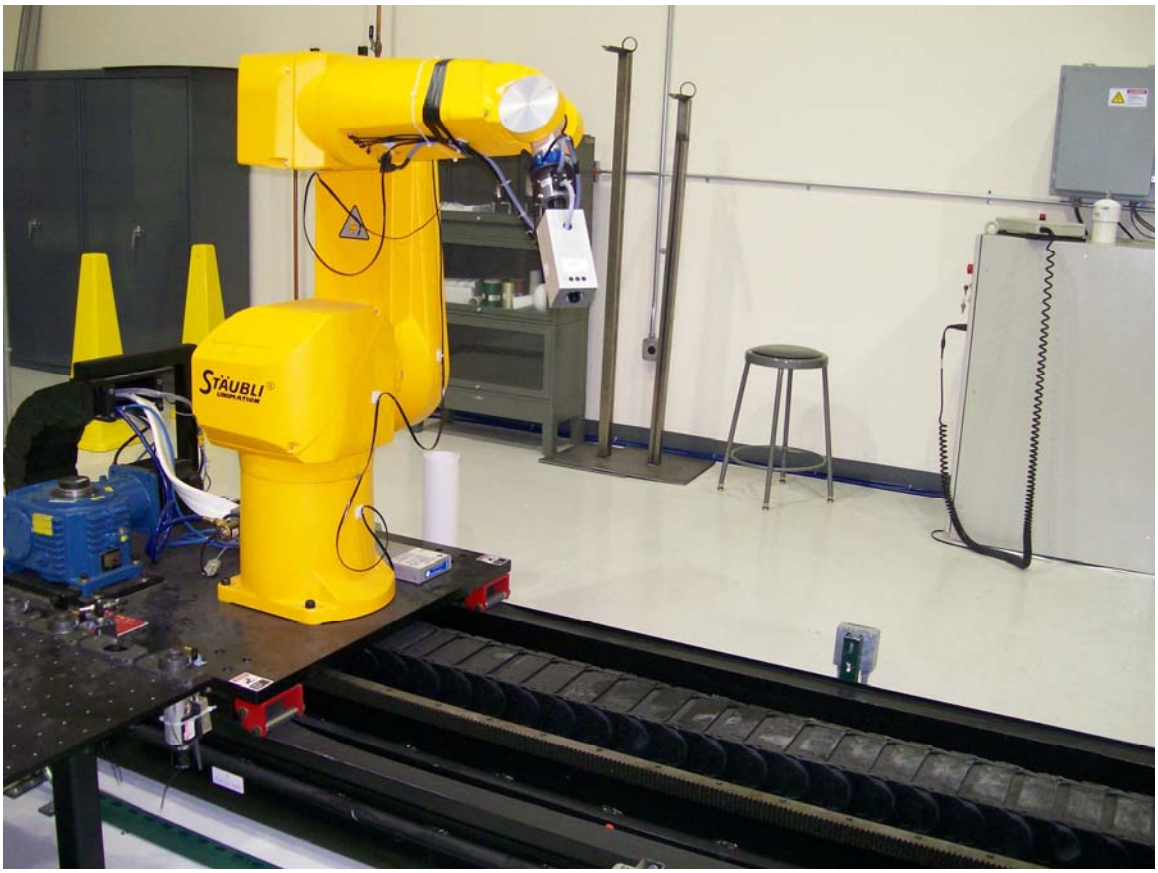

Figure 4.1. Staubli RX-130 robot, mounted on RTU, carrying a laser pointer tool.

4.3. Virtual Closed Kinematic Chain Calibration (ViCKi)

Calibration is the process of determining accurate values of all parameters of the model. For the Staubli RX-130 robot and laser tool model we have 22 parameters to be determined.

A new method called ViCKi [66], "virtual closed kinematic chain method," has been developed to calibrate the robot. This method falls under the class of closed-loop methods where only joint readings are used to calibrate the robot. A laser tool is attached to the end-effector of the robot. The laser tool on the robot acts as a virtual telescopic (prismatic) link giving the robot 7 DOF, the seventh joint being the length of the laser line from the end effector to the projected laser point on an object, e.g., distant wall. Thus

aiming the laser pointer at a fixed point (on a plane or some object) creates a virtual closed kinematic chain.

Calibration is accomplished by aiming the laser pointer at an arbitrary but fixed point *P* on an object (usually a plane), adjusting the joint values to maintain the laser on point *P*, using various joint configurations. This effectively becomes the single end point constraint for the 7 DOF system. The coordinates of the fixed point *P* in robot's base frame are unknown and must be included in the calibration model's parameters. Since the coordinates of the fixed point (P) are also included in the parameters to be determined, the scale factor of the model is indeterminate. To overcome this problem we need a second fixed point relative to the first fixed point. This can be accomplished in two ways.

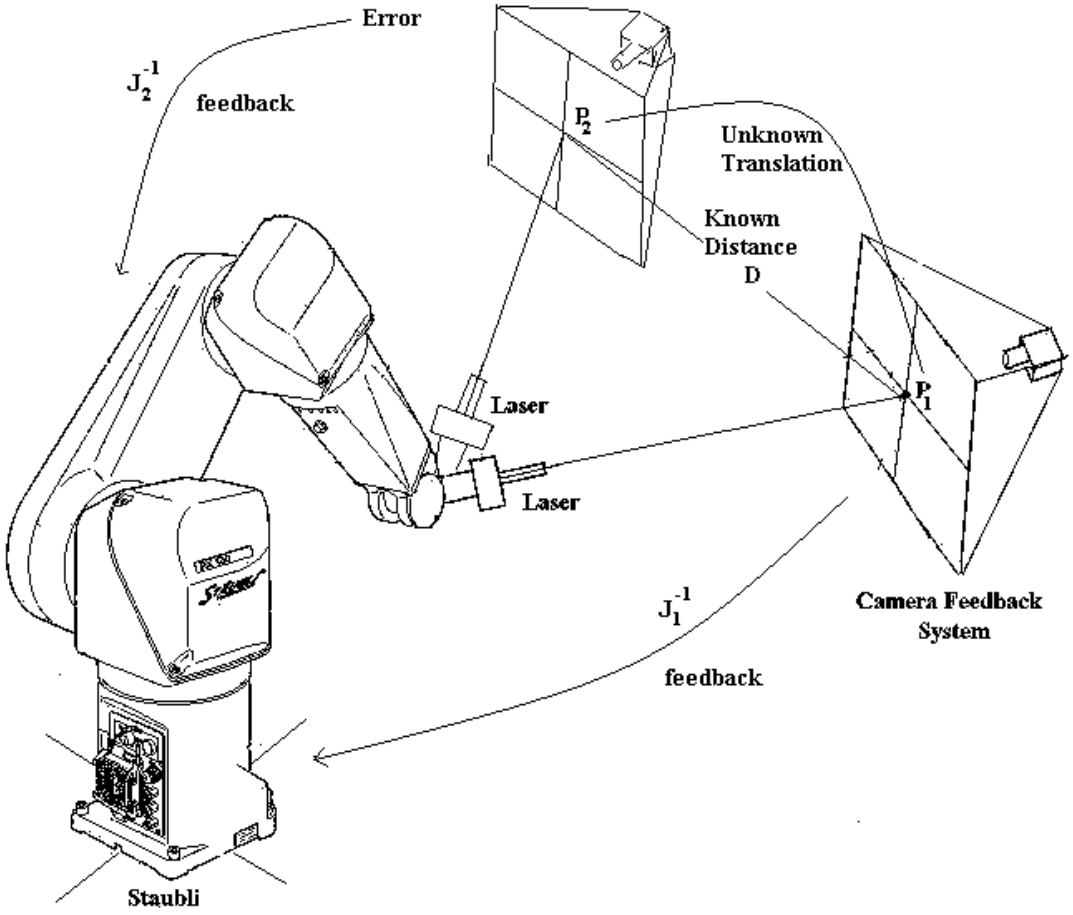

Figure 4.2. Calibration using two 3D fixed points.

A second fixed point is chosen such that the distance between the two fixed points is known (D) and the two parameters for the direction are included in the calibration model as shown in Figure 4.2. This can usually be achieved through a calibration plane.

The same fixed point is used but the base of the robot is purely translated by known distance (D) and the two parameters for the unknown translation direction are included in the calibration model as shown in Figure 4.3.

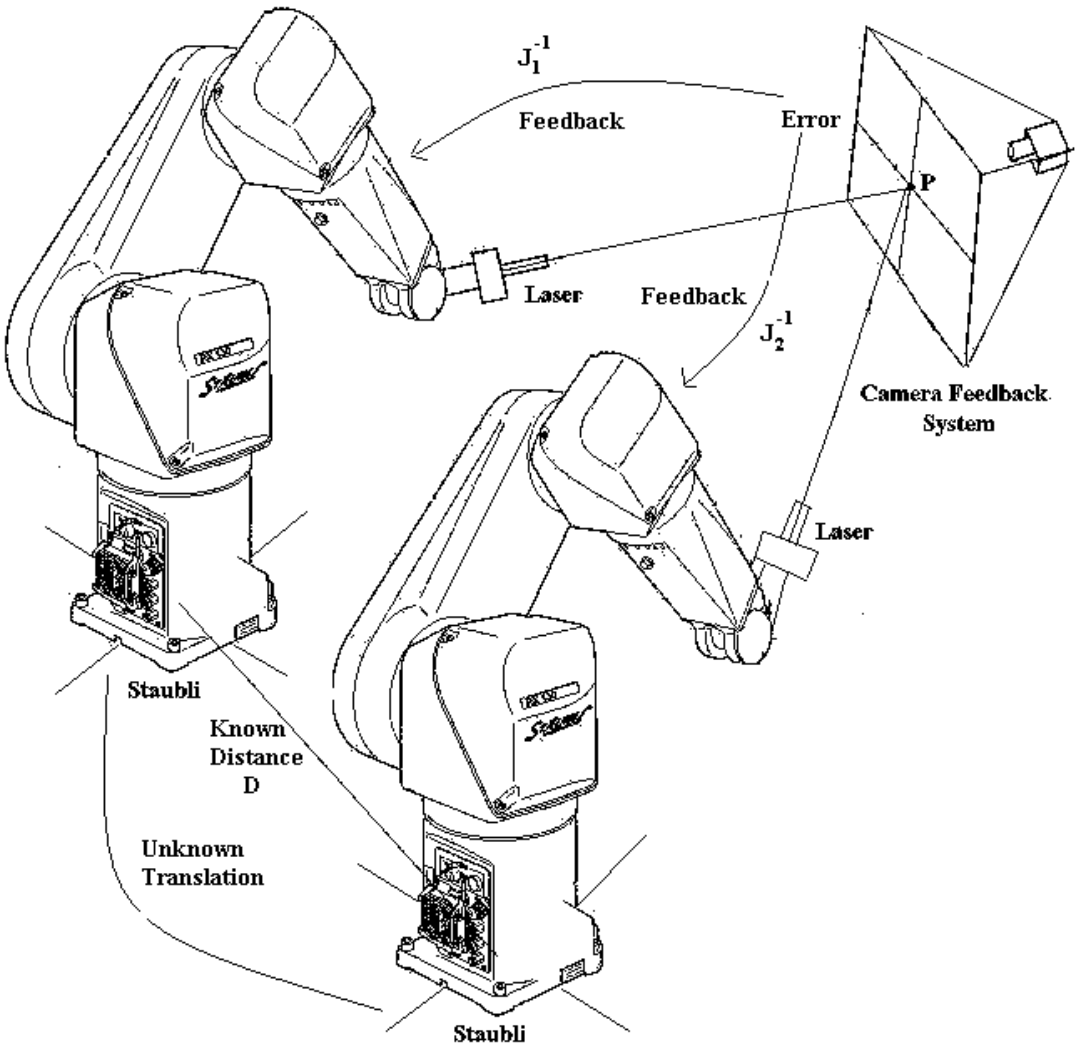

Figure 4.3. Calibrating using same 3D point but by translating base of robot.

In our case, since we have a RTU (Robot transport unit) which can translate the robot along the rail by a known distance, we use the second alternative. This eliminates

the effort of measuring the distance between two fixed points in space. The translation can be directly noted from the RTU position which has position uncertainty less than ±0.1mm. The complete calibration process now includes 18 parameters for the robot, 4 for the laser tool, 3 for the 3D coordinates of the fixed point, and 2 for the direction of unknown translation, either of the second point from first or the second position of the base frame from the first position depending on the method used (which corresponds to the direction of the RTU with respect to the robot base coordinate system).

The laser tool is aimed at the fixed point from various positions (robot joint configurations. The parameters of the model are determined by minimizing the sum of the normalized shortest distance of the fixed point from all the laser lines. Since the shortest distance error depends on the distance of the object plane from the end effector (*i.e.*, length of the laser line) it is normalized with the length of the laser line.

Consider a 3D point in space P_1. Let its coordinates in the base frame of the robot be ${}^{B1}P_1 = \begin{Bmatrix} p_x \\ p_y \\ p_z \end{Bmatrix}$ If we use method 1, with two fixed points, the second point is given by ${}^{B1}P_2 = \begin{Bmatrix} p_x \\ p_y \\ p_z \end{Bmatrix} + D * \begin{Bmatrix} t_x \\ t_y \\ t_z \end{Bmatrix}$ where $\begin{Bmatrix} t_x \\ t_y \\ t_z \end{Bmatrix}$ is the unit vector of the unknown direction of translation. D is the known distance of translation. If we use method 2, the same fixed point but from two different base locations, we have ${}^{B1}P_1$ and ${}^{B2}P_1$ where B_1 and B_2 are the two positions of the base. Now B_1 is translated by a known distance D along an unknown direction $\begin{Bmatrix} -t_x \\ -t_y \\ -t_z \end{Bmatrix}$ to reach B_2 we have $B_2 = B_1 + D * \begin{Bmatrix} -t_x \\ -t_y \\ -t_z \end{Bmatrix}$, $B_1 = B_2 + D * \begin{Bmatrix} t_x \\ t_y \\ t_z \end{Bmatrix}$. The

coordinates of the same point in base 2 coordinate system are given by

$${}^{B2}P_1 = {}^{B1}P_1 + {}^{B2}B_1 = \begin{Bmatrix} p_x \\ p_y \\ p_z \end{Bmatrix} + D * \begin{Bmatrix} t_x \\ t_y \\ t_z \end{Bmatrix} \text{ thus } {}^{B2}P_1 \equiv {}^{B1}P_2 .$$

The two ways to perform calibration are shown in Figure 4.2 and Figure 4.3. Points B_1 and B_2 are the two positions of the robot on the RTU separated by a distance of D along the unknown direction of the translation. The 'observation plane' is an arbitrary plane (any arbitrary orientation facing the robot base) passing through the 3D point at which the laser is aimed and can be adjusted to coincide with the point. The coordinates of the 3D point P_2 in the base coordinate system of B_1 are identical to the coordinates of the point P_1 in base coordinate system of B_2. Since the robot base coordinate system is not yet determined, the direction of translation is unknown. The calibration procedure thus includes three parameters for the 3D point and two parameters for the unknown direction. We choose the angles of rotation about the z-axis followed by the y-axis so as to align the base frame x-axis along the translation direction. Since the direction of translation in our present case is almost aligned along the x-axis of our robot the initial guess for all rotation parameters is zero.

We denote the parameters for the robot and laser tool as Φ_{RL} which is the list of 22 parameters for the robot and the laser tool. The parameters for the 3D point are denoted as Φ_{3D} and the parameters for the direction of translation as Φ_l and Φ_m.We have two sets of joint configurations, one for each fixed point (or one for each base position). We use the notation ${}^1J_i$ for joint set 1, i^{th} reading. For each joint configuration in first set transformation matrix from base to laser frame ${}^1T_i\left(\Phi_{RL}\ {}^1J_i\right)$ is computed which gives the

position ${}^{1}P_i\left(\Phi_{RL}\ {}^{1}J_i\right)$ and the direction of the Z-axis ${}^{1}Z_i\left(\Phi_{RL}\ {}^{1}J_i\right)$ of the laser line in columns 4 and 3, respectively, of the transformation matrix. The shortest distance of the 3D point Φ_{3D} from the laser line is determined and normalized with the length of the laser line. This is the error, *i.e.*, the cost function ${}^{1}E_i\left(\Phi_{RL}\ \Phi_{3D}\ {}^{1}J_i\right)$ for this joint configuration. For the second set the error ${}^{2}E_i\left(\Phi_{RL}\ \Phi'_{3D}\ {}^{1}J_i\right)$ is computed with the 3D point given by

$$\Phi'_{3D}=\Phi_{3D}+\begin{Bmatrix} cos\left(\Phi_l\right)cos\left(\Phi_m\right) \\ sin\left(\Phi_l\right)cos\left(\Phi_m\right) \\ -sin\left(\Phi_m\right) \end{Bmatrix}*D \qquad \textbf{(4.1)}$$

where Φ_l and Φ_m are the angles of rotation about z and then y to align the x-axis along the unknown translation and D is the known length of the translation. The translation direction in the present case is almost parallel to the x-axis of the robot base coordinate system hence zero can be used as the initial guess for these parameters. The complete parameter set $\Phi=\left\{\Phi_{RL}\ \Phi_{3D}\ \Phi_l\ \Phi_m\right\}$ is determined by minimizing the total sum of the squares error. The required parameter set Φ_{min} for N readings in each set is obtained by

$$\Phi_{min}=MIN_{\Phi}\left(\sum_{i=1}^{N}{}^{1}E_i^2+\sum_{i=1}^{N}{}^{2}E_i^2\right) \qquad \textbf{(4.2)}$$

4.3.1. Feedback System and Stability

The calibration procedure requires many different robot joint configurations that aim the laser tool at the particular point on a distant object. This process can be time consuming if a teach pendant is used. Instead we use the approximate parameter values

and approximate coordinates of the 3D point and compute various joint values that aim at the fixed location. But since our computation is only approximate, the laser tool only aims close to the desired point. We developed a feedback system that uses the errors in the projected point to aim the laser tool at a desired point on a plane by adjusting the joint angles. The data acquisition process is thus accelerated by this feedback system that redirects the laser spot on a plane to a desired location.

A model of the robot and laser tool is constructed in Simulink. Using approximate parameter values of the model, the joint values can be computed to aim the laser tool close to the desired point on a plane. The errors in aiming the laser at a fixed point due to errors in parameters are within ±10mm, which are well within the limits of the feedback system that can redirect the laser to the desired location from an initial location w, more than ±500mm away from the desired point. The feedback system uses the initial parameters and approximate position and orientation of the camera system to compute the inverse Jacobian. The system appears very stable since the direction of motion of each joint to move the desired laser point is not sensitive to the small differences in the robot calibration parameters. We have constructed two different feedback systems. The first uses position control as shown in Figure 4.4 to control the joint angles. The second uses a PID (position, integral and differential) control as shown in Figure 4.5. The gains and parameters for the PID feedback system are chosen to make the system stable.

Using this approach, the calibration procedure can be automated in industry. A camera detects the laser point and is used as feedback to change the robot joint angles so as to move the laser point to the desired point P as shown in Figure 4.6.

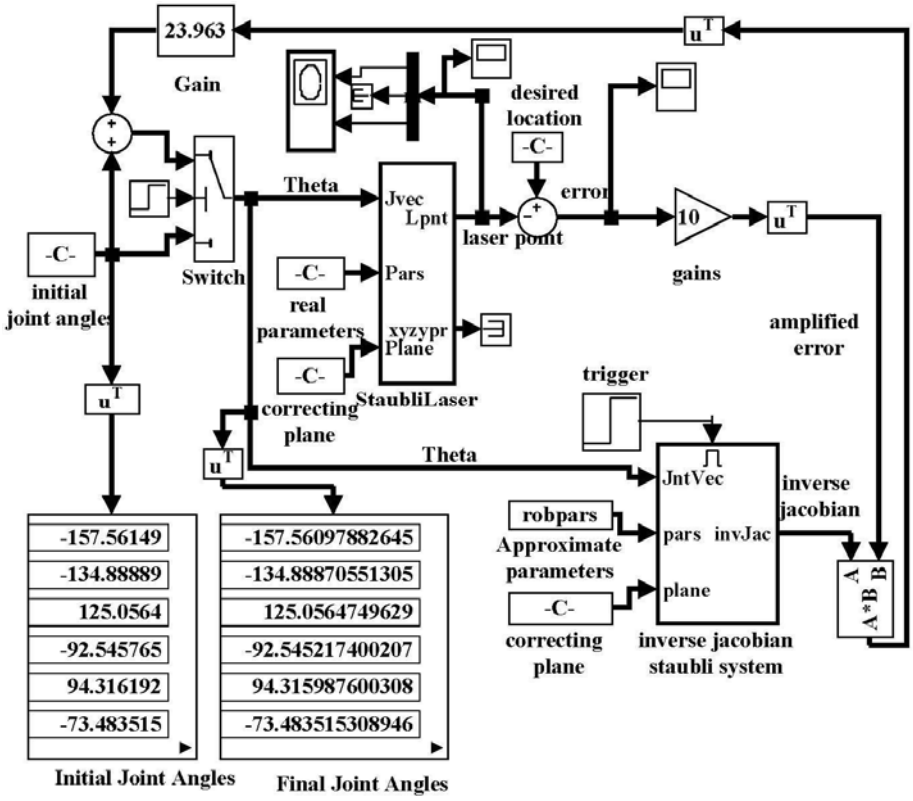

Figure 4.4. Position control Feedback system to aim the laser tool, in Simulink.

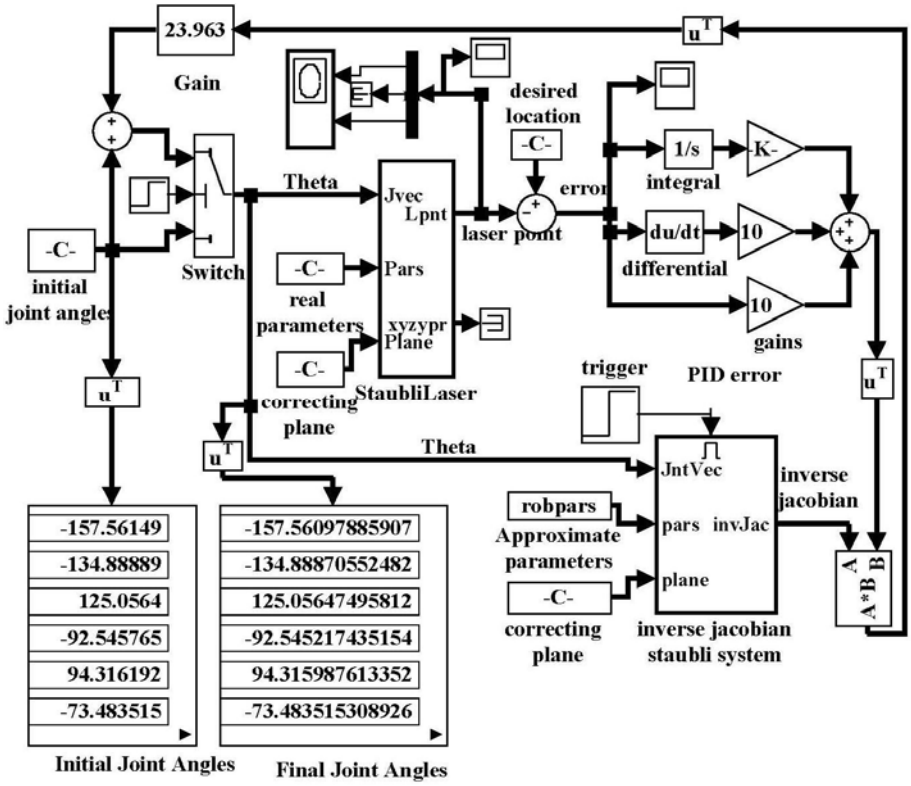

Figure 4.5. PID Feedback system to aim the laser tool, in Simulink.

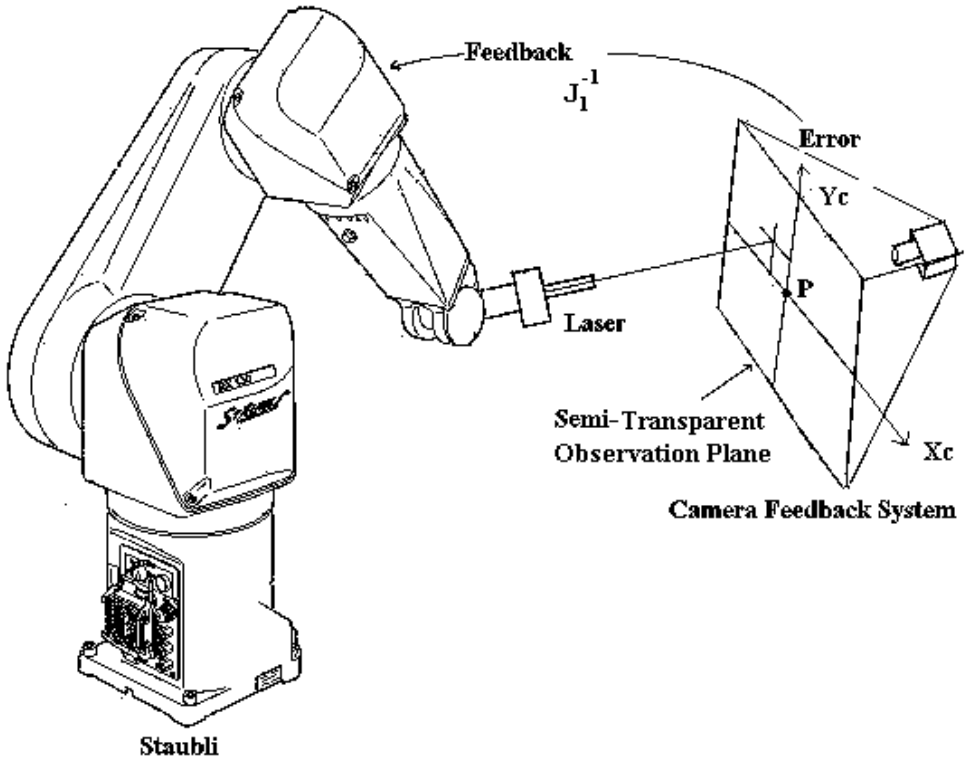

Figure 4.6. Staubli Feedback system, using a camera.

4.3.2. Minimization

We employ a MATLAB minimization routine (lsqcurvfit) to compute the values of the parameters. The algorithm computes the solution for the system using Levenberg-Marquardt (LM) algorithm and the solution obtained is perturbed with random noise and the algorithm is repeated. The amount of noise added to the solution in each subsequent step is reduced. This randomness in the initial guess to the LM algorithm and the repetition of the algorithm ensures a better solution. The user is referred to [63], [64] for description of the "trust region" problem and [59] for Levenberg-Marquardt Algorithm. The number of constraints required by the algorithm to converge to a good model is usually at least five times the number of parameters in the model. Therefore, we need at least 5*27=135 sample configurations that aim at the same point.

4.3.3. Procedure

The procedure is summarized in these steps

1. A very large set (M) of random joint configurations within the required ranges is generated automatically.
2. Using the approximate parameter values of the robot, laser and the 3D point a subset (N) of joint configurations is selected that aim the laser tool close to the 3D point.
3. The robot is moved to each of the N locations in the joint configuration set and the joint values are adjusted to aim the laser point at the constant location. This accurate joint configuration is stored.
4. Step 3 is repeated for all of the N joint configurations.
5. Move the Robot on the RTU by a known distance D=1000mm (or use a second fixed point at a known distance from first).
6. Repeat the steps 1 through 4 by aiming at the same location used previously (or aim at the second fixed point).
7. Using these data and a nonlinear minimization routine (lsqcurvfit), compute the parameter values. We used the shortest distance of the aimed point from the laser line normalized by the distance of the aimed point from origin of laser coordinate system as the cost function for minimization.

4.4. Analysis of ViCKi

4.4.1. Magnification of Observation Error

The present method uses a laser pointer tool on the robot's end-effector to aim at a constant location on a distant object. Small variations in position and orientation of the end-effector are magnified on the distant object, thus the position of the laser point is sensitive to joint values. It can be shown mathematically that by projecting the laser pointer onto a distant object the resolution of observations is improved, effectively increasing the accuracy of measurements of the joint angles required to calibrate the robot. The observation plane is a plane which passes through the 3D fixed point. By adjusting the robot joint values, the laser tool can be aimed at the desired point. It should be noted that this observation plane can be chosen arbitrarily passing through the 3D fixed point; the only effect will be a change in the Jacobian used in feedback. Hence, without loss of generality assume the observation plane is parallel to *X*-*Z* and at a distance "β", so that $Y= \beta$ is the equation of the plane. Let the transformation matrix of the laser pointer with respect to the base of the robot be

$$ {}^{0}T_{7}=\begin{bmatrix} R_{11} & R_{12} & R_{13} & O_x \\ R_{21} & R_{22} & R_{23} & O_y \\ R_{31} & R_{32} & R_{33} & O_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \textbf{(4.3)} $$

Then we know that $O= (O_x, O_y, O_z)^T$ is the origin of the laser coordinate system and $Z=(R_{13}, R_{23}, R_{33})^T$ is the Z-axis, which is coincident with the laser line. Any general point on this line (*i.e.*, on the laser line) is $P=O+\lambda Z$. The laser is aimed at the observation plane (whose equation is $Y=\beta$) hence the aimed point (P) lies on the observation plane and its solution is

$$P_y=\beta \Rightarrow O_y+\lambda R_{23}=\beta \Rightarrow \lambda=\frac{\left(\beta-O_y\right)}{R_{23}} \quad \textbf{(4.4)}$$

Using above equation we can compute the location of the laser point on the observation plane in its planar 2D coordinates (x-z in the present case), which is given by

$$\begin{Bmatrix} K_x & K_z \end{Bmatrix}^T = \left\{ O_x+\left(\beta-O_y\right)\frac{R_{13}}{R_{23}} \quad O_z+\left(\beta-O_y\right)\frac{R_{33}}{R_{23}} \right\}^T \quad \textbf{(4.5)}$$

Since we choose β to be large, the change in the 2D point of projection is magnified as compared to just the change in the position and/or orientation of the end effector. (β-Oy) is the distance between the end-effector or TCF (Tool Control frame) and the plane. R13/R23 and R33/R23 are the tangents of the angles of the laser line with the Y- axis.

Figure 4.7 shows the variations of the positions of the TCF and the laser point on planes at β=3000 mm and β=5000 mm with variations in joint 1 angle. It can be inferred from the plot that to have same error (0.1mm) in measurements of either TCF or laser position (on plane β=5000 mm), the errors in joint 1 angle are 0.01deg and 0.001deg respectively, *i.e.*, the error for laser point is 10 times lower than that of TCF. Consequently, if the error associated with aiming the laser point at the desired point that is 5000mm away is less than 0.1mm, the error associated with joint 1 will be less than 0.001deg. Figure 4.8 shows the variations of the positions of the TCF and the laser point on planes at β=3000mm and β=5000mm with variations in joint 2 angle. The variations in laser point and TCF have the same order of magnitude as variations in other joint angles. Thus this method is more accurate in obtaining the joint configuration data when compared to any method that uses the TCF position or orientation measurements.

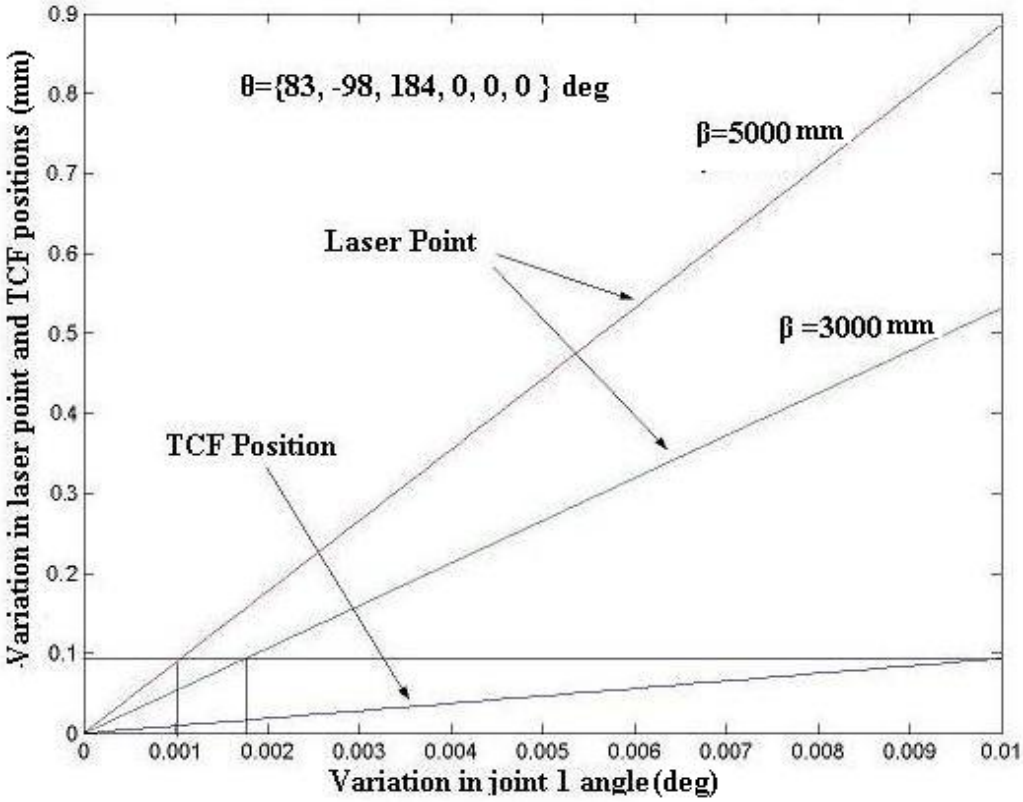

Figure 4.7. Changes in positions of laser point and TCF with joint 1 angle.

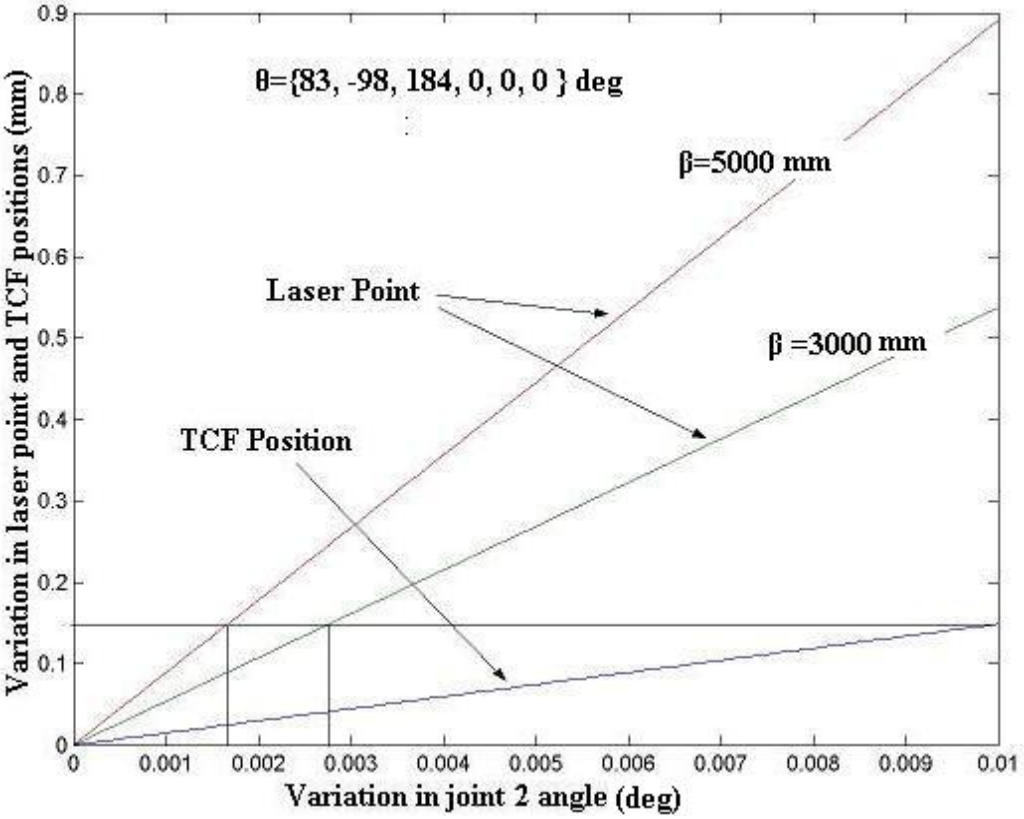

Figure 4.8 Changes in positions of laser point and TCF with joint 2 angle.

4.4.2. Optimum Distance for Observations

The proposed method magnifies the error in observations as discussed in previous section. This magnification is directly dependent on the distance (β) of the observation plane from base of the robot. Though the magnification increases with the distance of the observation plane, the uncertainty in aiming the laser (due to limited resolution of the joints) increases. Hence there is a maximum distance (β) beyond which the magnification in error does not help in getting better joint angle data.

The uncertainty in origin of laser line (ΔO) and laser angle ($\Delta\theta$) are given by the following equations respectively.

$$\Delta O = O_m + O_s \quad \textbf{(4.6)}$$

$$\Delta\theta = \theta_m + \theta_s \quad \textbf{(4.7)}$$

where O_m and O_s are mean and standard deviation of errors in origin of laser, and θ_m and θ_s are the mean and standard deviation of errors in direction of laser due to limited resolution of joints.

The uncertainty in aiming laser (ΔL) at a distance λ is given by

$$\Delta L = \Delta O \pm \lambda \sin(\Delta\theta) \quad \textbf{(4.8)}$$

Let ΔP be the uncertainty in observations due to its finite resolution (of camera system or human observer). To get better observations we have $\Delta L = \Delta O \pm \lambda \sin(\Delta\theta) \le \Delta P$. Hence the distance of aiming λ can be computed as

$$\lambda \leq \left(\frac{(\Delta P \pm \Delta O)}{\sin(\Delta\theta)} \right)_{\min} \quad \textbf{(4.9)}$$

The distance computed by the above equation is the optimum distance of the observation plane from the laser origin. A shorter distance does not magnify the errors sufficiently and a longer distance does not provide better joint angle data.

The real robot has limited resolution. The encoder counts and the resolution of each joint for Staubli robot are shown in Table 4.2. A large number (30000) of positions in the workspace are determined. Two robots are created in simulation, one is moved to each of the joint configurations, and the other is moved to the same joint configurations but with offset equal to maximum joint resolution errors. The norm of the difference in position and the angle between laser lines are computed. The mean and standard deviation of these values for all joint configurations are computed. The mean and standard deviation of the norm of position error were O_m= 0.000236mm and O_s= 0.0000577mm, respectively. The mean and standard deviation of the angle between laser lines were θ_m=0.00217deg and θ_s= 0.002583 deg respectively.

Table 4.2. Joint Resolutions

Joint	Encoders Counts/deg	Resolution (deg)
1	147219	6.79e-6
2	147219	6.79e-6
3	115019	8.69e-6
4	100124	9.99e-6
5	30720	3.26e-5
6	36409	2.75e-5

If a camera system is used the resolution of the camera system is 1 pixel which corresponds to the resolution of approximately 0.05mm in the observation plane. Hence we can safely use ΔP =3x0.05=0.15mm which accounts for other errors in observations (computing center of laser point). The distance of aiming can be computed using (4.9) as $\lambda \approx$ 1800mm. The average distance of the laser origin from base of the robot is around 1200mm for various joint configurations. Hence the optimum distance of the observation plane is β =1800+1200=3000mm.

If observations are taken by a human, the resolution of observations can be around ΔP =0.5mm. The distance of aiming can be computed by (4.9) as $\lambda \approx$ 6000mm. Hence the optimum distance of the observation plane is β =6000+1200=7200mm.

4.4.3. Effect of Scaling Distance

The distance (D) between two 3D fixed points or the robot base positions as described in Section 4.3 has a scaling effect on all the length parameters of the model. The ratio of error in D is same as the ratio of errors in parameters. Hence choosing D large enough to outweigh its uncertainty is necessary, *i.e.,* if D=1000mm and it repeatability is 0.1mm then the length parameters have repeatability of 1.0×10^{-4}.

4.5. Experiments

4.5.1. Calibration with Precise Data in Simulation

Simulation of the calibration experiment was performed using very accurate joint readings. A robot was created in Simulink with a known set of parameters different from the industrial parameters (approximate DH parameters used by the robot manufacturer). The feedback system was used to accurately aim the laser tool at a fixed point from two different RTU locations (1000mm apart). The laser point is finely adjusted using the feedback loop as discussed in section 4.3.1 until the errors in projected points were within ±0.0005mm and the joint configurations were recorded with full accuracy. These joint configurations were used to calibrate the robot; the industrial parameter set was used as the initial solution. The solution was compared to the actual parameters used to generate the joint configuration data. The experiment was repeated with different fixed points and

various sets of joint configurations. The deviations of the parameters from the actual parameters were computed.

Table 4.3 shows the results of the calibration with accurate data. Column 2 shows the actual parameters used by the simulation robot to obtain the data, *i.e.*, the true robot parameters. Column 3 shows the initial parameters used in the minimization routine. Column 4 shows the optimum solution and column 5 shows the standard deviation of the parameters from the actual parameters obtained by repeating the experiment number of times. The results show that the optimum solution obtained by the calibration experiment was very close

Table 4.3. Calibration Results (Precise Data)

Parameters	Actual	Initial values	Minimum Error	STD ($X10^{-4}$)
α_2 (deg)	-89.99	-90	-89.999012	8.318
a_2 (mm)	0.05	0	0.049711	5.0281
θ_2 (deg)	-0.03	0	-0.030618	7.0947
d_2 (mm)	0.70	0	0.699360	4.2889
α_3 (deg)	0.01	0	0.010936	3.0462
a_3 (mm)	625.5	625	625.500531	1.8965
θ_3 (deg)	-179.99	-180	-179.997532	1.9343
β_3 (deg)	-0.04	0	-0.041009	6.8222
α_4 (deg)	-89.98	-90	-89.991267	3.0276
a_4 (mm)	-0.001	0	-0.001211	5.4167
θ_4 (deg)	0.036	0	0.036313	1.5087
d_4 (mm)	626.25	625	626.253214	6.9792
α_5 (deg)	90.01	90	90.011123	3.7837
a_5 (mm)	-0.53	0	-0.530183	8.6001
θ_5 (deg)	-0.09	0	-0.088865	8.5366
d_5 (mm)	0.55	0	0.550291	5.9356
α_6 (deg)	-90.02	-90	-90.012512	4.9655
a_6 (mm)	0.53	0	0.53094	8.9977
$l\theta_x$ (deg)	0.05	0	0.049648	8.2163
$l\theta_y$ (deg)	0.16	0	0.15952	6.4491
p_x (mm)	-0.18	0	-0.17961	8.1797
p_y (mm)	1.63	0	1.6293	6.6023

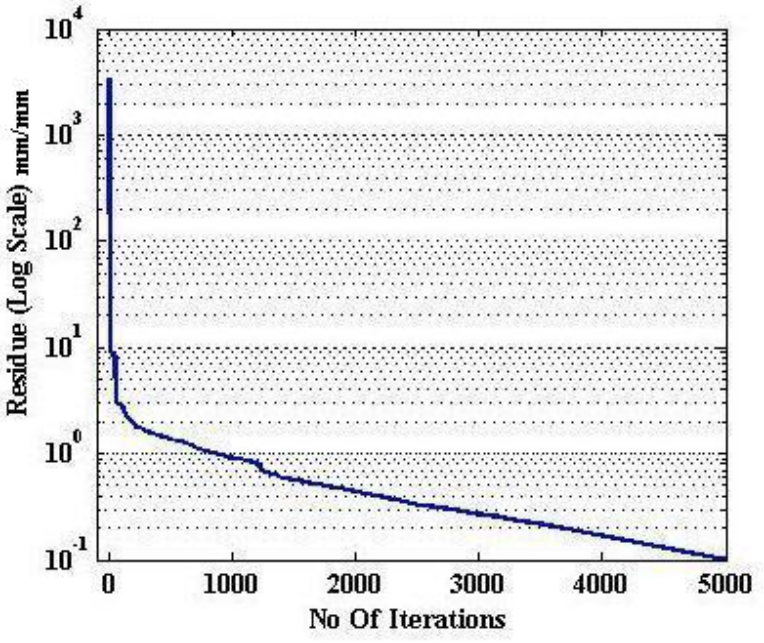

Figure 4.9. Minimization Routine Residue (log scale) Vs Number of Iterations.

to the actual solution. The standard deviation of the solution was very small (10^{-4}) indicating the stability of the procedure.

The minimization routine residue error was plotted against the number of iterations as shown in Figure 4.9. The error in projection decreases as the model is improved in each iteration.

4.5.2. Calibration with Noisy Data in Simulation

A real robot joint has limited resolution and can only be commanded to move to a particular encoded value closest to the desired angle. The encoder counts per degree for all six joints of Staubli were shown previously in Table 4.2. Therefore the accuracy of the joints is limited by the number of encoder counts per degree. In the previous experiment we obtained the joint readings with very high accuracy which is not possible for the real robot.

Hence, to make the simulation more realistic, we use joint readings with accuracy similar to that of the real robot. This was achieved by adding noise and rounding to the closest encoder value of the joint. The calibration experiment was repeated with these noisy data.

Table 4.4. Calibration Results (Noisy Data)

Parameters	Actual	Initial values	Minimum Error	STD ($X10^{-3}$)
α_2 (deg)	-89.99	-90	-89.988292	3.2004
a_2 (mm)	0.05	0	0.059942	9.6010
θ_2 (deg)	-0.03	0	-0.025602	7.2663
d_2 (mm)	0.70	0	0.703400	4.1195
α_3 (deg)	0.01	0	0.013142	7.4457
a_3 (mm)	625.5	625	625.503650	2.6795
θ_3 (deg)	-179.99	-180	-179.986067	4.3992
β_3 (deg)	-0.04	0	-0.034084	9.3338
α_4 (deg)	-89.98	-90	-89.978802	6.8333
a_4 (mm)	-0.001	0	-0.000618	2.1256
θ_4 (deg)	0.036	0	0.040585	8.3924
d_4 (mm)	626.25	625	626.258698	6.2878
α_5 (deg)	90.01	90	90.019342	1.3377
a_5 (mm)	-0.53	0	-0.527355	2.0713
θ_5 (deg)	-0.09	0	-0.088396	6.0720
d_5 (mm)	0.55	0	0.558728	6.2989
α_6 (deg)	-90.02	-90	-90.017621	3.7048
a_6 (mm)	0.53	0	0.536458	5.7515
$l\theta_x$ (deg)	0.05	0	0.059668	4.5142
$l\theta_y$ (deg)	0.16	0	0.166649	0.4389
p_x (mm)	-0.18	0	-0.171296	0.2718
p_y (mm)	1.63	0	1.630099	3.1269

Table 4.4 shows the results of the calibration with noisy data. Column 2 shows the true robot parameters used by the simulation. Column 3 shows the initial parameters used in the minimization routine. Column 4 shows the optimum solution and column 5 shows the standard deviation of the parameters from the actual parameters obtained by repeating the experiment number of times.

The results show that the optimum solution obtained by the calibration experiment with truncated joint data was very close to the actual solution but not as close as the un-truncated data simulation case described in Section 4.5.1. As expected, the standard deviation of the solution using discrete joint values was greater than that of the continuous case, but it was small enough (10^{-3}) to justify that the procedure produces a usable result.

4.5.3. Calibration of Staubli Robot

The calibration experiment was performed on real Staubli RX130 robots. A point was chosen on a distant plane and the various joint values that aim at this location were determined as described in Section 4.3.3. We used the second method, *i.e.*, the robot was translated on an RTU (We assume that the direction of the translation is unknown.) and the laser was aimed at the same point. The joint readings were used in a minimization routine to find the parameters of the model. The calibration experiment used 540= 20*27 joint configurations, to compute the parameters. The experiment was repeated multiple times with different fixed points and the calibration parameters resulting from these calibrations (which all should be the same) were compared by computing the mean and deviations of the parameters. Using various fixed points and conducting the experiment

ensures that the parameters obtained are stable and accurate. The optimum solution is also shown in the tables.

Table 4.5 and Table 4.6 shows the results of the calibration of the real robot-laser tool systems also the direction of the RTU with respect to each robot is shown. Column 2 shows the initial parameters used in the minimization routine, *i.e.,* the model parameters provided by the factory controller. Column 3 shows the optimum solution; column 4 shows the mean of the parameters and column 5 shows the standard deviation of the parameters from multiple trials. The standard deviation of the solution was also small $\left(10^{-3}\right)$ indicating the stability of the procedure. The accuracy of the obtained solution is discussed in next section.

Table 4.5. Calibration Results for Robot 1

Parameters	Initial values	Minimum Error	Mean	STD ($X10^{-3}$)
α_2 (deg)	-90	-89.980456	-89.981214	0.5407
a_2 (mm)	0	0.069093	0.070235	1.2749
θ_2 (deg)	0	-0.002119	-0.003301	1.3985
d_2 (mm)	0	0.688913	0.683071	1.6236
α_3 (deg)	0	0.006887	0.007243	0.4739
a_3 (mm)	625	625.508587	625.506285	2.5518
θ_3 (deg)	-180	-179.998170	-180.002541	2.2271
β_3 (deg)	0	-0.043046	-0.045487	2.4402
α_4 (deg)	-90	-89.989007	-89.987637	1.4748
a_4 (mm)	0	-0.000193	-0.002914	2.8074
θ_4 (deg)	0	0.036948	0.037581	0.404
d_4 (mm)	625	626.245066	626.243121	2.0783
α_5 (deg)	90	90.008879	90.009645	0.6830
a_5 (mm)	0	-0.526270	-0.524321	2.1335
θ_5 (deg)	0	-0.097083	-0.099621	4.3357
d_5 (mm)	0	0.546777	0.545122	1.4124
α_6 (deg)	-90	-90.025081	-90.023421	1.7659
a_6 (mm)	0	0.530810	0.534164	3.1956
$l\theta_x$ (deg)	0	0.050220	0.056354	5.8913
$l\theta_y$ (deg)	0	0.167296	0.164381	2.6737
p_x (mm)	0	-0.185588	-0.1831072	2.3211
p_y (mm)	0	1.638139	1.636521	1.6786
Φ_l(deg)	0	0.012141	0.012201	1.1761
Φ_m(deg)	0	0.002541	0.002593	1.0572

4.5.4. Accuracy of Staubli Robot

To test the accuracy of the calibration procedure and to compare the accuracy of computed parameters with the industrial parameters, the laser was aimed at a fixed location using various robot configurations. The laser lines were computed for each position using industrial parameters and calibrated parameters. These laser lines usually do not intersect at a common point. Therefore an optimum 3D 'closest point' which is closest to all the lines was found in both cases. The projections of all the lines onto a plane passing through this 'closest point' were plotted. The larger the error in the parameters, the greater will be the scattering in the projected points.

Table 4.6. Calibration Results for Robot 2

Parameters	Initial values	Minimum Error	Mean	STD ($X10^{-3}$)
α_2 (deg)	-90	-90.017231	-90.018914	0.8172
a_2 (mm)	0	-0.032134	-0.033294	2.3143
θ_2 (deg)	0	0.011395	0.017531	1.8593
d_2 (mm)	0	0.986912	0.985234	1.2734
α_3 (deg)	0	-0.612876	-0.614178	4.0712
a_3 (mm)	625	624.987433	624.991747	3.2156
θ_3 (deg)	-180	-180.120867	-180.125123	4.5191
β_3 (deg)	0	0.033167	0.035199	0.7722
α_4 (deg)	-90	-90.114360	-90.119038	5.1782
a_4 (mm)	0	-0.341193	-0.347345	2.8141
θ_4 (deg)	0	0.008118	0.009256	0.9054
d_4 (mm)	625	626.873422	626.877954	3.1378
α_5 (deg)	90	90.112369	90.112139	1.2700
a_5 (mm)	0	-0.526270	-0.525183	0.8765
θ_5 (deg)	0	0.019845	0.019882	1.7653
d_5 (mm)	0	0.753422	0.754841	2.7804
α_6 (deg)	-90	-90.067334	-90.068136	5.2309
a_6 (mm)	0	0.315344	0.311782	6.7126
$l\theta_x$ (deg)	0	-0.332719	-0.332910	3.3119
$l\theta_y$ (deg)	0	0.912828	0.914921	2.4285
p_x (mm)	0	-0.876296	-0.878541	1.2077
p_y (mm)	0	1.221845	1.226981	2.9541
Φ_l(deg)	0	-0.572108	-0.578287	2.7691
Φ_m(deg)	0	0.098826	0.099328	3.8715

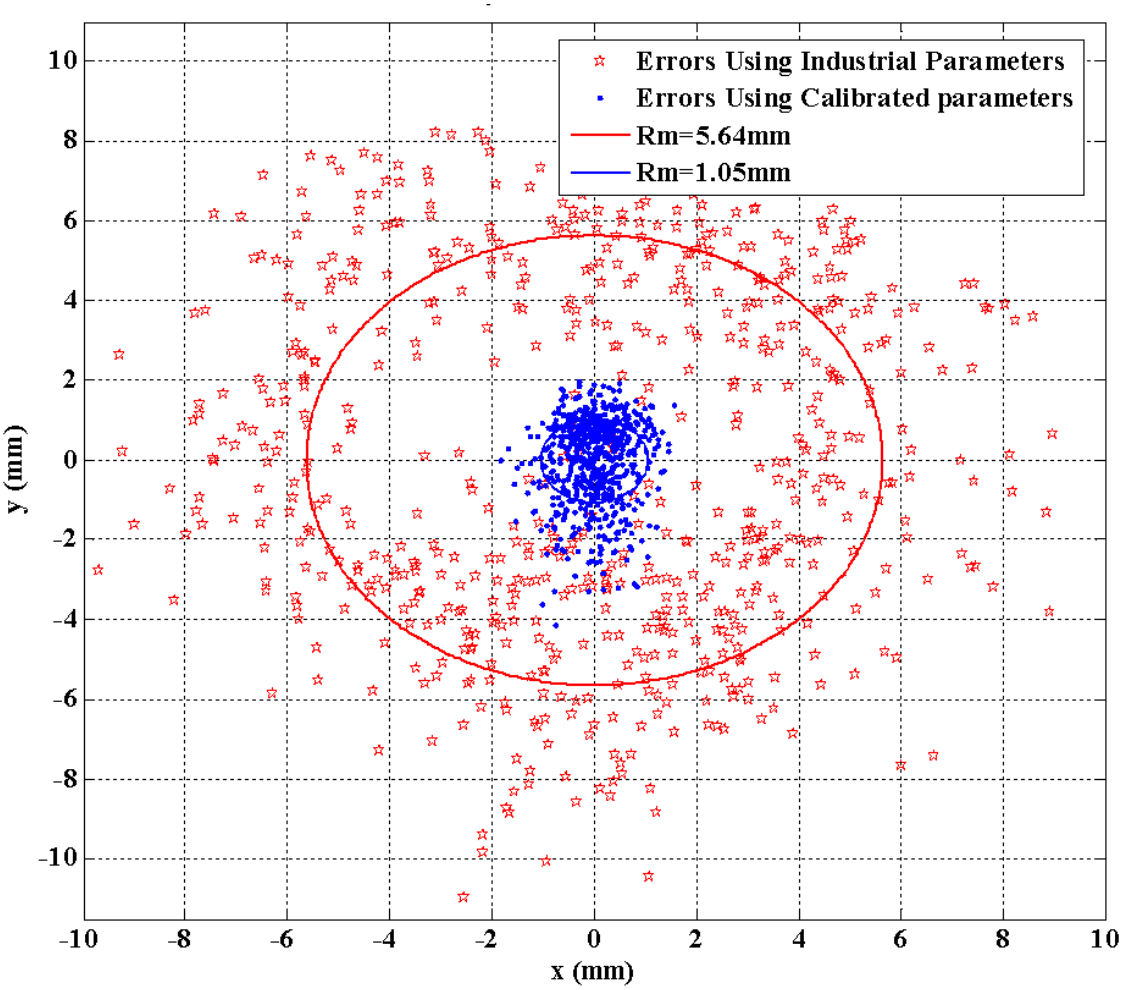

Figure 4.10. Laser projection errors using industrial (red) and calibrated parameters (blue).

The errors in the projection using industrial and calibrated parameters for real robot are shown in Figure 4.10. Since the robot was aiming at the same point from various robot configurations, all of the projected points should be coincident. The spread of these projected points relates to the accuracy of the estimated DH parameters of the robot. Figure 4.10 shows that the spread of projected points. Using the original DH parameters built into the commercial controller the maximum, mean and standard deviation of the radius of spread were 11.25mm, 5.64mm and 1.89mm, respectively. After calibration using the calibrated parameters the maximum, mean and standard deviation of the radius of spread were 4.22mm, 1.05mm and 0.587mm, respectively. The

improvement in the system is around 5.37(=5.64/1.05) times. Though the errors are dependent on the distance of the observation plane from base of the robot the ratio of errors or in other words the improvement ratio is independent of it. For small angular errors $\Delta A_1 \approx \tan(\Delta A_1) \approx \frac{\Delta E_1}{\beta}$ and $\Delta A_2 \approx \tan(\Delta A_2) \approx \frac{\Delta E_2}{\beta}$. Hence $\frac{\Delta A_1}{\Delta A_2} = \frac{\Delta E_1}{\Delta E_2}$, *i.e.,* the improvement ratio in aiming is independent of the aiming distance. This ratio also gives us the improvement in the angular errors of the end effector. Therefore the angular errors are reduced around five times with the ViCKi method.

4.6. Industrial Automation

The present calibration method can be automated to calibrate industrial robots. The feedback system studied in simulation can be extended to a real system. A camera, to record the position of the laser point, can be used to read the error in aiming at a desired point. These errors can be fed back to the system to compute the changes in joint values required to adjust the laser point closer to the required location. An initial set of joint values which aim at a fixed location were computed using the approximate parameters and using this feedback system it is very easy to obtain the data required for calibration making the process suitable for automation.

4.7. Limitations

The present method of calibration does not take into account other sources of error such as temperature, load variations, and elasticity of the arms. Having a general model which includes other effects apart from the inaccurate geometric model can also be calibrated using the procedure herein. The robot should be able to connect to the laser tool or at least be able to hold the laser tool rigidly if there is no provision for tool

interchange. Selection of a high quality laser whose light does not diverge significantly with the distance of projection is also important.

4.8. Conclusions

An accurate calibration procedure is developed for industrial robots. Most of the previous methods that calibrate a robot use the pose of the end-effector measured by some ancillary equipment. The accuracy of measurements of the end-effectors position and orientation is limited by the measuring equipment and its resolution. Other closed-loop methods use physical constraints such as linear, planar or other end point constraints. The methods that use a planar constraint on the end effector have limitations in obtaining accurate joint readings that satisfy the constraints due to the presence of contact forces. The methods using linear constraints have difficulty maintaining the end effector constraints while obtaining the readings. The single end point method has restrictions in obtaining the joint configuration readings for various poses due to the physically constrained environment. The proposed method uses a laser pointer tool on the robot's end-effector to aim at a fixed location on a distant object. By projecting the laser pointer onto a distant object, the resolution of observations is improved, increasing accuracy of measurements of the joint angles required for accurate calibration of the robot. The method is verified using both simulation and real experiments. It is also shown in simulation that the method can be automated by a feedback system. The calibration of the two industrial robots is accomplished and the accuracy of their parameters is improved. These accurate robots parameters give us a more accurate position of the PTU camera systems attached to them.

Chapter 5. RTU Calibration

5.1. Introduction

The goal of the dissertation is to find accurate 3D structure. The accuracy of the computed 3D points depends in the positions of the cameras with respect to global coordinate system. The cameras are mounted on the PTUs and PTU are picked up by the robots moving on the RTU. Hence the accuracy of the 3D structure depends on RTU model parameters accuracy. Also the present application involves multiple robots interacting with each other; it is necessary to determine their global location with respect to each other. The relative global locations of one robot from the other are usually computed by using inaccurate industrial design parameters. Hence a software calibration approach to identify the DH parameter values of the global position of one robot with respect to other is needed to increase our overall accuracy.

We use our ViCKi method to include the Rail Transport unit also and calibrate the global positioning of one robot with respect to other.

5.2. RTU and Staubli Models

We used the model of Staubli as described in section 3.1.1. The total number of independent calibration parameters (22) which describe the complete robot laser system for each robot are already determined using the industrial robot calibration (ViCKi) as described in Chapter 4. We used method 2 where the robot is translated on the RTU to get two different locations to aim the laser. One of the side effects of the calibration procedure was that we found the direction of the translation of the robot with respect to the robots base. This direction of translation gives us the direction of the RTU with

respect to the robot base coordinate systems. Thus we also know the two parameters for the RTU direction Φ_l and Φ_m.

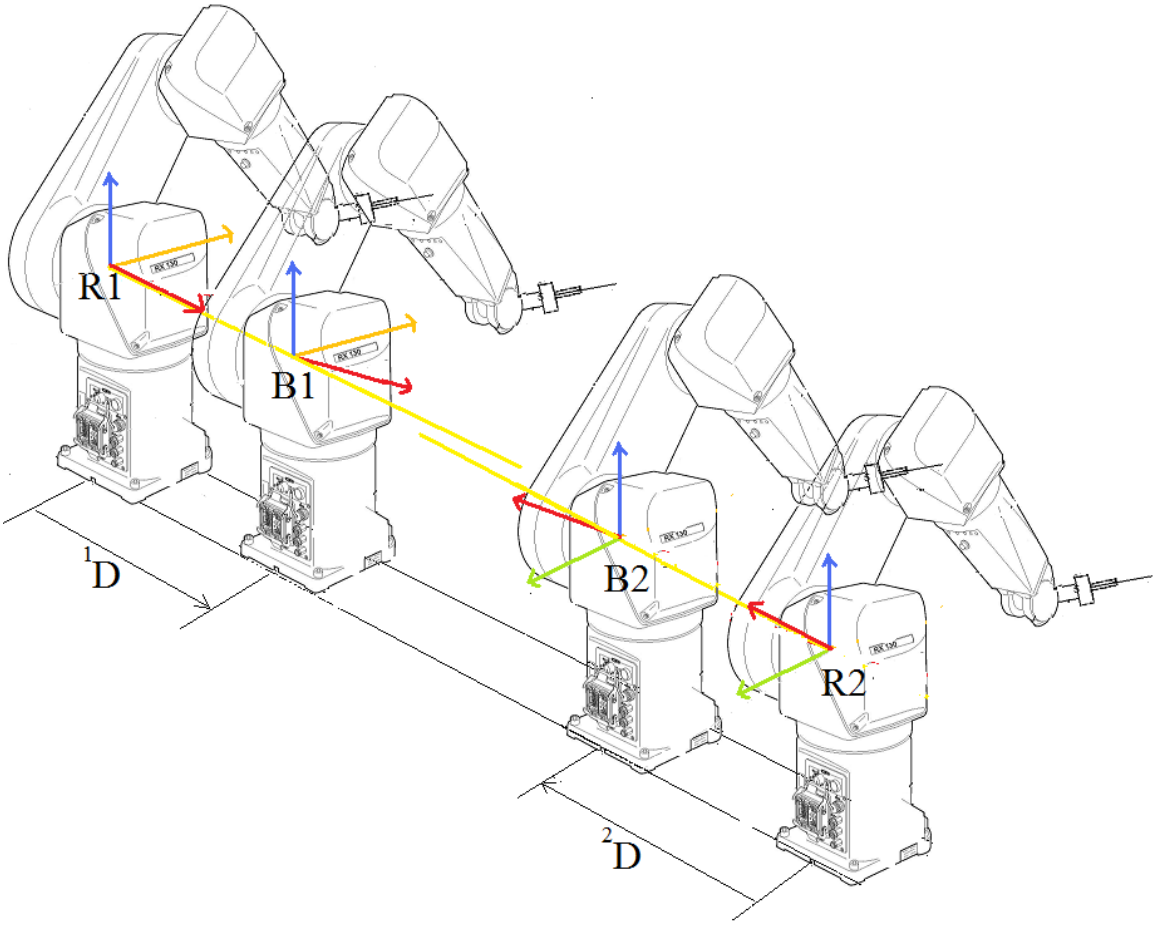

Figure 5.1. Coordinate systems for the RTU and robots.

Since we already know the two robot-laser systems parameters we can compute the transformation matrices ${}^{B1}T_{E1}$ and ${}^{B2}T_{E2}$ for any particular joint configurations.

The model of RTU and Staubli system is described section 3.2.1. The transformation from any RTU position of one robot to any RTU position of the other robot on the RTU is given by (3.4). All the required parameters are already calibrated except the transformation parameters Ψ_x, Ψ_y, Ψ_z and Ψ_α from one robot RTU base to other robot RTU base. The calibration procedure in this chapter describes the ViCKi extension

to determine these parameters. Then, the transformation matrix in (3.5) will be able to convert from one robot base coordinate system to other from any RTU positions.

5.3. Virtual Closed Kinematic Chain Calibration (ViCKi)

Section 4.3 describes a new method ViCKi to calibrate the industrial robots. We have used that method to calibrate both the robots. Thus all 22 parameters for the robot laser system and the two parameters for the RTU direction with respect to the robot base coordinate system are determined for each robot.

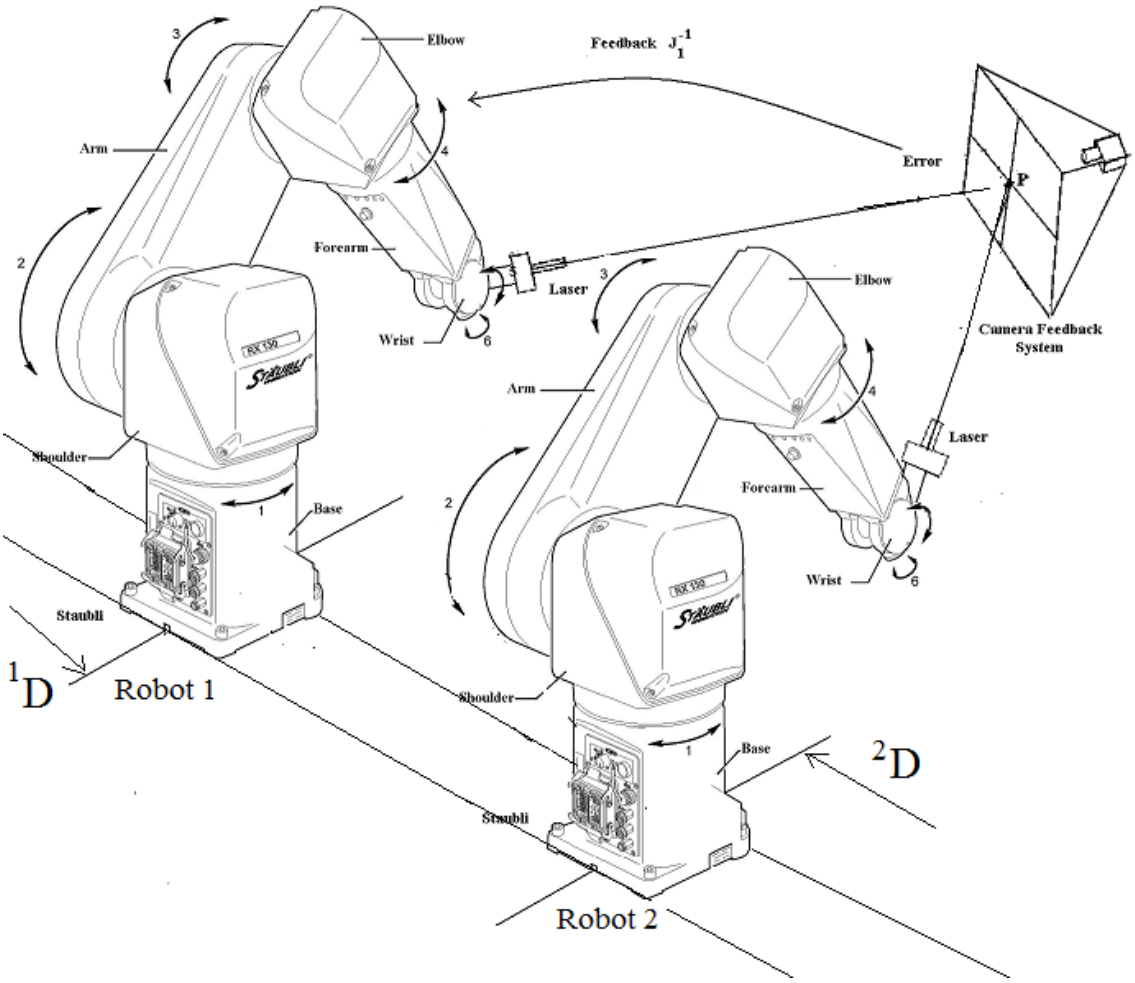

Figure 5.2. RTU is calibrated by aiming at same point by two robots' lasers.

The calibration is performed by aiming the laser pointer at an arbitrary point *P* on a distant object (usually a plane), then adjusting the joint values of the other robot to aim

its laser pointer at *P* as shown in Figure 5.2. The joint configurations and the RTU locations of both the robots are recorded. We are aiming at the same physical point by the two lasers on two robots. This effectively becomes a virtual closed loop for the two, 7 DOF systems. It is not necessary to know the coordinates of the point *P*. For each configuration, we use the constraint that the laser lines intersect. The parameters of the model are estimated by minimizing the sum of the normalized shortest distance of the two laser lines. Since the shortest distance error depends on the distance of the object plane from the end effectors (*i.e.*, length of the laser lines) it is normalized with the mean length of the laser lines. We denote the 22 parameter set for the first robot, laser system as ${}^{1}\Phi_{RL}$. The two parameters for the direction of RTU in first robots base frame as ${}^{1}\Phi_{lm}$. We use the notation ${}^{1}J_{i}$ for joint readings of first robot, ith reading. The transformation matrix for robot 1 with respect to its base RTU coordinate system, denoted as R1, is given by

$$ {}^{R1}_{i}T_{E1} = {}^{R1}T_{B1}\left({}^{1}\Phi_{l} \quad {}^{1}\Phi_{m} \quad {}^{1}D\right) {}^{B1}_{i}T_{E1}\left({}^{1}\Phi_{RL} \quad {}^{1}J_{i}\right) \qquad \textbf{(5.1)} $$

The transformation matrix for robot 2 with respect to its base RTU coordinate system (R2) is given by

$$ {}^{R2}_{i}T_{E2} = {}^{R2}T_{B2}\left({}^{2}\Phi_{l} \quad {}^{2}\Phi_{m} \quad {}^{2}D\right) {}^{B2}_{i}T_{E2}\left({}^{2}\Phi_{RL} \quad {}^{2}J_{i}\right) \qquad \textbf{(5.2)} $$

We use the unknown transformation matrix ${}^{R1}T_{R2}$ to compute the transformation matrix for robot 2 with respect to robot 1's base RTU coordinate system R1 frame as

$$ {}^{R1}_{i}T_{E2} = {}^{R1}T_{R2}\left(\Psi_{xyz\alpha}\right) {}^{R2}_{i}T_{E2} \qquad \textbf{(5.3)} $$

Using equation (5.1) the position and direction of the laser line for first robot are found in columns 4 and 3 respectively. Using equation (5.2) the position and direction of

the laser line for second robot are found in columns 4 and 3 respectively. The shortest distance of these two lines is determined and normalized with the mean length of the laser lines. This is the error, *i.e.*, the cost function for this reading given by

$$E_i\left({}^1\Phi_{RL}\ {}^1\Phi_l\ {}^1\Phi_m\ {}^1J_i\ {}^2\Phi_{RL}\ {}^2\Phi_l\ {}^2\Phi_m\ {}^2J_i\ \Psi_{xyz\alpha}\right) \quad \textbf{(5.4)}$$

The unknown parameter set $\Psi=\{\Psi_x\ \Psi_y\ \Psi_z\ \Psi_\alpha\}$ is determined by minimizing the total sum of the squares error

$$\Psi_{min}=MIN_\Psi\left(\sum_{i=1}^{N}E_i^2\right) \quad \textbf{(5.5)}$$

where Ψ_{min} is the required parameter set obtained by minimizing the total sum of squares of errors and N is the number of readings. We have developed a MATLAB routine which uses Levenberg-Marquardt (LM) [59] algorithm (MATLAB routine: lsqcurvefit) repeatedly by varying the solution obtained with random percentage values and using it as the initial solution for next repetition. The magnitude of the random percentage values is reduced with each repetition. Introducing randomness to initial values for LM algorithm reduced its chances of getting struck in local minima.

5.3.1. Feedback system

The feedback system described in section 4.3.1 , redirects the laser spot on a plane to a desired location. We used the same feedback system to accelerate the data acquisition process. Models of the RTU with two robot-laser systems are constructed in Simulink. The errors in aiming the laser at a fixed point due to errors in parameters are within ±10mm, which are well within the limits of the feedback system that can correct errors of ±500mm away from the desired point.

A camera detects the laser point and the error is used as feedback to change the robot joint angles so as to move the laser point to the desired point.

5.3.2. Procedure

The procedure is summarized in these steps

1) The second robot aims its laser at an arbitrary point on a distant plane.
2) Using the approximate parameter values of the robot laser systems and the 3D point, the first robot aims its laser at the same point.
3) The first robot's joint values are adjusted such that the centroids of both laser points are as close to each other as possible. The joint readings of both robots and their RTU positions are recorded.
4) Steps 1 to 3 are repeated large number of times (N = 500). For each reading, the robot's joint configurations and its RTU locations are varied to span their domain, *i.e*., to exercise all degrees of freedom, often known as the persistent excitation problem.
5) Using these data and a nonlinear minimization routine, compute the parameter values.

5.4. Experiments

5.4.1. Calibration with Precise Data in Simulation

Simulation of the calibration experiment was performed. Two robots were created in Simulink with known sets of parameters different from the industrial parameters (approximate DH parameters used by the robot manufacturer) on an RTU with known parameters. The experiment is conducted in simulation as discussed in section 5.3.2. The

actual known parameters for the robot laser systems are used and the parameters of the RTU are estimated and compared with the known values. The experiment was repeated multiple times with different joint configurations and RTU positions. The deviations of the parameters from the actual parameters were computed. Table 5.1 shows the results. Columns 2 and 3 show the true parameters used to obtain data and the initial parameters used in the minimization routine respectively. Columns 3 and 4 show the optimum solution and standard deviation from true parameters respectively.

Table 5.1. Calibration Results (with actual robots parameters)

Parameters	Actual	Initial values	Minimum Error	STD ($X10^{-4}$)
Ψx (mm)	5654	5650	5654.0001	4.6231
Ψy (mm)	3	0	2.9995	3.1274
Ψz (mm)	3	0	3.0003	2.2855
$\Psi \alpha$ (deg)	0.01	0	0.008	0.6141

5.4.2. Calibration with Noisy Data in Simulation

Simulation of the calibration experiment was performed. Two robots were created in Simulink with known sets of parameters different from the industrial parameters (approximate DH parameters used by the robot manufacturer) on an RTU with known parameters. The experiment is conducted in simulation as discussed in section 5.3.2. The robots' joints have finite resolutions as shown in Table 4.2. Hence to make the simulation more realistic noise is added to the joint values with magnitude of the maximum resolution. These joint configurations were used to calibrate the RTU. The experiment was repeated multiple times with different joint configurations and RTU positions. The deviations of the parameters from the actual parameters were computed. Table 5.2 shows the results of the calibration with noisy data.

Columns 2 and 3 show the true parameters used to obtain data and the initial parameters used in the minimization routine respectively. Columns 3 and 4 show the optimum solution and standard deviation from true parameters respectively. The standard deviation of the solution was small enough (10^{-3}) to justify that the procedure produces a usable result.

Table 5.2. Calibration Results (Noisy Simulation)

Parameters	Actual	Initial values	Minimum Error	STD ($X10^{-3}$)
Ψx (mm)	5654	5650	5654.0043	3.2561
Ψy (mm)	3	0	2.9978	2.8954
Ψz (mm)	3	0	3.0016	2.5475
Ψα (deg)	0.01	0	0.014	0.3271

5.4.3. Calibration of RTU with two Staubli Robots

The calibration experiment was performed on the real RTU with two Staubli RX130 robots. The experiment was repeated multiple times with different joint configurations and RTU positions. The mean and standard deviation of the calibration parameters were computed.

Table 5.3 shows the results of the calibration of the real system. Column 2 shows the initial parameters used in the minimization routine. Column 3 shows the optimum solution. Column 4 and 5 show the mean and standard deviation of the parameters and from multiple trials. The standard deviation of the solution was also small (10^{-3} mm/deg) indicating the stability of the procedure.

Table 5.3. Calibration Results

Parameters	Initial values	Minimum Error	Mean	STD ($X10^{-3}$)
Ψx (mm)	5654	5654.8610	5654.8604	3.4381
Ψy (mm)	0	-3.3288	-3.3272	1.5725
Ψz (mm)	0	1.4338	1.4321	1.3431
Ψα (deg)	0	0.0134	0.0135	0.4352

5.4.4. Accuracy of RTU Transformation

To compare the accuracy of computed parameters with the approximate parameters, the laser was aimed at a fixed 3D point using various joint configurations and RTU positions of first robot. Then the calibration parameters are used to transform the laser lines into the other robot's base coordinate system, and, these laser lines are projected on to a plane passing through the transformed 3D point. Since these are the same laser lines, they should intersect at the transformed 3D point, but due to the errors in the parameters this will not be the case. The errors of the projection are compared by using the laser lines before transforming, and after transforming using un-calibrated, calibrated parameters to find the improvement. The larger the error in the parameters, the greater will be the scattering in the projected points.

The maximum, mean and standard deviation of the radius of spread before transformation due to errors in robots parameters were 4.34mm, 1.12mm and 0.573mm respectively. These points were transformed to the other robots coordinate system using calibrated and un-calibrated RTU parameters for the transformation. The errors due to the inaccuracy of the RTU parameters are shown in Figure 5.3.

The maximum, mean and standard deviation of the radius of spread were 16.07mm, 5.22mm and 2.72mm respectively using uncalibrated RTU parameters. Transforming using calibrated parameters the maximum, mean and standard deviation of the radius of spread were reduced to 4.43mm, 1.78mm and 0.817mm respectively which are comparable to the errors associated with a single calibrated robot, *i.e*., very little error is added to the system as a result of adding the RTU into the system.

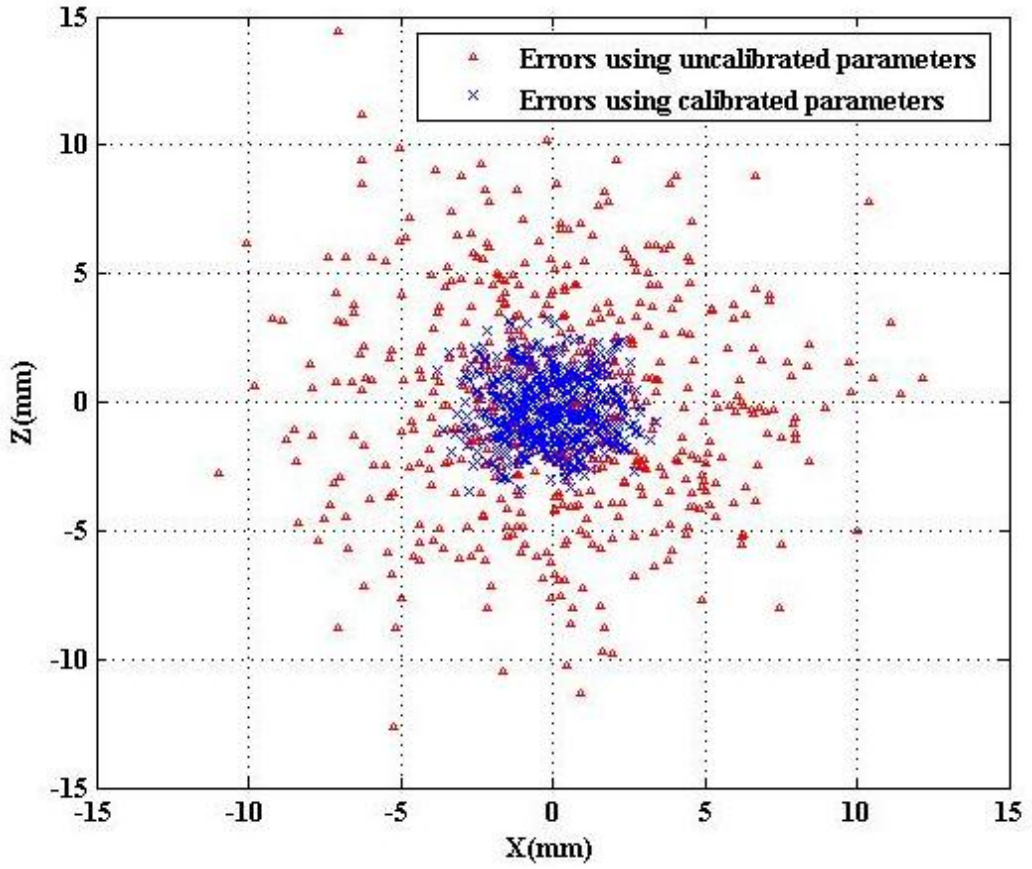

Figure 5.3. Errors in aiming the laser at fixed points after transformation using un-calibrated and calibrated parameters.

5.5. Limitations

The present method of calibration does not take into account other sources of error such as temperature, load variations, and elasticity of the arms and backlash. Having a general model which includes other effects apart from the inaccurate geometric model can also be calibrated using the procedure herein. Selection of a good laser whose light does not diverge much with the distance of projection is important.

5.6. Conclusions

The calibration procedure developed in previous chapter is extended to include the RTU also. The two robots on the RTU are calibrated independently in previous chapter and their relative global position and orientation accuracy is improved in this

chapter. The method is verified using both simulation and real experiments. It is also shown in simulation that the method can be automated by a feedback system.

Chapter 6. Pan-Tilt Camera Calibration

6.1. Introduction

The previous chapters calibrated the robots and the RTU giving us more accurate parameters for their models. Using those parameters the location and orientation of the base coordinate system of the Biclops is found accurately. The problem still remains that the PTU parameters are not accurate. This chapter particularly focuses on calibration of the PTU and the camera systems. This relation between the 3D coordinates of points in the workspace and their 2D image locations can be computed by using those parameters. The calibration includes internal and external parameters of the camera. Calibration of internal parameters finds the relationship between image coordinates and ray directions in the camera coordinate system. This relationship is described by the perspective projection of the ideal pinhole camera model. The parameters which need to be determined are the focal length, the principal point (image center pixel) and scale factors in the x and y directions. It is possible to include compensation for lens distortion when a more accurate model is desired. Calibration of external parameters involves finding the position and orientation of the camera in some world coordinate system (here the base of the PTU coordinates). External camera calibration is important in stereo vision where one needs to find the relative position of the coordinate systems of the two cameras and it is also important in hand-eye coordination in robots. Introducing pan-tilt motion to the cameras makes the external calibration dependent on the values of the pan and tilt angles. The calibration problem is to identify the values of the internal and external parameters of each camera in a stereo pair so that they can work together effectively, e.g., to determine 3D points of interest through triangulation.

6.2. Biclops Model

The Biclops is modelled as described in section 3.4.1. The parameters for the Biclops are not accurate. These parameters need to be calibrated. The calibration parameters for the camera can be divided into two groups. One group is the internal parameters which do not depend on the position or orientation of the camera. The other group of parameters is the one that affects the accuracy of the position and orientation of the camera with respect to some coordinate system, i.*e.,* the robot end effector in this case.

6.2.1. External Parameters

The transformation matrix which converts points from the base of the robot to a coordinate frame fixed on the camera is called the external transformation matrix. The external transformation matrix gives the location and orientation of the camera in the base coordinate frame of the robot (knowing the TCF position and orientation). The parameters of this external transformation matrix described in section 3.4.1 denoted by (t_x, t_y, t_z, θ_x, θ_y, θ_z, θ_1, α_1, a_1, d_1, θ_2, α_2, a_2, d_2), are the external parameters.

6.2.2. Internal Parameters

The transformation matrix that converts points from the coordinate frame fixed on the camera to the 2D image coordinates is given by the following expression

$$I_{T_3} = \begin{bmatrix} fp_x & 0 & u_0 & 0 \\ 0 & fp_y & v_0 & 0 \\ 0 & 0 & 1 & f \end{bmatrix} \quad \textbf{(6.1)}$$

where f is the focal length, (u_0, v_0) is the location of the point where the axis of the camera intersects the image plane, *i.e.*, the principal point. Parameters p_x, p_y are the number of pixels per mm (inverse scale factors) in the x and y directions, respectively, of

the image. The parameters f, u_0, v_0, p_x and p_y are the internal parameters. This model does not include the lens distortion and the pixel skew, though the model can be easily extended to include them.

6.2.3. Calibration Matrix

The matrix that transforms points from the base coordinate frame of the robot to the image coordinate frame is defined to be the calibration matrix, given by ${}^{I}T_{B}$.

$$ {}^{I}T_{B} = {}^{I}T_{3}\left({}^{2}T_{3}\right)^{-1}\left({}^{1}T_{2}\right)^{-1}\left({}^{0}T_{1}\right)^{-1}\left({}^{E}T_{0}\right)^{-1}\left({}^{B}T_{E}\right)^{-1} \quad \textbf{(6.2)} $$

This matrix is clearly dependent on the values of the pan and tilt axis angles because of ${}^{0}T_{1}$ and ${}^{1}T_{2}$. ${}^{I}T_{B}$ can be computed for any pan and tilt angles of a PTU if all the internal and external parameters are known. Given a 3D point X_B in the base coordinate frame of the robot, the calibration matrix is used to compute the location of the point in the image by $W = {}^{I}T_{B}\, X_B$. Vector $W = \{w_1\ w_2\ w_3\}^T$ contains the homogenous coordinates of the image point U(u, v), computed by $u = w_1/w_3$, $v = w_2/w_3$.

6.3. Calibration

Calibration is the process of determining the values of all the parameters $\{ t_x, t_y, t_z, \theta_x, \theta_y, \theta_z, \theta_1, \alpha_1, a_1, d_1, \theta_2, \alpha_2, a_2, d_2 f, u_0, v_0, p_x, p_y \}$ of the model Φ. Given a point in the workspace, the corresponding image point can be computed using the values in Φ. The error between the computed and the observed points, as registered in the image plane, is called the projection error, and is the result of error in the estimation of the model parameters. Thus, choosing model parameter values that minimize the mean square projection error is the goal of the calibration procedure. This is described mathematically by

$$\Phi_m = \underset{\Phi}{Min} \sum_{i=1}^{n} (U_i(\Phi, X_i, \theta_{pi}, \theta_{ti}, T_i) - \overline{U_i})^2 \quad \textbf{(6.3)}$$

where $\overline{U_i}$ is the observed image point and $U_i(\Phi, X_i, \theta_{pi}, \theta_{ti}, T_i)$ is the computed image point of the projected 3D point X_i in the base coordinate frame of the robot when the position of the end effector is T_i and pan and tilt values are θ_{pi}, and θ_{ti} respectively. In (6.3), T_i= ($x, y, z, \alpha, \beta, \gamma$ } is the set of translation and rotation values of the end effector, parameter 'n' is the total number of 3D points observed, and Φ_m is the required parameter set that minimizes the mean square projection error.

To calibrate the model, the usual approach is to use a large set of 3D points and their corresponding 2D image points with respect to some pan, tilt angles and end effector position and orientation. The mean square of the projection error is minimized using Levenberg-Marquardt [59] nonlinear minimization. This procedure requires a known set of 3D points (X_i) in the workspace which are difficult to measure and time consuming to collect. The innovation of the proposed approach is that the calibration procedure herein is altered to use a single 3D point (X) and images of the point from various camera positions and orientations are taken. The modified calibration is represented by equation

$$\Phi_m = \underset{\Phi}{Min} \sum_{i=1}^{n} (U_i(\Phi, X, \theta_{pi}, \theta_{ti}, T_i) - \overline{U_i})^2 \quad \textbf{(6.4)}$$

The second distinguishing characteristic of the calibration procedure is to replace the known 3D point with an unknown but stationary 3D point in the workspace. The coordinates of the unknown 3D point are added to the set of calibration parameters to get a new set $\{\Phi, X\}$. The calibration procedure itself will determine the 3D location of the point (X). The final calibration is represented by equation

$$\{\Phi,X\}_m = \underset{\{\Phi,X\}}{Min} \sum_{i=1}^{n} (U_i(\{\Phi,X\},\theta_{pi},\theta_{ti},T_i)-\bar{U}_i)^2 \quad \textbf{(6.5)}$$

where X is the unknown stationary 3D point in the base coordinate frame of the robot. Using a single stationary 3D point does not affect the excitation of all the degrees of freedom of calibration as we still have the pan, tilt and the end effector location and orientation that can be varied extensively by moving the robot that carries Biclops.

6.4. Experiments

6.4.1. Laser Pointer Tool

To determine that a calibration is accurate, an independent measurement must be made. To enable this, a laser pointer tool that can be attached to the robot wrist has been fabricated. It consists of a laser pointer (with pivot to adjust orientation) connected to a tool base. The robot picks up this tool by connecting to the tool base, as shown in Figure 6.1. The robot can then aim the laser tool at any location in the workspace. This approach provides a simple but effective check on the calibration procedure. After the calibration procedure, the location of the calibration point can be computed and the laser pointing tool can overlay its laser point on the calibration point, thus verifying the results of the procedure.

Figure 6.1. Staubli RX-130 robot carrying a laser pointer tool.

6.4.2. Experimental Setup

The workcell consists of two Staubli RX-130 robots, one carrying Biclops and the other carrying the laser tool, as shown in Figure 6.2. These robots are on a robot transport unit (RTU) and can be moved along the track. An accurate calibration of the RTU system has been done which gives the transformation from the base of one robot to the other, measured experimentally to be within 1mm. The laser pointer aims at an arbitrary stationary location in the workspace. The robot carrying Biclops moves through a sequence of locations to exercise all of the degrees of freedom on the camera and PTUs. At each programmed location, the cameras on Biclops capture an image.

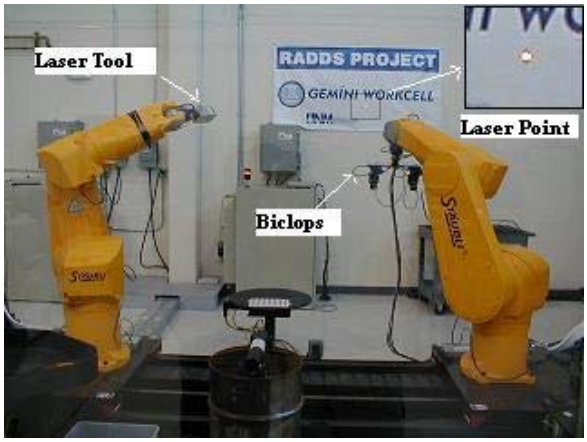

Figure 6.2. Workcell with two Staubli RX-130 robots on the RTU and laser point projected onto the wall .

6.4.3. Image Acquisition and Processing

The cameras on Biclops can be programmed for specific values of exposure time. Since the laser spot is quite bright relative to the other parts of the scene, a short exposure time ensures capturing only that spot. We assume that there are no other high intensity light sources in the workspace. This makes it very easy to process the images quickly to determine the location of the laser point in the images.

6.4.4. Procedure

The number of constraints required by the Levenberg-Marquardt algorithm to converge to a good model is usually five times the number of parameters in the model. Consequently, we used 60, 2D image points, which is equivalent to 120 constraints. Since the positions of the robot and values of the pan-tilt mechanism are under computer control, it is easy to ensure that each variable is exercised from minimum to maximum values. This persistent excitation prevents ill-conditioned matrices that result in poor models. The algorithm for the calibration procedure is as follows:

1) A laser pointer is aimed at an arbitrary location.
2) Random pan, tilt and TCF position and orientations are computed. This set is reduced to 60 positions, where the laser point is in the field of view of both cameras.
3) Biclops is moved by the robot to each of the set of positions and images of the laser point are captured.
4) These images are processed to determine the 2D coordinates of the laser point.
5) Once the data are collected for all positions the minimization routine (Levenberg-Marquardt) is used to determine the calibration parameters.

6.4.5. Results

Calibration of the pan-tilt cameras on Biclops requires only one unknown point to determine the values of the model parameters. Ideally, the same values of the model parameters should be obtained regardless of the location of the unknown point. In this

section we repeat the calibration procedure of using different unknown points to measure the difference between the models obtained.

Table 6.1 shows the 19 parameter values for each camera along with the obtained coordinates of the unknown stationary 3D point in two different trials using different unknown 3D points.

Table 6.1. Calibration Parameters

	Trial I Point (500, -3198, 1368)		Trial II Point (700, -2148, 1254)	
Parameters	Left	Right	Left	Right
tx (mm)	111.4615	-80.7387	111.3205	-80.0354
ty (mm)	-60.2324	122.6854	-60.35	121.9943
tz (mm)	221.7315	230.9312	221.37	230.2173
θx (deg)	0.0592	0.1125	0.0761	0.0925
θy (deg)	-0.0854	-0.2831	-0.0732	-0.2478
θz (deg)	45.7232	45.5747	45.9421	45.0732
α1 (deg)	90.2854	90.3832	90.5531	90.2987
a1 (mm)	-8.9323	-9.5688	-9.0615	-9.8233
θ1 (deg)	88.1076	86.7735	87.9864	86.1378
d1 (mm)	9.8212	10.1115	9.5616	10.4156
α2 (deg)	89.9718	90.2965	90.0515	90.3193
a2 (mm)	54.1533	44.8755	54.3460	44.8043
θ2 (deg)	-90.0723	-90.6572	-90.1289	-90.0667
d2 (mm)	42.7024	27.9636	42.5957	27.5537
f (mm)	12.2565	11.9414	12.17111	11.9871
u0 (pxls)	322.4034	333.5779	323.10	334.1599
v0 (pxls)	260.584	264.3231	259.6133	265.1217
px (pxls /mm)	101.0467	101.4207	100.9734	101.8278
py (pxls /mm)	101.0243	100.2475	100.9958	100.0450
3D-X (mm)	499.297	498.265	700.325	699.654
3D-Y (mm)	-3197.035	-3196.214	-2148.014	-2147.092
3D-Z (mm)	1368.093	1367.215	1254.032	1253.256

An initial estimate of the parameters was determined from the "as designed" dimensions of the Biclops pan-tilt system. The Levenberg-Marquardt nonlinear minimization routine was used on the calibration data to compute values for the parameters that minimize the projection error. The model parameter values obtained from the two different unknown 3D points were within acceptable limits, as shown in Table 6.1. For Trial 1, the "unknown" point was measured at (500, -3198, 1368). The computed values for the left and right cameras, the last three parameters in the model, show very small errors. Similarly, for Trial II, where the "unknown" point was measured to be (700, -2148, 1254), the last three parameters in the model also show small error, certainly acceptable from the standpoint of grasping an object.

The calibration process was repeated in a similar fashion 90 times to test the accuracy of the calibration. The histogram of the projection error for the set of points was plotted as shown in Figure 6.3 and Figure 6.4. The projection error had range [0.5 to 2.8] pixels and mean close to 1 pixel.

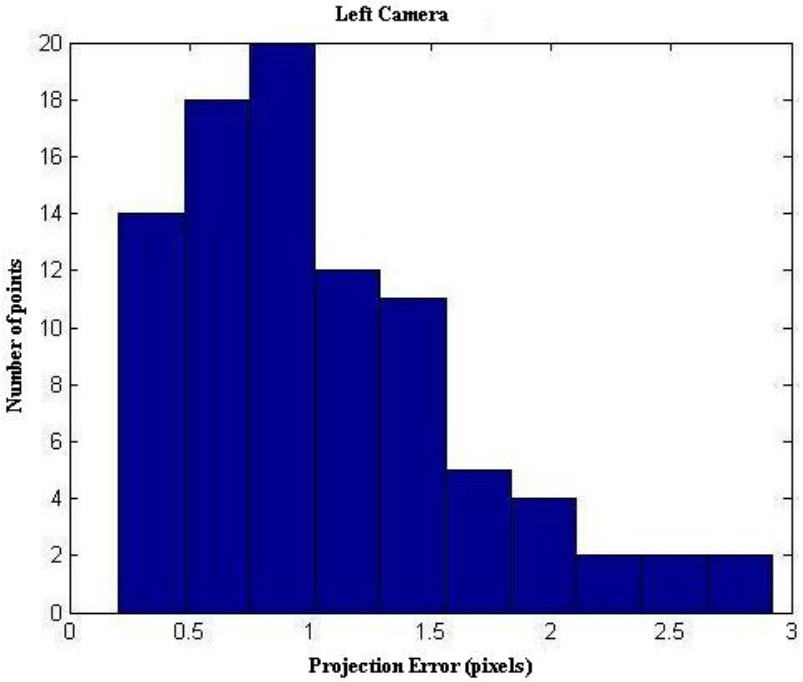

Figure 6.3. Histogram of the Projection errors for left camera.

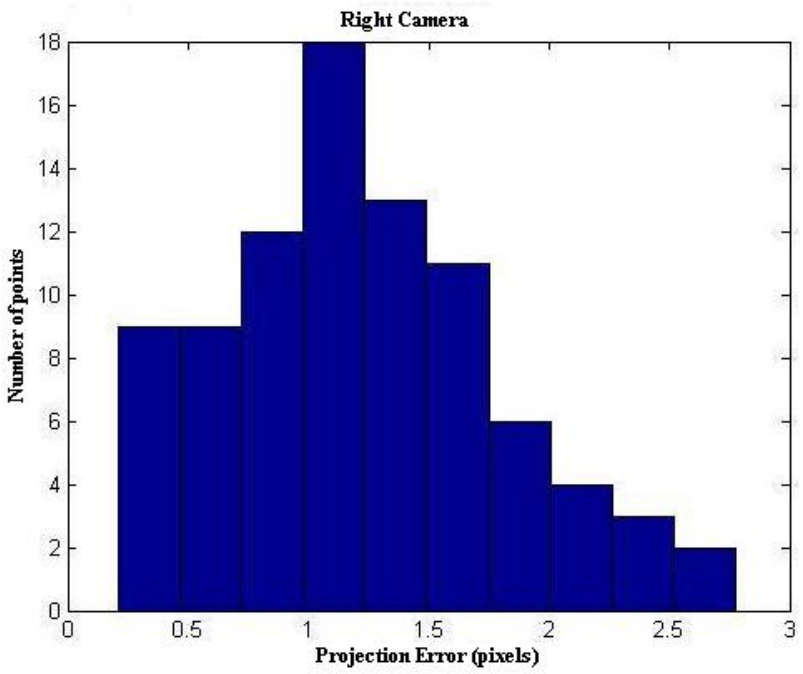

Figure 6.4. Histogram of the Projection errors for right camera.

Using the model computed for each camera and triangulating to a point 1000 mm away, a 1 pixel error creates an error of 3 mm. The time required for the complete calibration procedure is limited by

the speed of the motion of the robot and PTUs. The procedure takes about three minutes to run. This compares favorably with the 20 to 30 minutes required for our previous calibration procedure that used a multi-point calibration grid. We are unaware of published calibration times for alternative approaches. Calibration using a grid requires more complex image processing and human involvement to verify the correctness of the corresponding image point locations; these problems were eliminated completely in the present method.

6.5. Conclusions

A novel method is developed to calibrate a pair of cameras mounted on PTUs where a pair of cameras analyzes images from a single fixed point in space. The correspondence problem has been eliminated completely, and image processing has been simplified to finding the only bright dot in an image.

A complete model without any assumptions about the PTU geometry is considered. The present method does not require either an extensive set (or any, for that matter) of known calibration points in the workspace or an expensive routine to make the image features correspondences.

A single unknown stationary 3D point in the workspace, designated by aiming a laser pointer at some surface in the workspace, is sufficient to derive the vision system model parameters. The values of robot location, pan, and tilt can be generated automatically using inverse kinematics (with initial approximation of the parameters) so that the algorithm can run unattended whenever needed.

Chapter 7. IDI-Points

7.1. Introduction

In this chapter we introduce the various image processing functions to find the low level features in images like the edges, corners and lines. A new image feature called Indirectly Determined Intersection Point (IDI) is presented. The various low level image processing needed to locate these IDI points in the image is presented. The geometrical and image properties of IDI points are presented. It is also shown how these points are less sensitive to image noise as compared to the traditional corners in an image. Various experiments are presented to prove this concept.

7.2. Sum of Squared Difference

It is often required to match two image regions to see how similar they look. A measure of similarity is defined where each pixel is compared with its corresponding one to see how different it is. This method is called Sum of Squared Difference (SSD). For an image patch of size (2W+1, 2H+1) at (u, v) and at the patch shifted to (u+x, v+y) is defined as

$$S(x,y)=\sum_{u=-W}^{W}\sum_{-H}^{H}\left(I(u,v)-I(u+x,v+y)\right)^2 \qquad \textbf{(7.1)}$$

7.3. Linear Filtering

Filtering is an operation on the digital image producing a new image. Consider a digital image, $I(x,y)\forall x\in[0,W), y\in[0,H)$, now applying a function on the image results in another image $I_G=G\left(I(x,y)\right)$ the operation 'G' is called the filter. A filter can be usually any kind of operation resulting in any size of the image (may not be same as the original). A linear filter is a filter which uses only linear operations on the image thus

the filter of a linear combination of images will be equal to the same linear combination of the individual images filtered first. Let I_1 and I_2 be two images and G be a linear transform. Then for any two constants a and b, the linear filter 'G' satisfies the following equation

$$G\left(aI_1\left(x,y\right)+bI_2\left(x,y\right)\right)=aG\left(I_1\left(x,y\right)\right)+bG\left(I_2\left(x,y\right)\right) \quad \textbf{(7.2)}$$

7.4. Convolution

Convolution is a linear filter. The resulting image pixels are a linear combination of the image pixels in its neighborhood. An array of the weights is chosen to apply on the neighborhood pixels of each pixel. This array of weights is called as the kernel and they are kept same throughout the operation. The resulting image is a new array with a weighted sum of the neighborhood pixel values. Convolution is shift-invariant - meaning that the value of the output depends on the pattern in an image neighborhood, rather than the position of the neighborhood. The convolution operation can be represented as

$$I_c\left(x,y\right)=K*I=\sum_{i=-a}^{a}\sum_{j=-b}^{b}K\left(i,j\right)I\left(x-i,y-j\right) \quad \textbf{(7.3)}$$

where K is the kernel of size (2a+1,2b+1). The edges of the image are handled specially by either zero padding or mirroring the inner pixels, or ignoring the edge pixels all together to output an image of size (W-2a, H-2b).

7.5. Smoothing or Blurring (Low Pass Filter)

A digital image is usually accompanied with noise due to either the imaging device itself or the image capturing method of the device. This noise needs to be removed by a smoothing operation. Smoothing or Blurring uses a smoothing kernel and convolves the image with it. It is also referred to as a low pass filter since effectively it filters out the

high frequency noise and retains the low frequencies. Simple blurring uses a kernel with all same weights. Gaussian blurring uses Gaussian kernel with the weights given by a Gaussian distribution

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \textbf{(7.4)}$$

where σ is the standard deviation of the distribution. The kernel size depends on the standard deviation. The amount of blurring is depends on the kernel size and values.

7.6. Edge Detection (High Pass Filter)

Assuming that objects in the scene are approximately uniform in intensity the edges in images are the pixels that separate one area from another. They are defined by the pixels where there are discontinuities. It is also referred to as high pass filter since it only retains the higher frequency edges and filters out the low frequencies. Since noise can be present in the images and can pass through the edge detection process, it is essential to remove the noise in the image before edge detection. Images are usually almost always smoothed using a Gaussian to reduce the noise.

There are many methods for edge detection, but most of them can be grouped into two categories, derivative based and zero-crossing based. The derivative based methods detect edges by first computing a measure of edge strength, usually a first-order derivative expression such as the gradient magnitude, and then searching for local maxima of the gradient magnitude in the gradient direction. The zero crossing based methods search for zero crossings in a second order derivative based expressions of the image. The expressions may consist of Laplacian or a non linear differential expression.

The gradient of the image at each pixel is a 2D vector with the components given by the derivatives in the horizontal and vertical directions. The gradient vector is a vector

along which there is a maximum change in intensity. Its magnitude is the change in that direction.

7.6.1. Roberts

Roberts edge detection is one of the earliest edge detection algorithms. The edges are computed by using two diagonal gradient approximations. The operation is represented as two convolutions with 2x2 kernels.

$$D_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \textbf{(7.5)}$$

$$D_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \textbf{(7.6)}$$

And the edge strength is computed as the sum of the absolute values of the gradients, the edges correspond to the local maximum of the edge strength.

$$E = |D_1| + |D_2| \quad \textbf{(7.7)}$$

7.6.2. Prewitt Edge Detection

Prewitt edge detection [67] uses a set of (in general 8) convolution kernels each of which is sensitive to edges in a different orientation, and convolves the image with them. For each pixel the local edge gradient magnitude is estimated with the maximum response of all 8 kernels at this pixel location. The orientation is given by the kernel which gave maximum response. This edge detection method is also called edge template matching, because a set of edge templates is matched to the image, each representing an edge in a certain orientation. The kernel for positive x direction is given by

$$G_1 = \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix} \quad \textbf{(7.8)}$$

7.6.3. Sobel

The Sobel operator is a discrete differentiation operator, approximating the gradient of the image. It uses two 3×3 kernels which are convolved with the original image to calculate approximations of the horizontal (G_x) and vertical (G_y) derivatives.

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \quad \textbf{(7.9)}$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \quad \textbf{(7.10)}$$

Sobel edge detection is based on convolving the image with these kernels. The kernels are small, separable, and integer valued and is therefore relatively inexpensive in terms of computations. The kernels are composed of the Gaussian and the gradient terms as can be seen in the separation in x and y direction. The edge strength and the direction can be computed as

$$G = \sqrt{G_x^{\,2} + G_y^{\,2}} \quad \textbf{(7.11)}$$

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right) \quad \textbf{(7.12)}$$

7.6.4. Canny

The Canny method [68] finds edges by looking for local maxima of the gradient of image. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges. Non maximum gradient intensity is suppressed in the direction normal to the edge direction.

7.7. Line Detection

Edges in the images indicate the discontinuities which often correspond to the objects' boundaries. For many tasks, it becomes a fundamental problem to detect object boundaries which are often straight lines. Edge detection is the preliminary process which detects the discontinuous pixels. But due to noise we often see more edges than necessary or sometimes the edge pixels can be missing in between line segments. Line detection is the process of finding and fitting lines to the edge data.

7.7.1. Hough Transforms

Hough transforms [70] were developed by Paul Hough to detect straight lines in edge images. It is a voting method where each edge pixel votes the various possible lines that pass through the pixel point. It is difficult to consider all infinite lines that pass through a point so the line parameter space is discretized. Consider a straight line with can be represented by

$$Y = mX + c \qquad \textbf{(7.13)}$$

where m is the slope and c is the y-intercept. Every line can be represented by a point (m, c) in the parameter space. So each edge in the image used to vote all the

possible lines through point. Since ‘m’ ranges from 0 to ∞ for horizontal to vertical lines hence it is difficult to digitize this parameter space. To avoid this problem Duda and Hart [71] introduced an alternate representation of the line as

$$X\cos(\theta) + Y\sin(\theta) = r \quad \textbf{(7.14)}$$

where r is the perpendicular distance from the origin to the line and θ is the angle the perpendicular line makes with x-axis. Parameters space (r, θ) is digitized where θ ranges from 0 to 180 and r depends on the image size. Lines are found with Hough transform by first scanning the edge image and determining all the possible lines that pass through the pixel. These lines are points in the parameter space which are voted accordingly. After all the edges finish voting the parameter space image is then scanned for peaks which should correspond to the lines in the original image. Each peak is used and its neighborhood is suppressed to avoid multiple lines corresponding to the same line in the original image but with sight change. O'Gorman and Clowes suggested that the local gradient of the image intensity can be used to control the accumulation process. The gradient direction is often found as a side effect when computing the gradient intensity magnitude. If a given image point (*x,y*) lies on a line, then the local direction of the gradient gives the θ parameter corresponding to the line, and *r* can be computed using (7.14). Since the estimated gradient direction is less accurate, a range of θ (±25°) is used and r for corresponding θ is computed and voted. This reduces the computation time by reducing the number of useless votes, thus enhancing the visibility of the spikes corresponding to real lines in the image.

Stephens [72] developed The Probabilistic Hough Transform where a model of feature error characteristics is proposed combining normally distributed measurement

errors with uniformly distributed correspondence errors. The Probabilistic Hough Transform H(y) is defined as the log of the probability density function (PDF) of the output parameters, given all available input features given by

$$H\left(\vec{y}\right)=\ln\left(f\left(\vec{y}\mid\vec{x}_1,\vec{x}_2\cdots\vec{x}_n\right)\right) \quad \textbf{(7.15)}$$

where $\vec{x}_i\left(x,y\right)$ is the input feature, *i.e.,* is the edge and $\vec{y}_i\left(r,\alpha\right)$ is a specific point in Hough space.

Using Bayes theorem we get

$$H\left(\vec{y}\right)=\sum_{i=1}^{n}\ln\left(f\left(\vec{x}_i\mid\vec{y}\right)\right)+\ln\left(f_0\right)+C \quad \textbf{(7.16)}$$

The conditional individual feature probability is calculated using

$$f\left(\vec{x}_i\mid\vec{y}\right)=\frac{\left(1-p\right)}{2\pi r_{\max}}+\frac{p}{2\pi\sigma\rho}\exp\left(-\frac{\varepsilon^2}{2\sigma^2}-\frac{\phi^2}{2\rho^2}\right) \quad \textbf{(7.17)}$$

where $\varepsilon=X\cos(\theta)+Y\sin(\theta)-r$ (the lateral error), and $\phi=\theta-\alpha$ (the orientation error) and p is the probability of individual point being on a line and σ and ρ are the standard deviations of the lateral and orientation errors. Thus the Hough accumulation value is computed by this individual conditional probability function. The peaks in the Hough transform are used to locate the lines in an image. These peaks are used to search on the corresponding line in the image to find the line segments. The different parameters are the tolerance to the gaps in segments and the minimum length of the segments that are required.

7.7.2. Radon Transforms (Fan Transform)

The Radon transform is a projection of a two-dimensional function *I(x,y)* into a set of line integrals. It is equivalent to rotating the image by the given angle and taking its

projection vertically on to the new x-axis for all valid regions of the image. The Radon transform at a particular angle (θ) is a line integral of the function perpendicular to that angle given by

$$R_\theta(x') = \int_{-\infty}^{\infty} I\left(x'\cos(\theta) - y'\sin(\theta), x'\sin(\theta) + y'\cos(\theta)\right)dy' \quad \textbf{(7.18)}$$

where

$$\begin{Bmatrix} x' \\ y' \end{Bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix} \quad \textbf{(7.19)}$$

Each projection angle produces a 1D vector. The Radon projections for angles (0-180) form a Radon image. One property of the Radon transform is that the peaks in the Radon image correspond to the lines in the original image. The Radon transformed image is thresholded to locate peaks and the same kind of local suppression of non maxima is done as in Hough transforms to locate the individual lines.

7.8. Corner Detection

For various algorithms it is important to locate point features in the images which are also referred to as corners which are small, two dimensional points of interest. The most unique points in the images are the corners. A corner is defined as a point where two or more edges meet, in other words corners are points where there are more than one image gradient. These often arise as the result of geometric discontinuities, such as the corners of real world objects, but they may also arise from small patches of texture. Most algorithms are capable of detecting both kinds of points of interest. The corners should have variation in two directions to locate them. The corner detection algorithms proceed to first detect corner strength at each image point. The corner strength is then thresholded

to obtain initial locations of the corners. Then post processing is usually done to localize these points to sub pixel accuracy and while rejecting the non maximum local corners.

7.8.1. Edge Contour Based

At corners of regions, the edge boundary changes direction rapidly. Several methods were developed which segment the image first and locate the edge chains and analyze their properties. Some techniques involve parameterizing edges with cubic splines. Langridge [73] and Medioni [74] determine edges by looking for fast changes in the spline first derivative. They locate points where large deviations of the spline occur from the control point. The Curvature Scale Space [75] detector computes the radius of curvature of the contour and detects maxima of curvature where there is a large difference between maxima and the closest minima. These methods rely on segmentation and contour generation method. Haralick and Shapiro [76] detect edgels and use these as candidate points for corners. They fit a line to the nearby edges of a point and look for its intersection with a circle around the point. Corners are found by thresholding the image gradient directions, since the edge direction is changes rapidly near them. They suggest using either a straight line or a cubic polynomial for the line fitting. Cooper [77] uses the idea that along an edge the image looks similar. They first finds edges and their directions then take a patch on an edge and compare it to the patches on either side in the direction of the local contour to detect self similarity. Corners are detected where large deviation occurs. Kitchen and Rosenfeld [78] look for rapid changes in the edge direction by measuring the derivative of the gradient direction along an edge, multiplied by the magnitude of the gradient. The resulting corner response is given by

$$C = \frac{g_{xx} g_y^2 + g_{yy} g_x^2 - 2 g_{xx} g_y}{g_x^2 g_y^2} \quad \textbf{(7.20)}$$

where g is the image or a 2D polynomial fitted locally to the image. And the subscript indicates the derivative in that direction. A quadratic polynomial gave good results.

7.8.2. Wang and Brady

Wang and Brady [79] propose a detector which searches for large total surface curvature on an image edge. The algorithm searches for high curvature. To ensure the points lie on an edge the gradient has to be large, so the points are also thresholded on gradient magnitude. A further restriction that corners should lie on the steepest part of the edge. The edge strength derived is given by

$$C = \nabla^2 I - S\left|\nabla^2 I\right|^2 \quad \textbf{(7.21)}$$

where S is the applied threshold. This method is very good for 90° corners and it performs poorly for 'T' corners.

7.8.3. Moravec

A corner is defined to be a point with low self-similarity. Moravec [80] proposed a feature detector which measures self similarity of an image by taking the sum of square difference (SSD) between a patch centered on the pixel and nearby, largely overlapping patches. The similarity is measured by taking the sum of squared differences (SSD) between the two patches. A lower number indicates more similarity. Pixels in a region of uniform intensity will have similar nearby patches. Pixel on an edge will have patches similar along the edge and different normal to the edge. Pixels on a feature point will have varying nearby patches in all directions.

The corner strength(C) is defined as the minimum SSD between the patch and its neighbors. A local maxima of the corner strength is used to locate the corners. One of the main problems with this operator is that it is not isotropic, if an edge is present that is not in the direction of the neighbors, it is less likely to be detected.

7.8.4. Harris

Harris [81] built upon Moravec's corner detector by computing an approximation to the second derivative of the SSD with respect to the shift. This method is computationally more efficient and can be made isotropic. The approximate second derivative of the SSD with respect to the shift is given by

$$H = \begin{bmatrix} \hat{I}_x^2 & \hat{I}_x\hat{I}_y \\ \hat{I}_x\hat{I}_y & \hat{I}_y^2 \end{bmatrix} \quad \textbf{(7.22)}$$

where the gradients are the average values in a circular patch around the point making it isotropic.

$$C_H = |H| - k\left(trace(H)\right)^2 \quad \textbf{(7.23)}$$

where k is a constant. This is large if both eigenvalues are large, and it avoids explicit computation of the eigenvalues. Thresholding and non-maximal suppression is then used on the corner strength image. Shi and Tomasi [82] suggested using the smallest eigenvalue of H as the corner strength function.

Zheng *et al*. [83] perform an analysis of the computation of H, and found some suitable approximations which allowed them to compute only two smoothed images, instead of the three previously required. They also derive a function k(x, y) to replace constant k in the Harris corner strength in order to improve detection and stability. They

also showed that computation of the local SSD roughly measures the rate of change of edge direction. The resulting response function is

$$C = I_x^2 I_{yy}^2 + I_y^2 I_{xx}^2 - k(x,y)\left(I_x^2 + I_y^2\right) \quad \textbf{(7.24)}$$

7.8.5. Scale Space Based

An alternative approach to find corners is to use the Laplacian of the image, more precisely the Laplacian of Gaussian (LoG). Since the LoG kernel is symmetric, this is effectively performing feature mapping. Gaussian variance determines the size (or scale) of features of interest. These locations of maxima of the LoG over different scales are stable. Lowe [84] obtains scale invariance by convolving the image with a Difference of Gaussians (DoG) kernel at various scales, the maximum of DoG in both space and scale are retained as corner locations. DoG is used since it is a good approximation for LoG and faster to compute. The DoG kernel also responds strongly to edges. To reject edge like features, the eigenvalues of the Hessian of the image are computed at each scale and points are rejected with large ratio of the eigenvalues.

Harris-Laplace [85] features are detected using a similar approach. An image pyramid is built with 1.2 as scale and features are detected by computing C_H at each layer of the pyramid. Features are selected if they are a local maximum of C_H in the image.

7.8.6. SUSAN

Another class of corner detectors is which observe a small patch at a point to look for corner like shape. Smith [86] used this idea to compute the corner strength by looking at the proportion of pixels near to a center, or nucleus, which are very different from the nucleus. They introduced USAN (the univalue segment assimilating nucleus) which

computes a weighted sum of the number of pixels inside a disc whose intensity is within some threshold of the center value. Pixels closer in intensity to the nucleus receive a higher weighting. A low value for the USAN indicates a two dimensional feature, since the center pixel is very different from most of its surroundings. SUSAN (Smallest USAN) is a local minima of the USAN which are the likely corner points.

7.8.7. Trajkovic and Hedley

Trajkovic and Hedley [87] proposed that a patch is not self similar if pixels generally look different from the center of the patch. This is measured by considering a circle. The pixel value at the center of the circle f_c, and the opposite points f_{p1} f_{p2} on the diameter are used to compute the corner response as

$$C = \min\left(\left(f_{p1} - f_c\right)^2 + \left(f_{p2} - f_c\right)^2\right) \quad \textbf{(7.25)}$$

7.9. Curve detection

7.9.1. Active Contour Models

Kass, Witkin and Terzopoulos [42] developed Active Contour Models which provide a solution to the image processing problem of determining the silhouette. Active Contour Models, known colloquially as "snakes," [42]-[45] are energy-minimizing curves that deform to fit image features. A snake is a list of control points that move and reach the feature of interest (for example the edge boundary of the object) in the image. The number of control points can be fixed or varied dynamically. The snake control points move in such a manner that the total energy reaches a minimum. The total energy of the snake is the combined energy of all the control points. The energy function of each point is composed of gradients of the three types of forces:

Internal forces: These forces give the model tension and stiffness which eliminate discontinuities in the contour.

External forces: External constraints come from high-level sources such as human operators or automatic initialization procedures. The aim of these kinds of constraints is to draw the contour towards some desired features in the images (e.g., corner points).

Image forces: Image energy is used to drive the model towards salient features such as light and dark regions, edges and terminations.

The internal energy function is intended to enforce a shape on the deformable contour and to maintain a constant distance between the points in the contour. Additional terms can be added to influence the motion of the contour. The external energy function attracts the deformable contour to interesting features, such as object boundaries, in an image. Any energy expression that accomplishes this attraction can be considered for use. The image gradient energy function attracts the deformable contour to edges in the image. An energy expression proportional to the gradient magnitude will attract the contour to any edge. Active contour models are a simple solution to low level silhouette detection. Figure 7.1

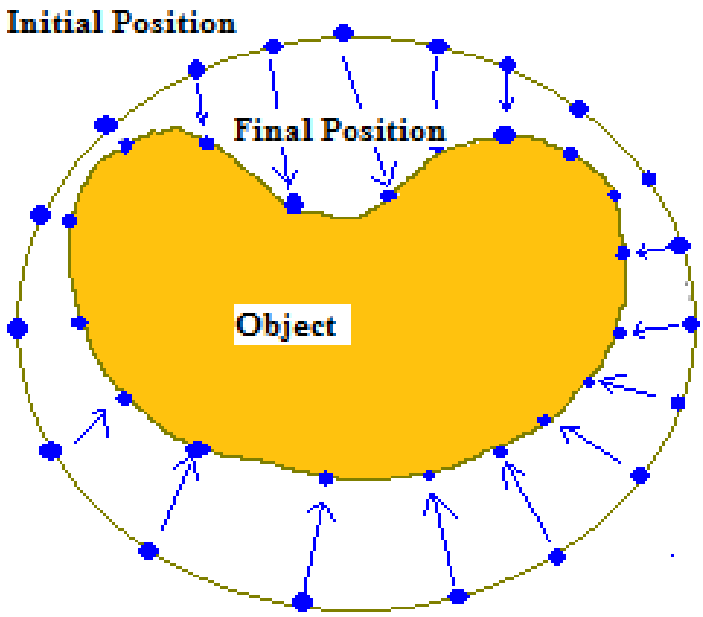

Figure 7.1. Active Contour Models.

shows the active contour model. The snake keeps moving towards the target as the energy function is minimized.

7.10. Indirectly Determined Intersection (IDI) Points

In this section we introduce new image features called IDI points, which is part of the contribution of this dissertation. We develop an error minimization routine to precisely locate line segments in the images. IDI points are the intersections of two or more such lines.

For determining the 3D structure of an object it is necessary to find the actual (or virtual) 3D points or lines fixed to the objects. These 3D points and lines are viewed in the images and we are thus interested in locating these features. Points are 2D features there are 2 DOF (the x and y position) for locating these points in the image. The image points determined by various corner or feature point detection methods are basically image points which have been found by various image processing functions. If the corners are detected by analyzing the local image patches then we also find points which locally look like corners but do not necessarily correspond to the 3D fixed points with respect to the object. We are not really interested in these points. The corner points found may also be affected by the noise in the image. All points found by various corner detection processes do not necessarily correspond to the object fixed 3D points. On the other hand lines have 2DOF (the position and the orientation). It is less error prone to localize these lines to the object edges than the points. This is the reason we have introduced the IDI points.

Definition: Error Minimized and Precisely Located (EMPL) Lines

EMPL lines are lines in the image which have been previously located and then a minimization procedure is applied to reduce an error function and adjust their location precisely.

The EMPL lines are less affected by noise by definition, since there is a minimization procedure to localize the lines based on various edge points on the lines. We develop a routine to precisely locate these individual line segments.

Definition: - Indirectly determined Intersection points

IDI points are 2D image points obtained by intersection of a pair of lines, EMPL lines, or an EMPL line and a detected curve. The lines are previously found by processing the edges in the image and minimizing the errors. The curves are found by active contour models.

Since IDI points are defined as the intersection of a pair EMPL lines, more often than not these IDI points correspond to the 3D object points. Various methods are introduced in following chapter to reject the points which do not correspond to the real object fixed points.

7.11. Locating EMPL Lines

The first step in finding the IDI points is to find EMPL lines. EMPL lines are found by first detecting the edges in the image. We use the Canny edge detection which finds the edges by using the derivative of a Gaussian and the non maximum values in the direction of the gradient within a neighborhood of the edgels are suppressed to thin the edges. Figure 7.2 shows a digital image of a metal object and canny edges are found as shown in the Figure 7.3.

Hough transforms are used to locate the straight lines in the image. The straight lines segments are found and localized using the Hough transform parameter map. Given the edge strength of an image I and the list of line segments produced by Hough transform, each of these lines are localized further by using the raw image edge data (the gradient magnitude and direction). The Derivative of Gaussian of the image is used and it is adaptively thresholded in the local region of the found line to reject all points with values which are lower than the determined threshold. The values of the gradient which is above the threshold are retained along with its gradient direction.

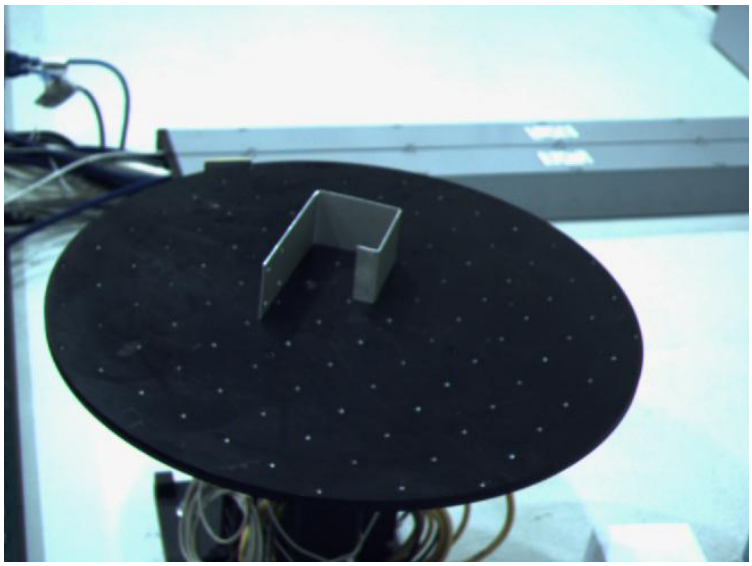

Figure 7.2. A digital image of a metal object.

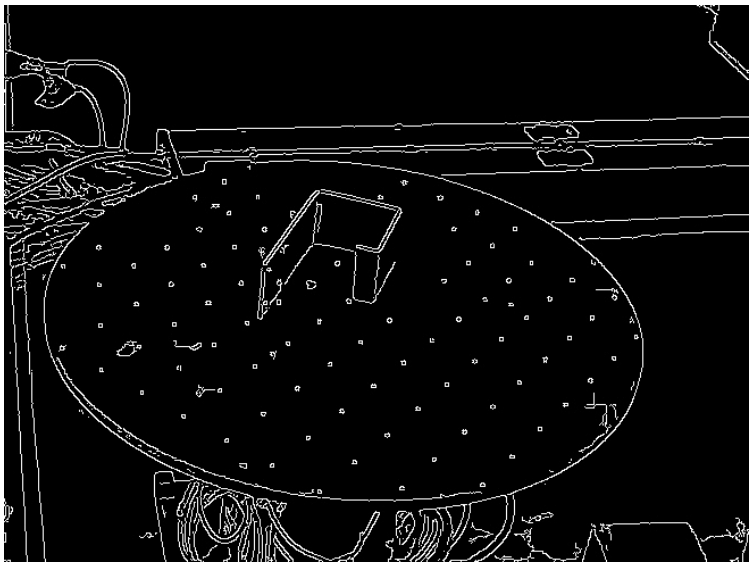

Figure 7.3. Canny Edge Image of a metal object.

A minimization procedure is applied to each line segment in the raw edge data image. The error function is sum of the least squares of the orientational errors and positional errors of the points in the image within a region (bounding box) of the given line as shown in Figure 7.4. If all the points in the bounding box contribute equally to the error function, it will be affected by noise, so only a set of selected points which are above a threshold are used. A restriction on the gradient direction is also applied. Points with large gradient deviations are not used as they correspond to noise.. The error for each pixel is also weighted according to their gradient magnitude to allow the line to correctly adjust to the desired position. The error function is represented as

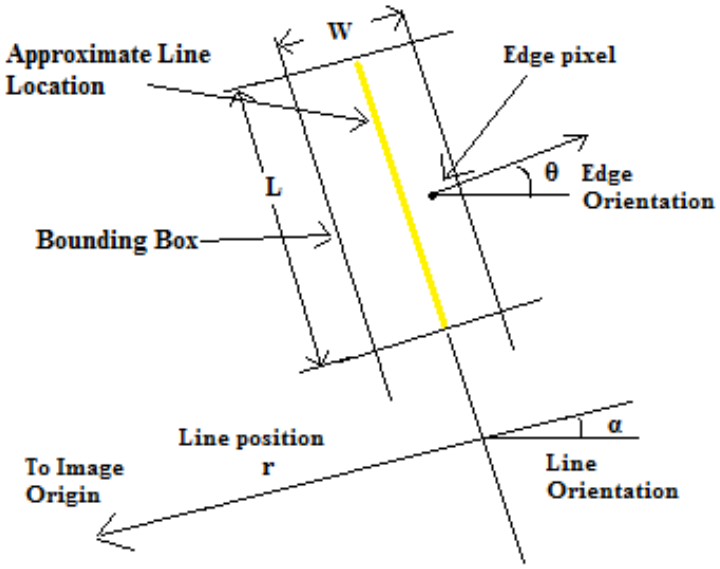

Figure 7.4. Bounding box for a line in consideration.

$$E_{x,y} = \begin{cases} \left|\bar{I}_e\right|\left(\left(X\cos(\theta)+Y\sin(\theta)-r\right)^2+\left(\theta-\alpha\right)^2\right) \forall \left|\bar{I}_e\right| > T, \left|\theta-\alpha\right| < T_\theta \\ 0 \forall \left|\bar{I}_e\right| \le T, \left|\theta-\alpha\right| > T_\theta \end{cases} \quad \textbf{(7.26)}$$

where (x, y) is the image point and $\bar{I}_e$ is its gradient magnitude and θ is its direction. A local coordinate system (i, j) is chosen x-axis is along the line and y-axis is in the normal direction. T_θ is a constant threshold applied to the angle error. T is the threshold which is computed locally in the region around the whole line using the mean as

$$T = \frac{\alpha}{N} \sum_{i=0}^{L} \sum_{j=-\delta}^{\delta} \left| \overline{I}(i,j)_e \right| \quad \textbf{(7.27)}$$

where L is the length of the line and δ is a constant width in consideration on both sides of the line, N is the total number of such image points which belong to the region which is approximately equal to $N = 2(\delta + 1)L$ and α is a constant. The threshold could also be computed by using the peak of the gradient as

$$T = \alpha \underset{i=[0,L], j=[-\delta,\delta]}{Max} \left(\left| \overline{I}(i,j)_e \right| \right) \quad \textbf{(7.28)}$$

The threshold used in (7.26) can also be a variable along the line segment computed locally in a small region dividing the line into small segment locally given by any of the following equations

$$T(p) = \frac{\alpha}{N} \sum_{i=-\varepsilon}^{\varepsilon} \sum_{j=-\delta}^{\delta} \left| \overline{I}(p+i,j)_e \right| \quad \textbf{(7.29)}$$

$$T(p) = \alpha \underset{i=[-\varepsilon,\varepsilon], j=[-\delta,\delta]}{Max} \left(\left| \overline{I}(p+i,j)_e \right| \right) \quad \textbf{(7.30)}$$

where ε is a constant length of the segment on two sides of the point along the line in consideration and p is a point along the line.

The thresholds computed by these two methods did not have much difference though one is a function along the length of the line the other is a constant for the whole line segment for smaller line segments. For longer line segments it is better to choose the variable threshold.

The total error computed by (7.26) for each point within the bounding rectangle around the line is used. This error function is minimized with respect to the line parameters. The Figure 7.5 shows the located edge segments in an image. The IDI points are found by finding the intersection points of all possible pair of EMPL lines computed.

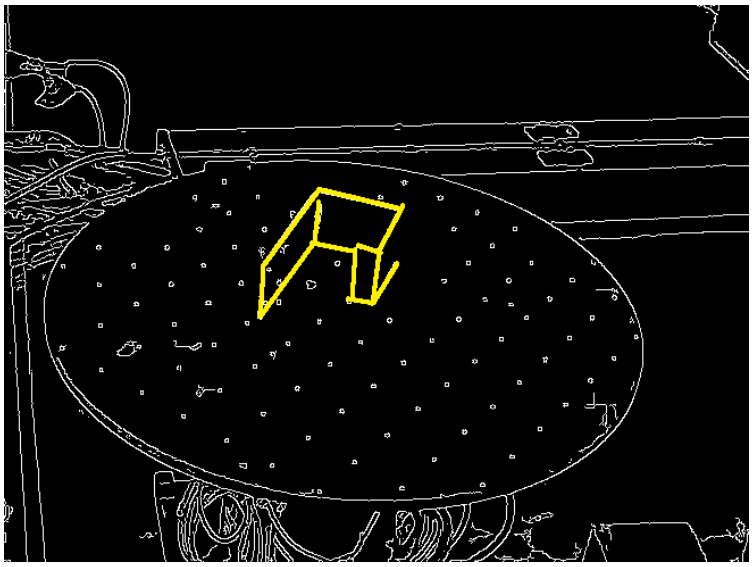

Figure 7.5. EMPL lines corresponding to edges of an object.

7.12. Comparison of IDI points and Corners

7.12.1.Location Accuracy

The accuracy of the IDI points depend on the accuracy of the EMPL lines since the EMPL lines are localized in the image using the image gradients. The accuracy of the lines are dependent on the length of the segments used since using longer line segments gives us more accurate localization. Thus the IDI points are more accurate as compared to the corners which only use very small image patch around to localize.

7.12.2.Detection Volume

Corner detection methods usually depend on the very local image gradients. This dependency often affects the volume of feature points found. The corner detection

methods also detect points where there might be lighting variance. These points detected are often difficult to differentiate from the actual ones of interest.

7.13. Experiments

7.13.1.Single Line Projection

An object with straight edges is placed on a rotary table at various locations. Biclops is used to take images of this object from various locations. We only concentrate on a single visible edge for this experiment. The images are processed to find the edges in

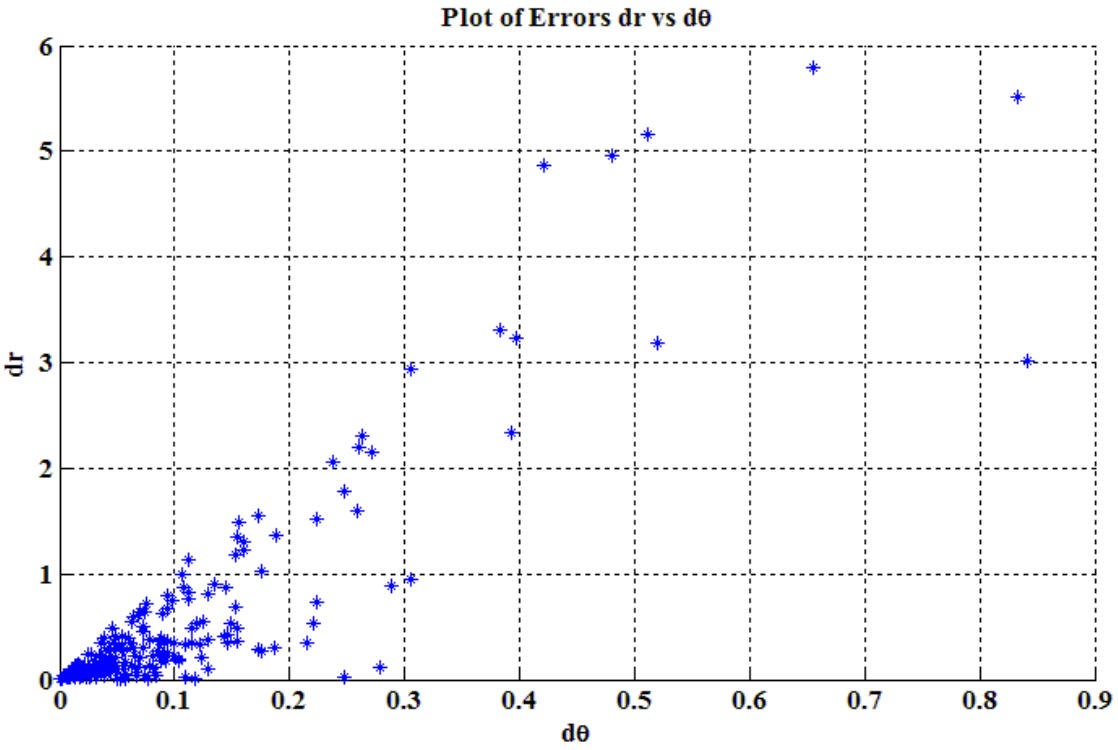

Figure 7.6. Parameter errors of EMPL lines of various lengths.

the image and the edge images are used to locate various line segments in the image.

For this experiment the corresponding line segment to the edge in consideration is picked manually. The parameters of the line segment extracted are compared to the actual parameters obtained by the projection of the 3D object edge in to the image and

correcting it manually. By correcting the edge in the image manually we are avoiding the errors due to camera locations and object locations. Thus the only errors are due to the noise in images and their acquisition. The errors in the parameters are noted down. The two parameters for each line used are the (r, θ). Apart from this the length of the line segment is also known.

The experiment is repeated with various lengths of edges. The errors in the parameters are plotted as shown in Figure 7.6.

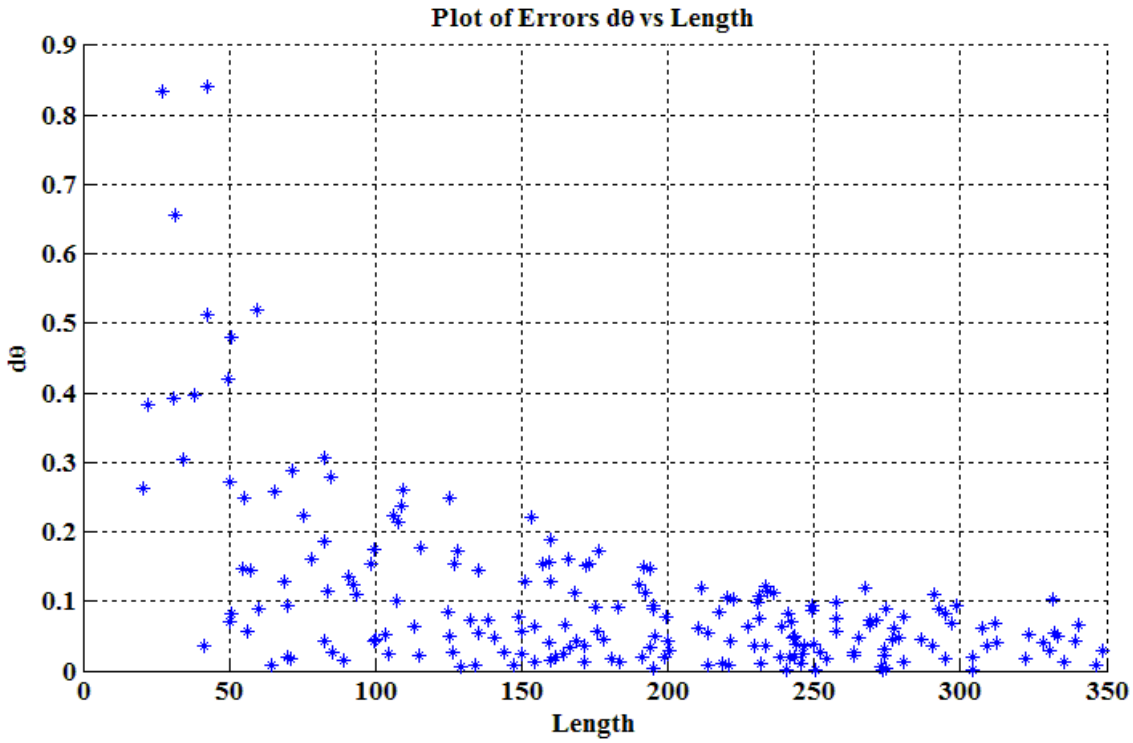

Figure 7.7. Orientation errors (deg) of EMPL lines plotted against length (pix) of the segments.

The positional errors of the lines segments are plotted against the lengths of the line segments as shown in Figure 7.7. The orientational errors of the line segments are plotted against the length of the line segments as shown in Figure 7.8.

It can be seen clearly that the errors depend on the length of the line segments. The error decreases as the segments get larger up to a certain order. The errors are not affected too much for larger length segments. This experiment proves that by utilizing more image information, *i.e.*, longer length segments, the errors in line extraction are

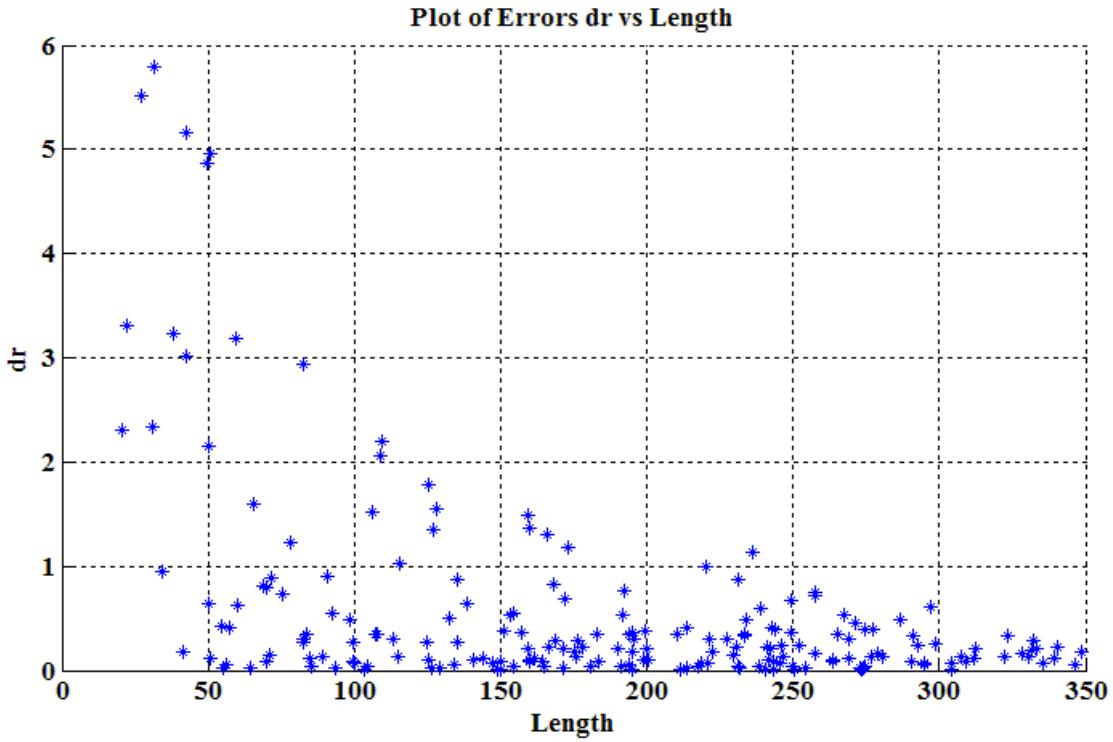

Figure 7.8. Positional error (pix) of EMPL lines plotted against length (pix) of the segments.

reduced.

7.13.2. Comparison: Corners vs. IDI Points

Corner detectors are not usually very robust and often require expert supervision or large redundancies have to be introduced to prevent the effect of individual errors from dominating the recognition task. The underlined hypothesis here is that the image noise has less effect in finding the 3D object points using the IDI points than using the regular corner points.

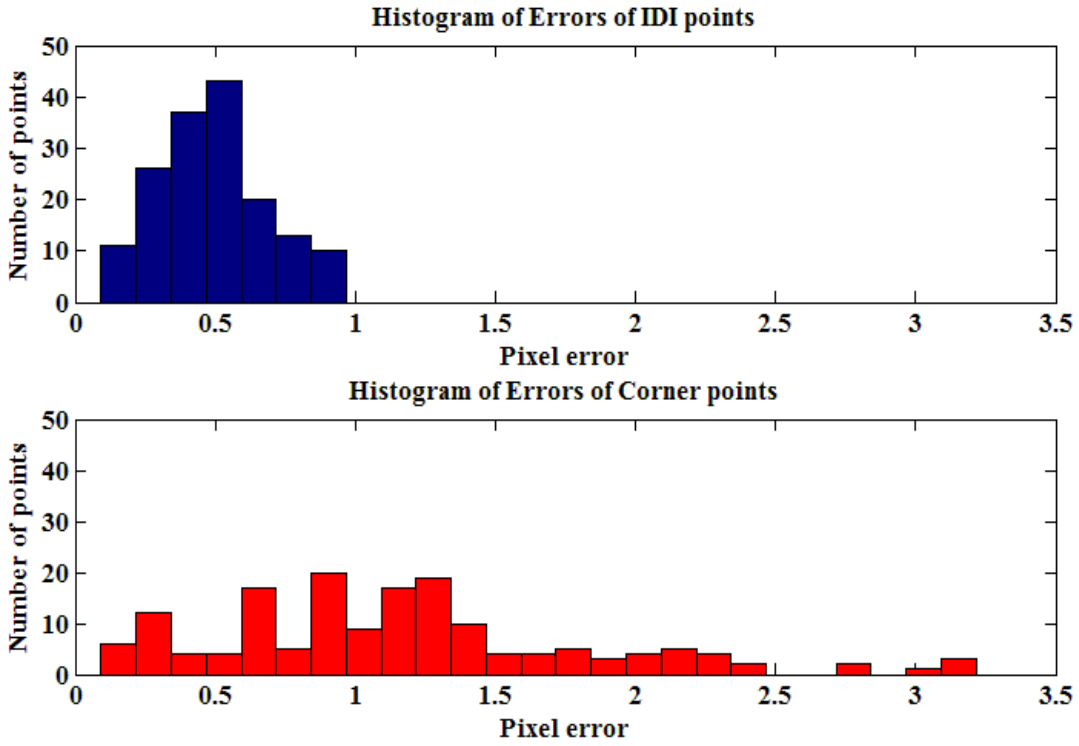

Figure 7.9. Histograms of errors (Pixels) of IDI Points and Corners.

To prove this concept an object with straight edges is placed on a rotary table at a known location. Biclops is used to take images of this object from various known locations. The object is placed such that two prominent edges are visible all the time. Images of this pair of intersecting lines at a known location are captured from various viewpoints. These lines are projected on to the images and their intersecting points computed.

The images are processed to find the edges in the image and the edge images are used to locate various line segments in the image. The two line segments corresponding to the two edges in consideration are picked manually. The two EMPL lines picked are used to find their IDI point.

The images are also processed by the Harris corner detection method to find the corners in the image. The thresholds are selected so as to find the required corners. Sub

pixel localization is carried out to find the precise locations of these corners. The corners picked by the algorithm are compared to the desired location and a best match is chosen. Sometimes the corner in consideration is not detected. In this case the rest of the process

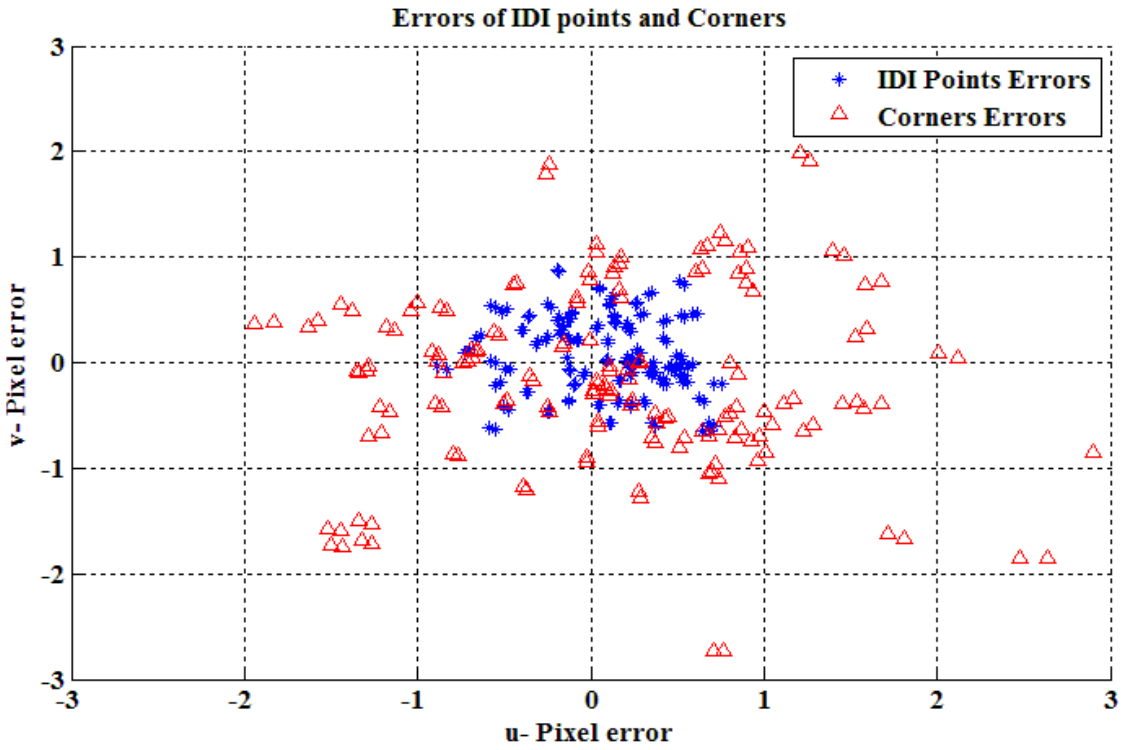

Figure 7.10. IDI points and Corner Points errors (Pixels).

is ignored.

The Object 3D points and lines are projected into the image by using their known locations these are corrected manually and used as ground truth corners. The IDI point and the detected corners are compared to the ground truth corner locations. The corners' locations in the image usually did not correspond to the IDI points. The experiment is repeated with various locations of the object.

The histograms of norms of the errors are plotted in Figure 7.9. The image point errors for both IDI points and corners are plotted in Figure 7.10. It can be clearly seen

that the IDI points have less errors compared to the localized corners. IDI points gave a better localization of the points (though these may not necessarily be the image corners).

This experiment proves that points located by intersection of the EMPL lines, *i.e.,* the IDI points are less error sensitive as compared to the corners detected by usual localized image processing methods.

The comparison is done in the number of points found correctly and the number of the real points of interest missed and the number of points found which do not correspond to any fixed points on the object. We found that the number of corners found was large as compared to the IDI points general.

7.14. Conclusion

It is necessary to find image points which more often correspond to object fixed points than not. So we have introduced the concept of IDI points. This chapter introduced EMPL lines which are obtained by Hough transforms and then further localized by error minimization procedure. EMPL lines are used to locate the IDI points in the images. Experiments prove the concept that IDI points more often correspond to the object fixed points. Also it is shown empirically that the IDI points are less affected by noise.

Chapter 8. 3D-Structure Using IDI Points

8.1. Introduction

Many applications, e.g., motion planning, virtual reality, CAD, vehicle navigation, object recognition, photogrammetry, remote sensing, etc., all require a geometrical representation of the three dimensional structure of a scene. Inference of 3D structure of objects in a scene from its 2D projections is a long studied problem. One of the important methods is to determine the 3D shape of visible objects in a static scene from images acquired by two or more cameras or a single camera at multiple view points. The images obtained from numerous viewpoints are processed for various primitives such as points, lines, curves, planar entities, etc., which are the input to the system.

8.2. 3D Structure

The 3D structure of a scene is a geometrical representation of all the objects in the scene. The geometrical representation of the scene consists of a set of 3D points and lines that make up the objects. For the sake of convenience a coordinate system may be defined with respect to the objects, called object coordinate system. All the object points and edges or curves in consideration are expressed in this object coordinate system or the global coordinate system as desired.

8.3. Perspective Projection Modelling

A pinhole model for the cameras is assumed. The pinhole model is described in detail in Section 1.4. The perspective projection is presented in section 1.5 and the coordinate systems are shown in Figure 1.2. Considering the projection of a 3D point and its image projection and using similar triangles we get two following equations

$$\frac{(u-u_0)}{\left(f/s_x\right)}=\frac{x_c}{z_c}\Leftrightarrow\frac{(u-u_0)}{f_x}=\frac{x_c}{z_c} \quad \textbf{(8.1)}$$

$$\frac{(v-v_0)}{\left(f/s_y\right)}=\frac{y_c}{z_c}\Leftrightarrow\frac{(v-v_0)}{f_y}=\frac{y_c}{z_c} \quad \textbf{(8.2)}$$

where focal length of the camera is expressed in pixels along u and v axis as f_x and f_y respectively and given by f/s_x and f/s_y respectively. Putting the above two equations in a matrix form we get

$$\begin{Bmatrix}\kappa u\\ \kappa v\\ \kappa\end{Bmatrix}=\begin{bmatrix}f_x & 0 & u_0 & 0\\ 0 & f_y & v_0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{Bmatrix}x\\ y\\ z\\ 1\end{Bmatrix}_c\Leftrightarrow\kappa\{U\}=\begin{bmatrix}K & \bar{0}_3\end{bmatrix}\{X\}_c \quad \textbf{(8.3)}$$

where 'κ' is the homogenous coordinates scale factor. The above equation transforms a point in the cameras 3D coordinate system to the image coordinates, *i.e.*, it projects the point into the image. The transformation matrix used for this purpose is the internal parameter matrix [K]. The inverse transformation can also be represented as

$$\begin{Bmatrix}x\\ y\\ z\\ 1\end{Bmatrix}_c=\begin{bmatrix}1/f_x & 0 & -u_0/f_x & 0\\ 0 & 1/f_y & -v_0/f_y & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{Bmatrix}\kappa u\\ \kappa v\\ \kappa\\ 1\end{Bmatrix}\Leftrightarrow\{X\}_c=\begin{bmatrix}K^{-1} & \bar{0}_3\\ \bar{0}_3^T & 1\end{bmatrix}\begin{Bmatrix}\kappa U\\ 1\end{Bmatrix} \quad \textbf{(8.4)}$$

Since 'κ' is an arbitrary constant the above solution is not a unique 3D point, in fact it is a ray that passes through the camera optical center $\{0\}_c$.

In general we will not be working in the camera 3D coordinate system, there is always another coordinate system with respect to which we have all the 3D point locations. So to project the points from that coordinate system to the images we will need

the transformation matrix between the two coordinate systems, *i.e.*, we will need the position (translation vector T) and orientation (rotation matrix R) of the camera with respect to that coordinate system. This transformation is called the external parameter matrix. The transformation is given by

$$\begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}_c = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}_o \Leftrightarrow \{X\}_c = \begin{bmatrix} R & T \\ \overline{0}_3^T & 1 \end{bmatrix} \{X\}_o \Leftrightarrow \{X\}_c = {}^c[P]_o \{X\}_o \qquad \textbf{(8.5)}$$

The inverse transformation is given by

$$\{X\}_o = \begin{bmatrix} R & T \\ \overline{0}_3^T & 1 \end{bmatrix}^{-1} \{X\}_c = \begin{bmatrix} R^T & -R^T T \\ \overline{0}_3^T & 1 \end{bmatrix} \{X\}_c \qquad \textbf{(8.6)}$$

The rotation matrix R is an orthonormal matrix hence its inverse is its transpose. Since the origin of the camera coordinate system is the camera center we get the camera center as $C = -R^T T$. Using the camera external parameter matrix the point in the given coordinate system can be projected in to the image as

$$\kappa\{U\} = \begin{bmatrix} K & \overline{0}_3 \end{bmatrix} \begin{bmatrix} R & T \\ \overline{0}_3^T & 1 \end{bmatrix} \{X\}_o \Leftrightarrow \kappa\{U\} = [P]_n \{X\}_o \qquad \textbf{(8.7)}$$

Using the inverse camera transformation matrix the camera ray can be expressed in the given coordinate system as

$$\{X\}_o = \begin{bmatrix} R^T & -R^T T \\ \overline{0}_3^T & 1 \end{bmatrix} \begin{bmatrix} K^{-1} & \overline{0}_3 \\ \overline{0}_3^T & 1 \end{bmatrix} \begin{Bmatrix} \kappa U \\ 1 \end{Bmatrix} \qquad \textbf{(8.8)}$$

Breaking down the above equation in to its components clearly shows this is a ray passing through the camera center

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_o = -R^T T + \kappa R^T K^{-1} U \Leftrightarrow \begin{Bmatrix} c_x \\ c_y \\ c_z \end{Bmatrix} + \kappa \begin{Bmatrix} d_x \\ d_y \\ d_z \end{Bmatrix} \quad \textbf{(8.9)}$$

Thus the cameras project the 3D space onto a 2D image plane as given by (8.7). So for every 3D point that is in the view there is a 2D image point. Considering the inverse problem for every image point there exist a ray that passes through the point and the camera optical axis given by (8.8).

8.4. Triangulation

Images from a particular camera position and orientation will only give us some clue about where the 3D points are located. From a single view we cannot judge the point's depth as measured from the camera optical axis. We will need more that a single view for each point that needs to be located in 3D. Two camera views are necessary to locate a 3D point in a scene.

The process of locating the 3D points is called triangulation [88]. The triangulation process involves finding the directional vectors of the 2D projection in both cameras and finding their intersection as the required 3D point. In reality the two rays may not intersect at all due to errors from various sources hence a point closest to both lines is

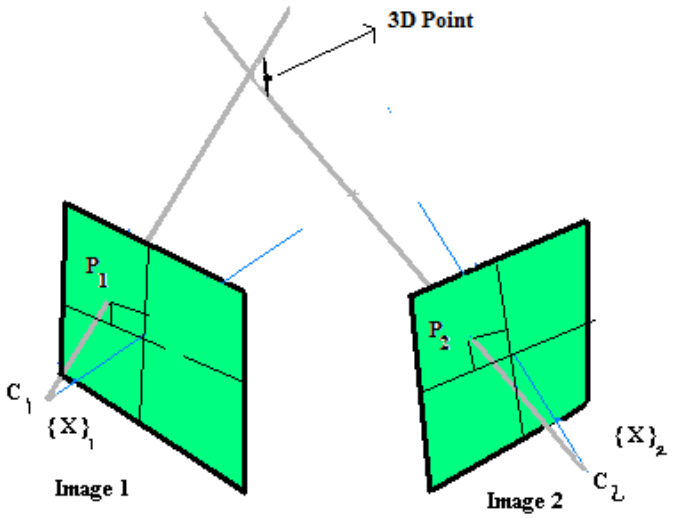

Figure 8.1. Triangulation of a 3D point.

used. The closest point to both the lines is the midpoint of the common normal. For more than two rays the goal of triangulation is to find a point lying on all these lines. Since most of the time the lines accompany errors they do not intersect. So we are looking for a point which is closest to all the given rays. The closest distance of a point from a line is its perpendicular distance from the line. A point location is found by minimizing the perpendicular distance of the point to the given rays.

8.5. Triangulating edges

Consider the two images of a straight line (L) from two different views as shown in Figure 8.2. Let the equations of the line in first and second image be

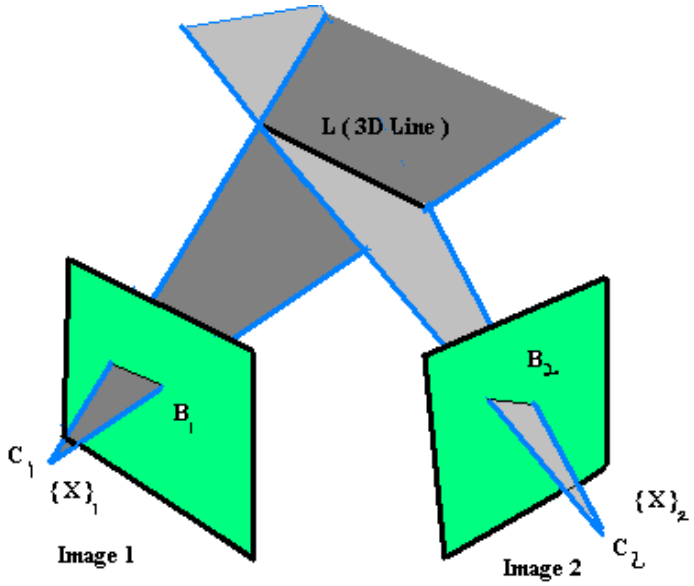

Figure 8.2. Projection of a 3D line in two images.

$$[B]_1 \{U\}_1 = 0 \qquad \textbf{(8.10)}$$

$$[B]_2 \{U\}_2 = 0 \qquad \textbf{(8.11)}$$

where $[B] = [b_1 \quad b_2 \quad b_3]$ is the row vector of the coefficients of the equation of a 2D line in an image.

Using equation (8.7) in (8.10) and (8.11) we get

$$[B]_1 \{U\}_1 = 0 \Leftrightarrow [B]_1 [P]_1 \{X\}_o = 0 \quad \textbf{(8.12)}$$

$$[B]_2 \{U\}_2 = 0 \Leftrightarrow [B]_2 [P]_2 \{X\}_o = 0 \quad \textbf{(8.13)}$$

Thus for two images from two different views (where the views are not degenerate giving parallel planes) we get two first order equations which in general represent the two planes in 3D as shown in Figure 8.2 whose intersection is the required line. Thus by knowing the 2D equations of the projected lines in two images, the 3D equation of the line can be computed.

8.6. Epipolar Geometry-Fundamental Matrix

Consider the two images of a point 'X' in 3D from two different views, as shown in the Figure 8.3. Let the corresponding points in the two images be $\{x\}_1$ and $\{x\}_2$.

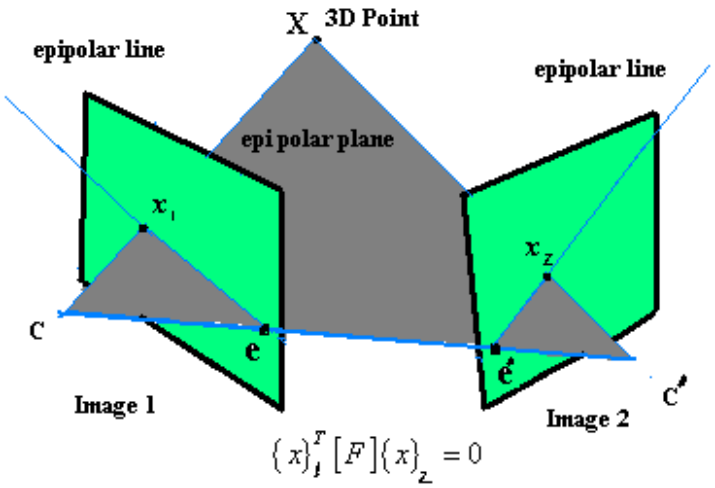

Figure 8.3. Projection of a 3D point in two images.

$$\{x\}_i = [P]_i \{X\} \quad \textbf{(8.14)}$$

For every point projected in one image, there is an epipolar constraint which restricts the corresponding point in the other image to lie on a line called epipolar line. Geometrically the epipolar constraint implies that the 3D point, the projected points in the images and the centers of the cameras all lie in a plane, as shown in Figure 8.3. Using the epipolar constraint for corresponding points in different images leads to the relation

$$\{x\}_i^T [F] \{x\}_j = 0 \quad \textbf{(8.15)}$$

where $\{x\}_i$ and $\{x\}_j$ are the corresponding points in two different images and [F] is the fundamental matrix [14] which is unique for a particular set of images. The fundamental matrix [14] can be computed from the camera matrices P, P'

$$F = [e']_x P' P^+ \quad \textbf{(8.16)}$$

where P^+ is the pseudo-inverse of P, $[e']_\times$ is the cross product matrix of the epipole in the second image e' as shown in the Figure 8.3, the epipole is given projecting the camera center C of the first camera into the second image as

$$e' = P'C \quad \textbf{(8.17)}$$

The camera center 'C' is the null vector of the Projection matrix P, given by

$$PC = 0 \quad \textbf{(8.18)}$$

Thus for every pair of images, there exists a constant fundamental matrix that can be computed using various linear and nonlinear methods [7], [8], [10], [14], [47].

8.7. Correspondence

As we have seen finding the 3D structure requires more than one view of an object. Using two views of a point we could compute the 3D location of a point and using two views of a line we could find the 3D line. The problem seems to be solved but if only we know which points are projected where in the two images, *i.e.*, we need to know the corresponding points in the images. For humans it is often very easy to locate corresponding points in various images though the views are radically different. For most of the applications like photogrammetry, image registration etc, it is usually fine to allow the humans to intervene and provide this necessary information. In this work we investigate automatic methods to find the corresponding points. Some applications

usually use views of objects that are close to each other so that the images look similar. They use the sum of square differences (SSD) to match the image patches around the points.

8.8. Automatic Bundle Correspondence

One of the contributions of this dissertation is to develop an algorithm for automatic correspondence of the points and lines in two images. We developed a bundle correspondence algorithm which considers all possible matches and uses the topology and intersection constraints to reject or confirm the potential matches. All the points and lines are considered in a bundle to find the various corresponding entities.

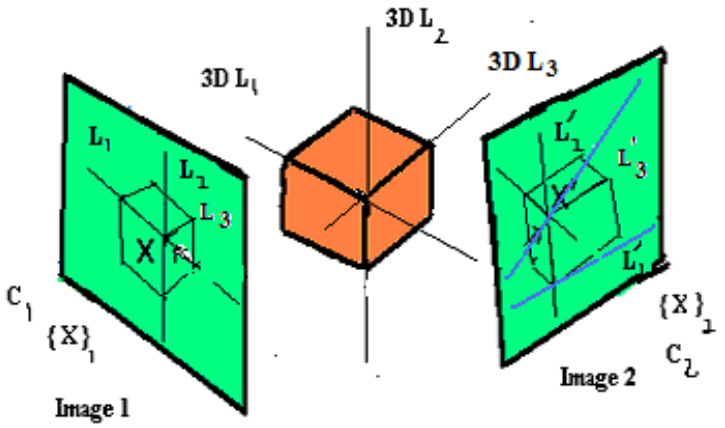

Figure 8.4. Correspondence of Lines and Intersection points.

The problem is put together as follows, given two images and various EMPL lines and their IDI points, the points and lines in each image need to be paired with the points and lines in the other image. In each image we have a list of IDI points and a list of EMPL lines and the intersection information is stored, *i.e.,* which EMPL lines intersect at which IDI points. We use the fundamental matrix for the correspondence. The epipolar lines for the IDI points in one image are computed in the second image and vice versa. If two IDI points correspond then epipolar lines of those points should pass through the

corresponding points. We use this information to match up the IDI points and the EMPL lines.

The points in one image are used to find the epipolar lines in the other image. Now for each point we also have the list of lines passing through it (intersecting at this point). The other image is investigated for various lines that intersect the epipolar line. These lines are potential matches to the lines that pass through the point. Also the intersecting points of the epipolar line with all the lines in the other image are potential matches to the point. So for one point and the lines that intersect at this point we have the list of matches.

Consider a cube projected into two images as shown in Figure 8.5. Consider the point X projected into first image as x_1. Now lines L_1, L_2 and L_3 intersect at this point. Considering the two IDI points on the line L_2, the corresponding epipolar lines for two points are shown. Now the various points where these corresponding epipolar lines intersect the lines in the other image are potential matches to the points. Considering the topology there is a line connecting the two points in image. The epipolar line of bottom point intersects with three lines in the other image. The epipolar line of top point intersects with 5 lines (of which 3 are same). Considering

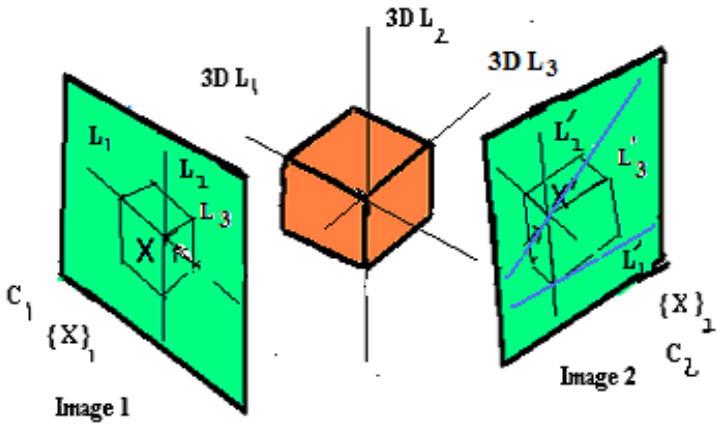

Figure 8.5. Using epipolar constraint for correspondence.

that the three lines L_1, L_2 and L_3 intersect at a single point the corresponding point to X is immediately known. Now only does one line exists that intersects the two epipolar lines. This line is the line corresponding to L_2. And the two intersecting points of the corresponding lines with the epipolar line are the corresponding points to the top and bottom points. The process is also repeated by considering the points and lines in second image and the epipolar lines in the first image. Ambiguities are resolved by considering all the points and lines from both images. Thus using various topological and intersecting constrains the correspondence is established. There might be some points and lines which do not have corresponding lines and points in the other image. Sometimes we introduce more IDI points in an image when breaking a line into two.

8.8.1. Summary of Bundle Correspondence

- Make a list of EMPL lines and IDI points
- Save list of EMPL lines passing through an IDI point
- Save a list of IDI points on an EMPL line.
- Use calibration to find the Fundamental Matrix
- Find epipolar lines in the one image of all the IDI points in another image and vice versa
- Find the intersections of the extended EMPL lines and the epipolar lines. This also gives us the information where this intersection is with respect the EMPL line segment whether inside the end points or whether to outer side of one end point. Store this information from both images.
- Start applying the constraints available with some tolerance to image noise.
- Go through each IDI point in one image

- The intersection of extended EMPL lines and the epipolar lines of the IDI point are stored as possible matches to the IDI point. And store these EMPL lines as possible matches to the EMPL lines passing through the IDI point in first image.
- Make the possible matches list of all the IDI points and all the EMPL lines in one image to the other and vice versa.
- Go through the list of EMPL lines in one image and the list of IDI points on it. To find more information about its corresponding EMPL line, the list of IDI points on this EMPL line is used. Since this line is passing through these IDI points it should intersect the epipolar lines of those points in the other image. Using this information the possible matches are reduced.
- Same step as above but it is done from image two to image one.
- Go through the list of EMPL lines in image 1. We know the list of IDI points on this line. Use these points to locate their corresponding epipolar lines in the other image.
- Thus considering the chains of IDI points and the EMPL lines the topology is utilized to come up with the final matches of corresponding entities.

8.8.2. Practical Issues

Non Unique Point

Sometimes in a real scene, more than two edges (say three) intersect at a point. Due to image noise when considering a pair of EMPL out of these (three edges) they intersect at some location, which is not often the same as the other pair. In other words all the EMPL lines do not intersect at a unique point. We allow a tolerance to the IDI points to declare them as unique. This tolerance in distance is in fact the part of the topology,

i.e., even if we make two points instead of one unique point the distance between the two points in the image is very small thus the topology information will automatically solve this issue. They could in reality be two different points but seen as one or are very close to each other. Again this is taken care by the topology.

EMPL Line segment length

In practice when finding EMPL lines we may have shorter segments (or two broken lines). To solve this problem the EMPL lines found are kept with the confirmed lengths and the unconfirmed extensions in length. These extended line segments are used to intersect the epipolar lines instead of the confirmed lengths. The intersection points (with the epipolar lines) too far on the extension are trusted less.

The extension tolerance in the segments is also used to join segments too close and confirm to the topology.

False Augmented EMPL segment

Sometimes due to the image view, two unique edges of an object so happens that they line up to show as one single line. The topology of the intersection with other lines shows that the line should be broken into two pieces to give a better match to their counterparts in the other image.

False Intersection

Sometimes it seems that two lines are intersecting in an image but truly they might not. These false matches if made initially will be corrected by using the constraints from adjacent points and lines. Consider two lines which do not intersect in 3D space. These lines may be projected as intersected lines in a particular image. When the epipolar

line to the intersecting point is investigated in the other image, the lines do not intersect on this epipolar line at the same location.

8.9. Bundle Adjustment

Once the correspondence is established for a point or a line in two or more images the triangulation methods for points and lines gives us the 3D points and lines. In other words the models of the objects are found.

The errors in the internal and external camera parameters of various views contribute to the errors in the computed 3D objects' structure.

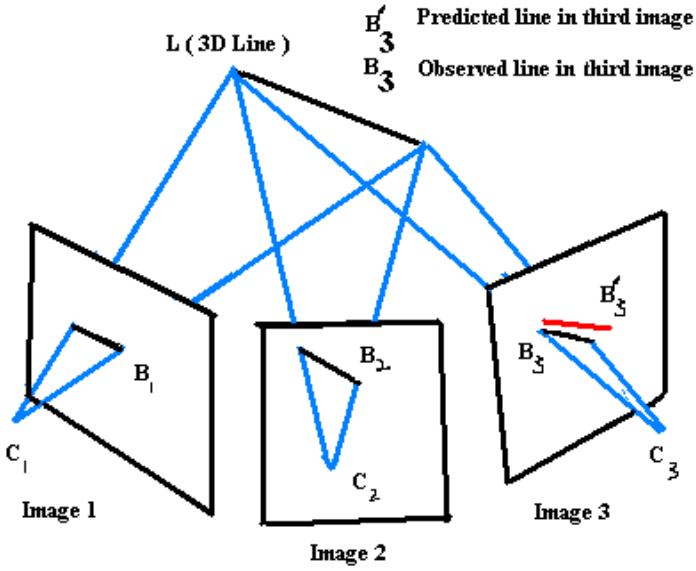

Figure 8.6. Projection of a 3D line in three images. Also shown is the predicted line.

The errors in the models found can be corrected by a global optimization procedure called bundle adjustment.

The reasonably accurate models found are projected into all the image views. If the computed positions are correct then these projected points and lines in the other image should have less error when compared with the actual IDI points and the EMPL lines found. But there are errors in the projections, often called reprojection errors.

Considering that the parameters to the optimization problem are the 3D structure points and the camera parameters the reprojection errors are minimized to obtain an optimum solution. Figure 8.6 shows the two end points of a line segment projected into 3

images. From first two images once correspondence is established the 3D points can be computed. Using this computed 3D points it can be projected into the third image. Now we already have the points in the image found by image process. The projected points and the computed points should be the same but usually not, so the errors between these points are found. Now if the 3D line is modified to correct error we will introduce the error in the other two images. To avoid the problem the 3D points are varied and all the three projections of the line in the three images are minimized at a time getting an optimum solution. This is called the bundle adjustment [89].

Usually it involves n points viewed from m camera poses. And the parameters of the different camera views are also included in the optimization. Let X_{ij} be the projection of the i^{th} point on j^{th} image. Let '$\mathbf{a}_j$' be a vector of all the parameters of the camera pose j. Let '$\mathbf{b}_i$' be the vector of all the 3D points. Let the projection of all the '$\mathbf{b}_i$' in image j be given by $x'_{ij} = R\left(a_j, b_i\right)$. Now we can measure the point's error in the images. The error is not valid for points which are not visible which should not be summed. So the total error is given by

$$E = \sum_{i=1}^{n}\sum_{j=1}^{m}\delta_{ij}\left|R\left(a_j, b_i\right) - x_{ij}\right| \qquad \textbf{(8.19)}$$

where δ_{ij} is the visibility parameter. It is zero if the point is not found in the image and one if the point is found in the image. By minimizing this total error with respect to all the parameters, *i.e*., the camera external parameters and the 3D points we get the desired solution as

$$\left\langle a_{j=1..m}, b_{i=1..n} \right\rangle_{\min} = \min_{\left\langle a_{j=1..m}, b_{i=1..n} \right\rangle} \sum_{i=1}^{n} \sum_{j=1}^{m} \delta_{ij} \left| R\left(a_j, b_i\right) - x_{ij} \right| \quad \textbf{(8.20)}$$

Note that each camera view adds 6 parameters to the bundle adjustment. So it is really good to include the camera parameters in the bundle adjustment only if we are looking at a large number of points in different views. Since we are dealing with only small set of points in our experiment we have not included the camera parameters in the adjustment. Our cameras are accurately calibrated using various methods discussed previously. We use the bundle adjustment idea only to improve the accuracy of the 3D points using

$$\left\langle b_{i=1..n} \right\rangle_{\min} = \min_{\left\langle b_{i=1..n} \right\rangle} \sum_{i=1}^{n} \sum_{j=1}^{m} \delta_{ij} \left| R\left(a_j, b_i\right) - x_{ij} \right| \quad \textbf{(8.21)}$$

8.9.1. Camera Calibration

The bundle adjustment can also be used to find the camera internal parameters. Since we used Biclops we have two cameras so there are two sets of internal camera parameters, one for each camera. These internal camera parameters can also be included in the bundle adjustment. But it is usually a good idea to find the internal camera parameters by other means so that the minimization procedure of the bundle adjustment becomes less complex and faster to solve. Thus a large number of views of a large number of points give a better approximation of the parameters of the camera model. Since we have already calibrated the cameras (internal parameters) and the underlined equipment (external parameters) we did not needed to include these parameters into the bundle adjustment. Figure 8.7 shows the flow chart of the complete procedure to obtain

3D structure from images. The dotted lines show how camera calibration parameters can be included in the minimization procedure to find the final values.

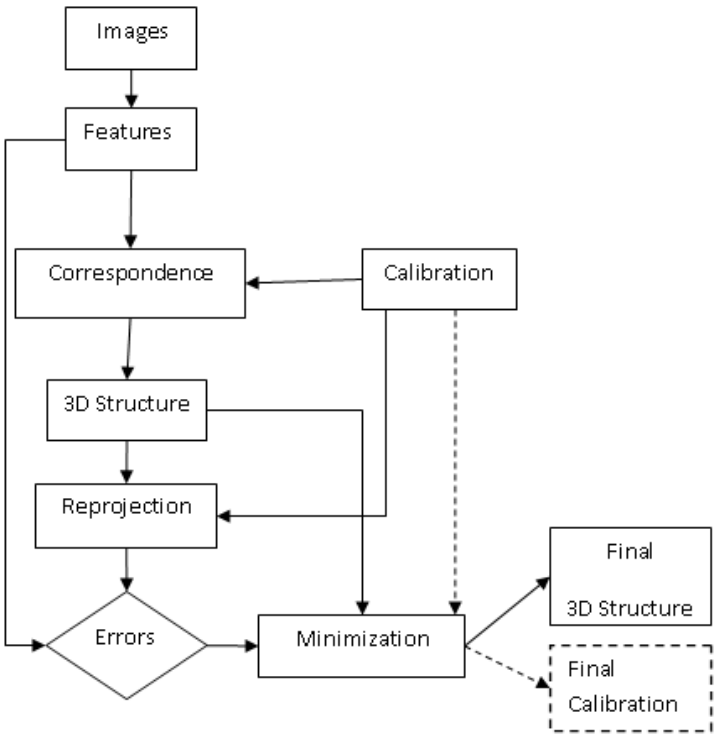

Figure 8.7. Flow chart.

8.10. 3D Modelling

For the sake of convenience in this dissertation the models of various objects are constructed with points and lines alone. The model of an object consists of the number of points and the number of lines connecting two or more of these points. The curves are only used to approximate with lines. The total set of points and lines together is considered as the 3D model of the object.

8.11. Grasping

Once the model of an object is computed it is used to plan how to pick up the object. For simple planar objects we use a curvature aided grasping algorithm developed by Gatla *et al*. [90]. The model is used to make various points on the silhouette of the

object similar to an active contour. The algorithm traverses the list of points and finds the best locations to place the fingers of the Barrett hand to grasp the object. For objects that are not planar, we use the topmost layer points to make a decision as to where to place the fingers.

8.12. Experiments

8.12.1. Correspondence Test

A cube is placed on the rotator table and the robot picks up the Biclops and takes pictures of the cube from various locations. A pair of intersecting lines (of the cube) is used. The EMPL lines and their IDI points in the images are determined. The epipolar constraint is used to find the epipolar lines of the IDI points in the second image. The corresponding lines and points are picked manually. The corresponding IDI points in the second image should lie on the epipolar lines. This hypothesis is tested in the other image by measuring the shortest distance of the IDI point from the epipolar line. Figure 8.8 shows the shortest distances of the IDI points from the epipolar lines. It can be seen that the errors are very small, *i.e.*, within a pixel. These epipolar errors are dependent on various factors which include the errors in the external parameters of the cameras and the noise in the images.

8.12.2. Triangulation Test (Projection of a pair of intersecting lines)

The corresponding IDI points are also used to triangulate and find the 3D points. The 3D points are compared with the known 3D locations. Figure 8.9 shows the triangulation error magnitudes. It can be seen from the histogram that the errors are very small, indicating a very good calibration of the equipment.

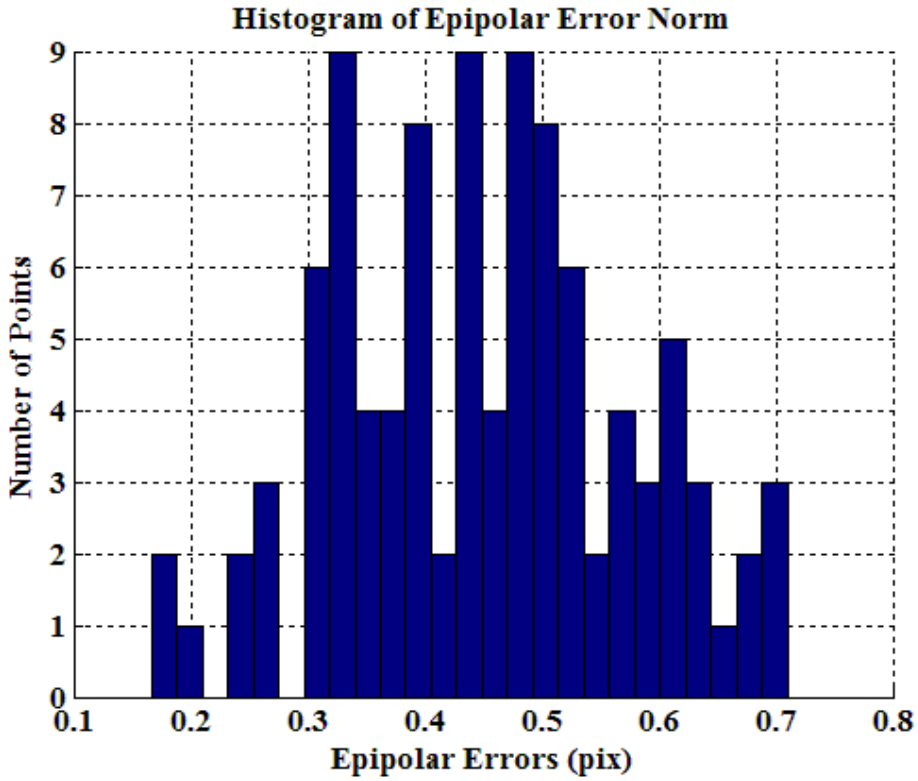

Figure 8.8. Histogram of Epipolar Errors(pix) for IDI points

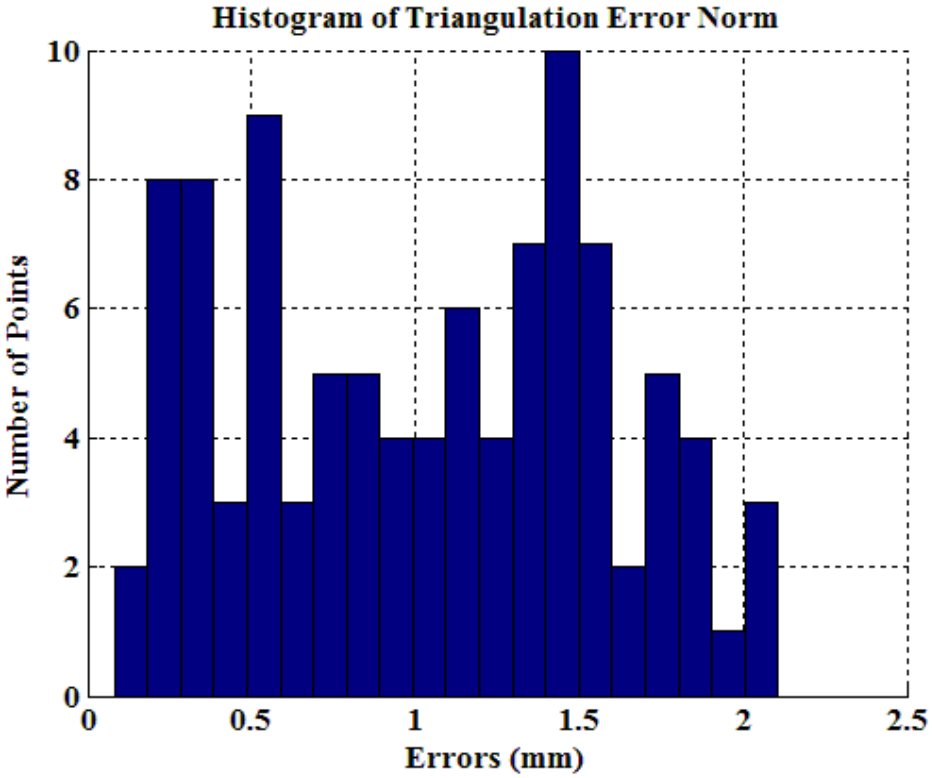

Figure 8.9. Histogram of Triangulation errors (mm) for IDI points.

8.12.3.Pick and Place Objects

The experimental setup consists of two Staubli robots on either end of a robotic transport unit (RTU). The details of the workcell are described in Chapter 3. The goal of our experiments is to find the 3D structure of a scene and be able to pick up the objects and manipulate them. The experimental setup consists of two Staubli robots each on either end of a robot transport unit.

Various objects are placed on the rotary table. The robot is commanded to pick up Biclops and move to various locations. The PTUs are also commanded to move to some desired orientation so as to be able to see the object. The cameras are commanded to takes pictures of the object. The position of the RTUs, the robot joint configurations and the pan and tilt angles of all the views are noted down.

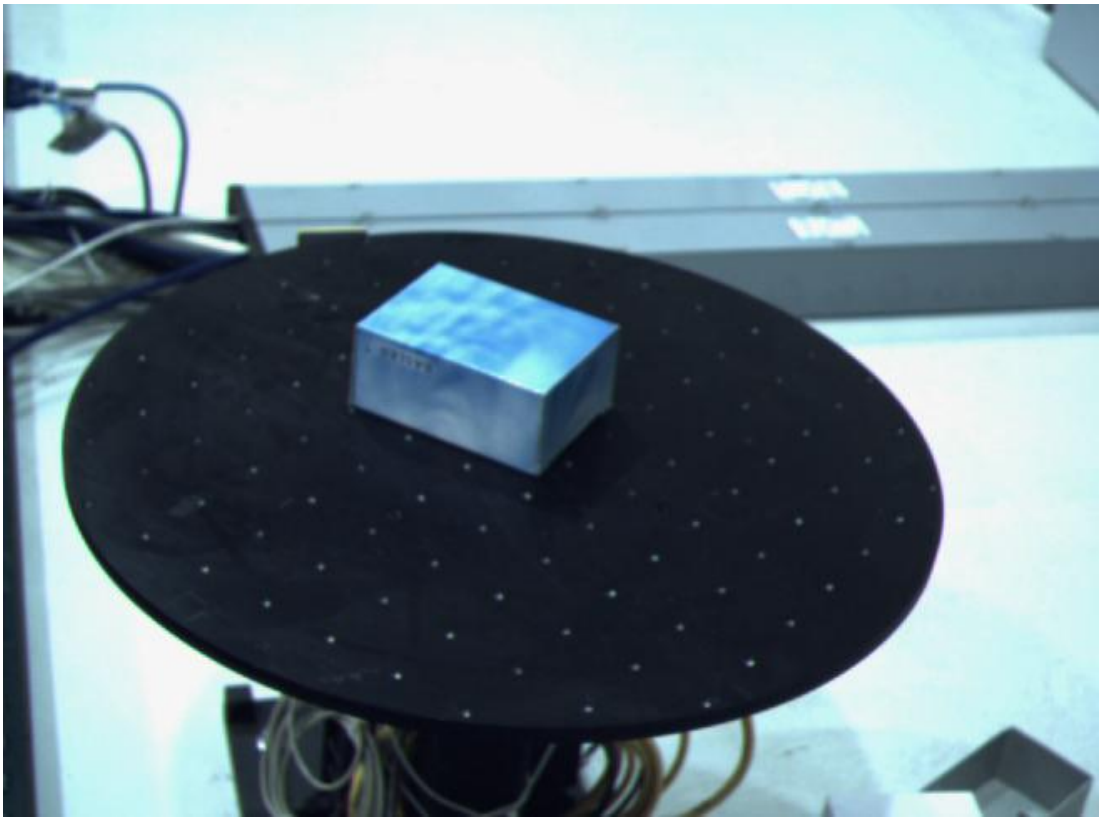

Figure 8.10. Image of a box from view 1

Using the noted values and the calibration parameters, the projection matrices (the position and orientation) of all the views are calculated. The images from various views are processed to detect the edges. Figure 8.10 and Figure 8.11 shows two such views of a box.

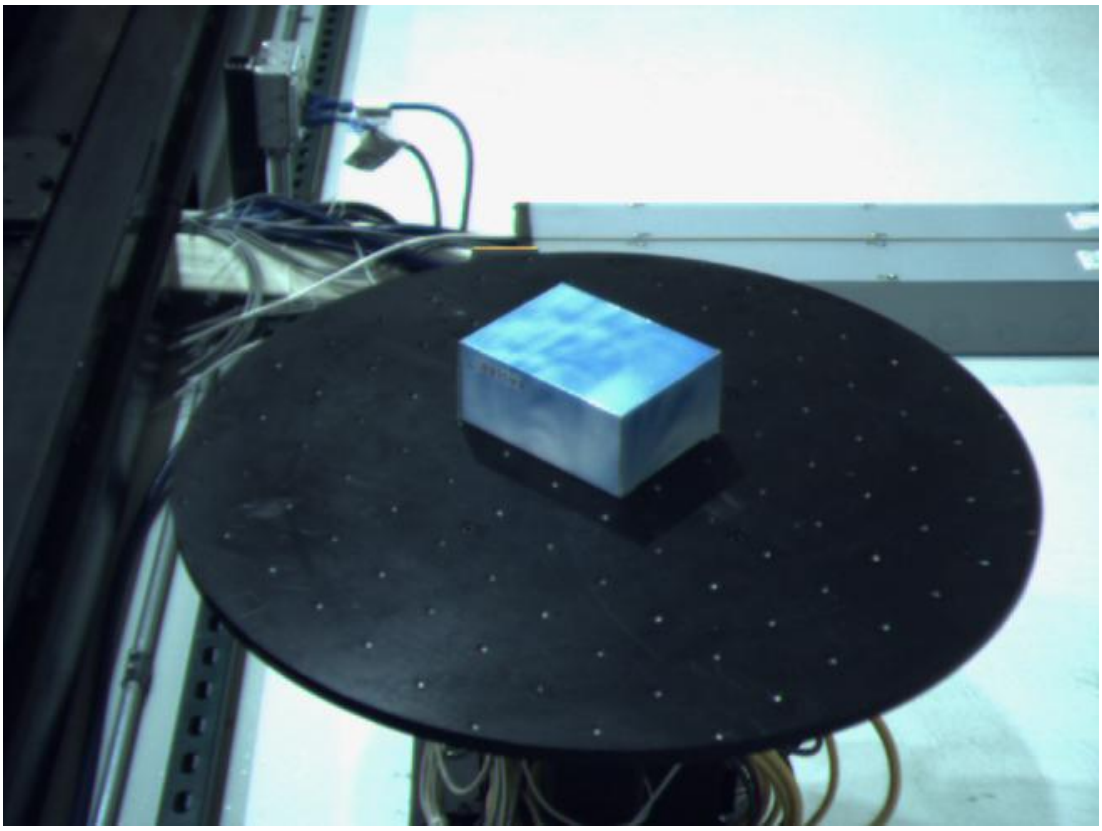

Figure 8.11. Image of a box from view 2

Figure 8.12 and Figure 8.13 show the edge processing of the two images. Note that there is lot of image clutter in the background. For the experiments we are only processing image in the region of interest (*i.e.*, the rotary table top).

The edge images are processed by the Hough transforms to find various line segments. The segments are localized to find the required EMPL lines.

Figure 8.14 and Figure 8.15 shows the EMPL lines located in the images. The IDI points are the various intersection points of the EMPL lines.

Table 8.1 and Table 8.2 show the IDI points found for the two views presented. They are arranged according to the correspondence.

We used the bundle correspondence algorithm to find the corresponding entities of the various IDI points and EMPL lines in all pairs of images. The corresponding entities are used to triangulate to get the initial 3D structure.

The 3D structure found is reprojected onto all views and refined using the bundle adjustment minimization. The 3D structure is represented in the base coordinate system of the robot holding the Biclops tool. This 3D structure is transferred to the other robots space and the robot uses this information to pick up the objects. For the sake of computing the errors some of the objects are placed at known locations and the 3D structure of the object is compared with the known values. Table 8.3 shows the triangulation errors of the eight points of the computed box model. Figure 8.16 shows the computed model of the box projected into the image view 1. The points are marked with labels to show them. The model is made up of list of points and the list of edges connecting two or more of those points.

Table 8.1. IDI points of view 1

X	Y
210.6899	193.7236
262.125	155.625
370.375	194.375
325.375	236.625
209.1891	242.4985
321.875	285.875
364.25	242.75

Table 8.2. IDI points of view 2

X	Y
273.375	200.625
336.375	165.375
430.875	207.125
370.5214	246.2504
274.4724	247.7511
370.25	294.25
429.625	255.875

Table 8.3. Errors in 3D points

Errors(mm)
2.3317
4.0694
0.5461
3.6288
3.3109
2.3775
0.8161
0.5343

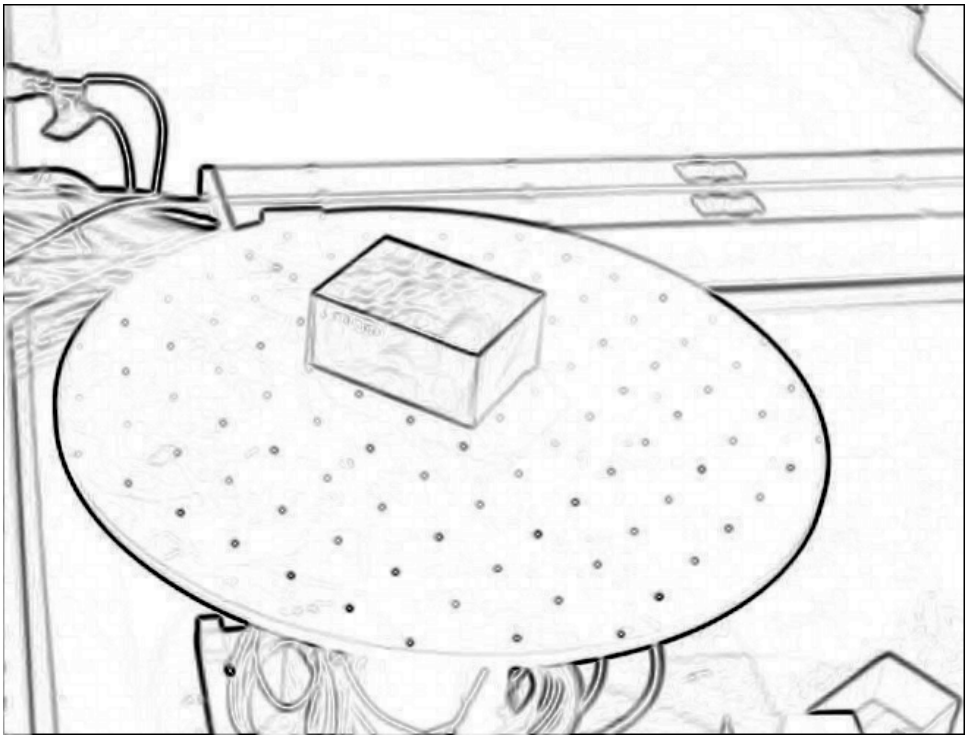

Figure 8.12. Negative Edge image of Image view 1.

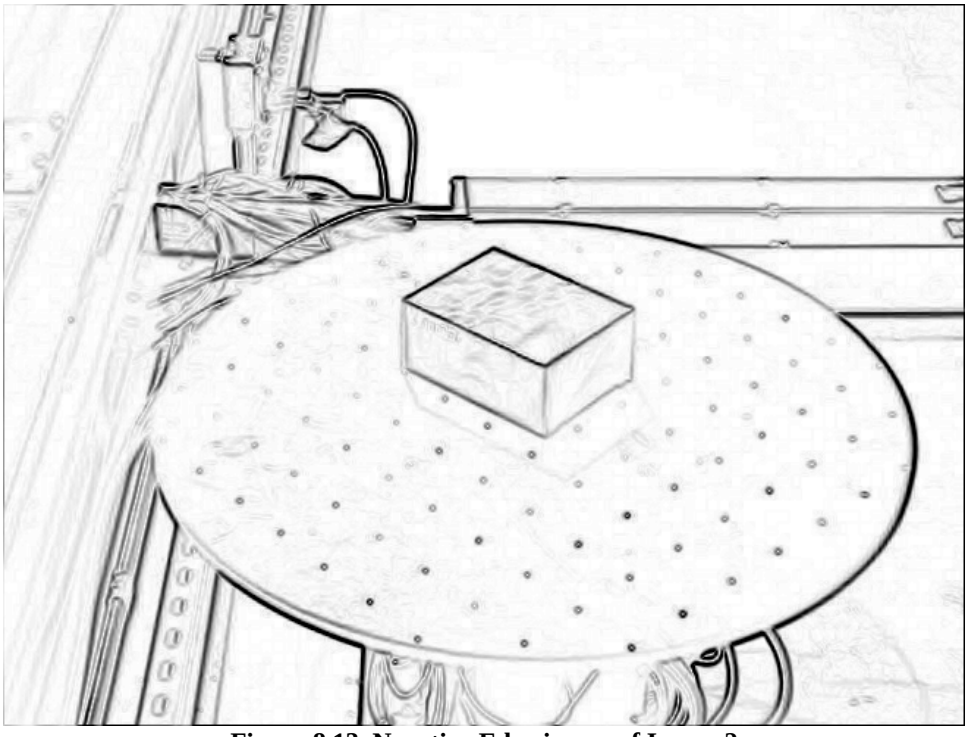

Figure 8.13. Negative Edge image of Image 2.

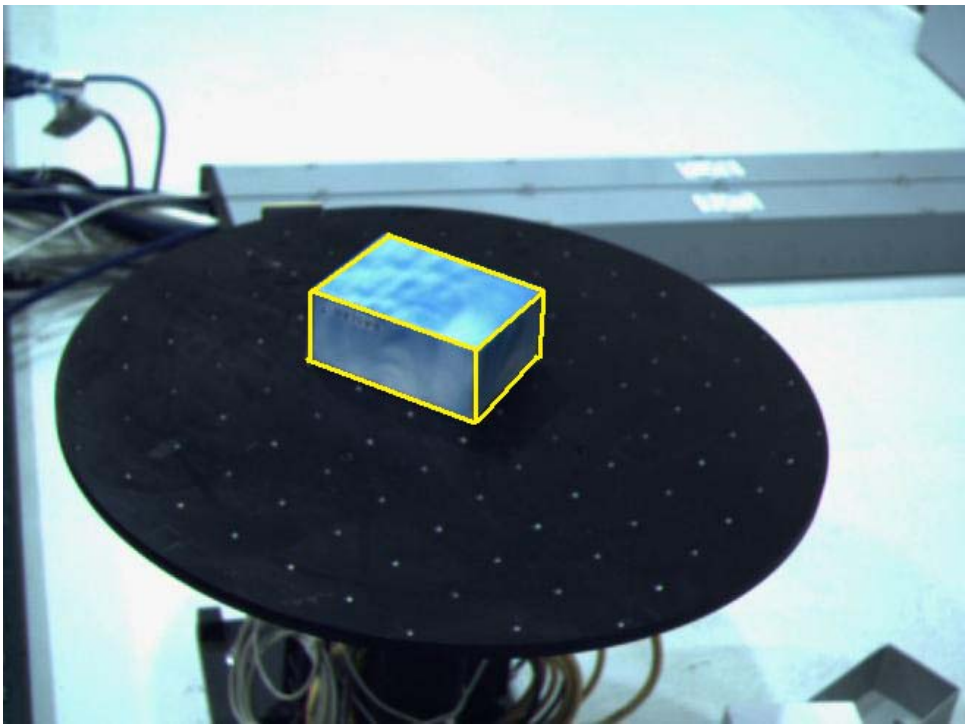

Figure 8.14. EMPL lines shown on Image view 1.

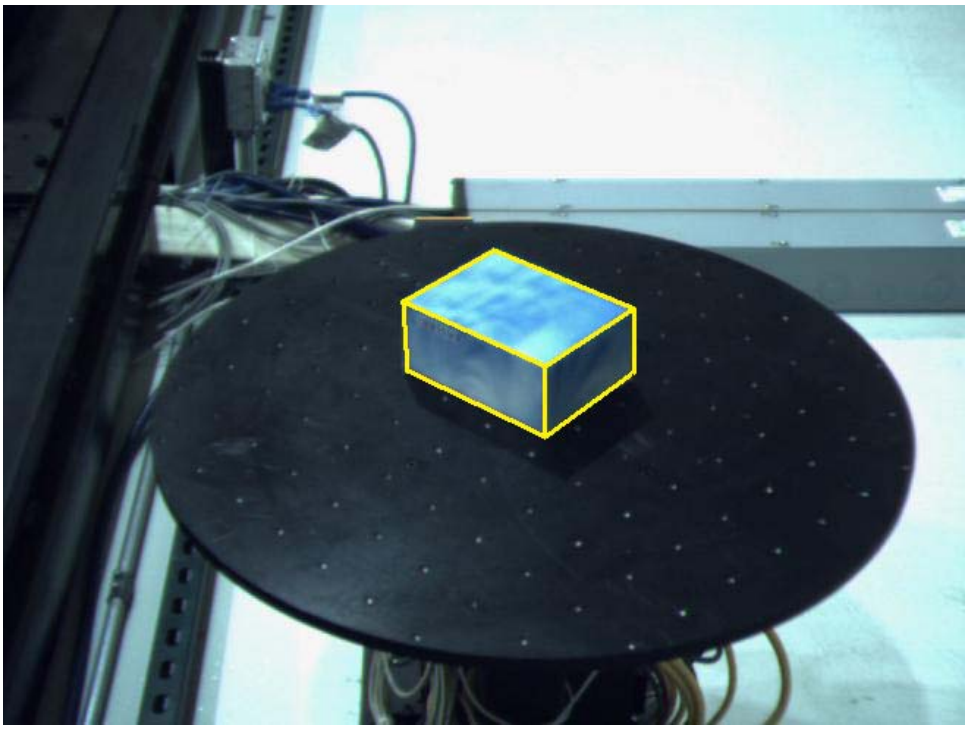

Figure 8.15. EMPL lines shown on Image 2.

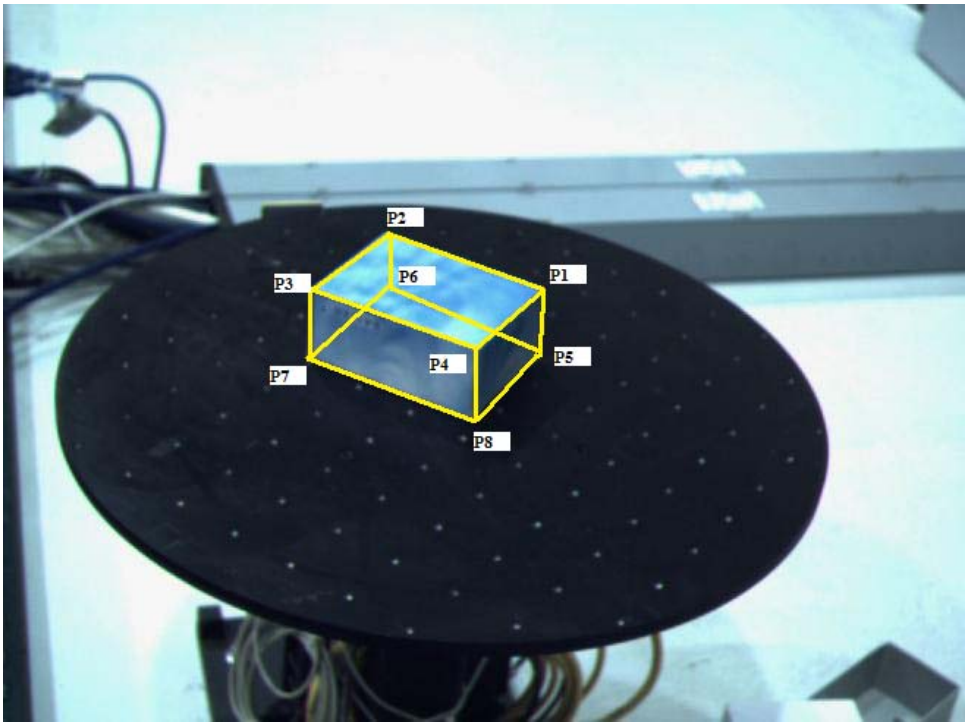

Figure 8.16. Model of the box projected

This experiment is repeated with various objects on the rotary table. Figure 8.17 shows a triangular object and the computed model is projected into the image view as shown in Figure 8.18.

Figure 8.19 shows a metal bracket object and the computed model is projected into the image view as shown in Figure 8.20. Figure 8.21 shows a cylinder object and the computed model is projected into the image view as shown in Figure 8.22. The cylinder is a special case. The image views are carefully planned to span around it with left and right camera views being close we have an approximate model made up of multiple points and lines approximating the cylinder.

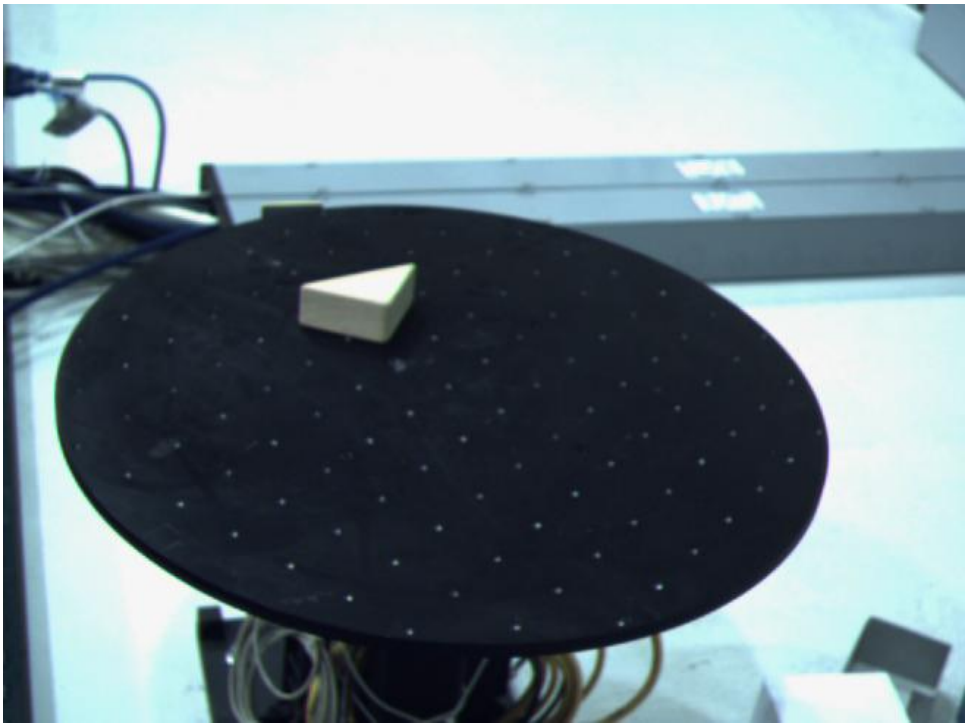

Figure 8.17. A triangular object on rotary table.

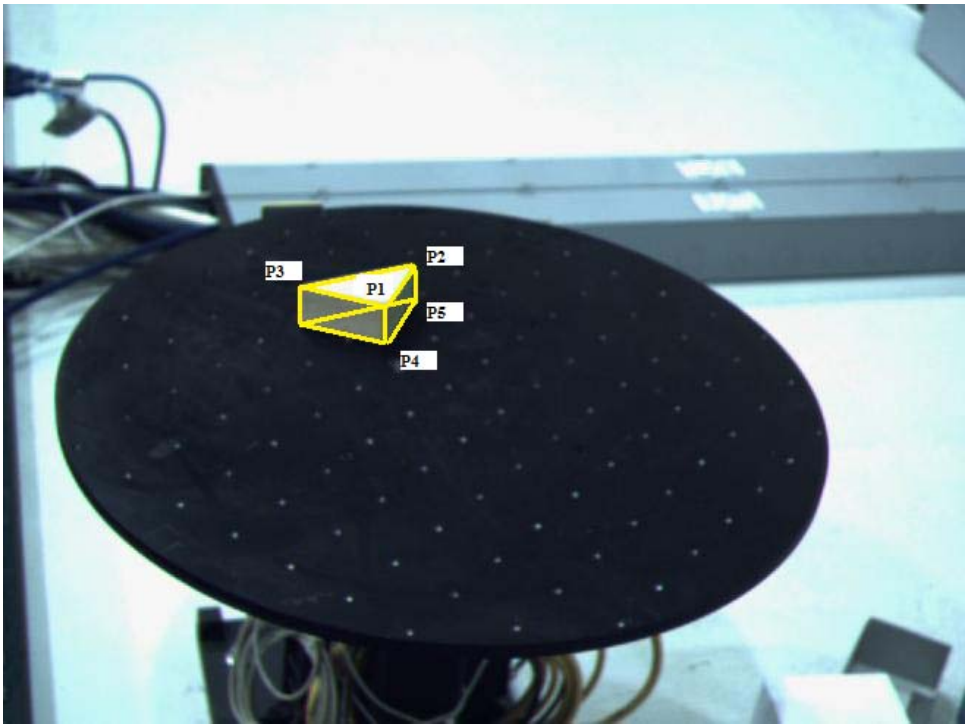

Figure 8.18. A triangular object Model projected.

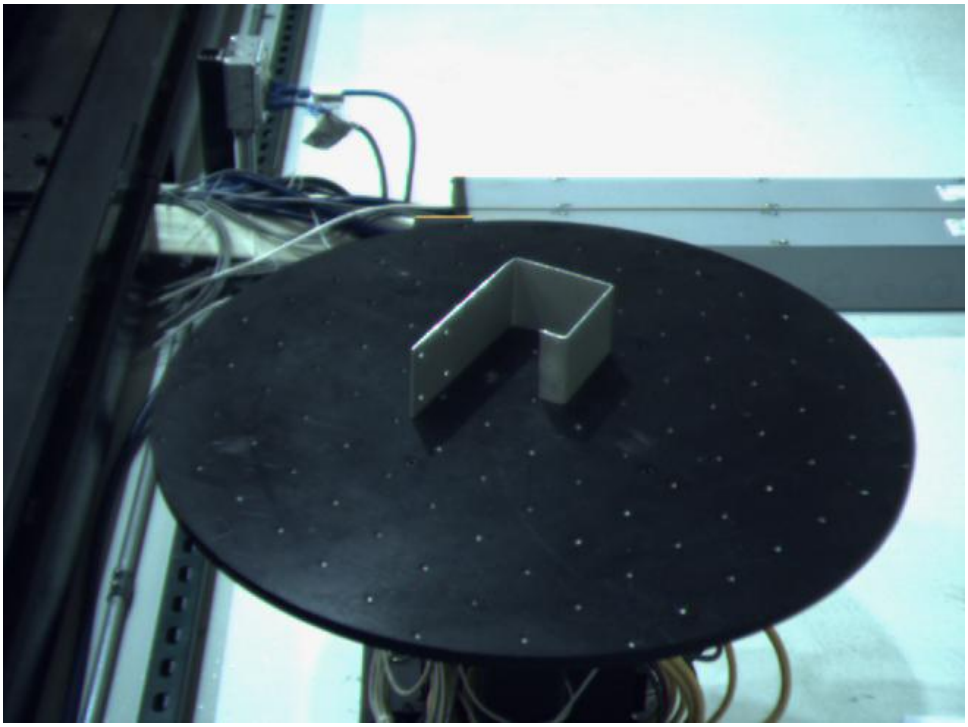

Figure 8.19. A metal bracket on rotary table.

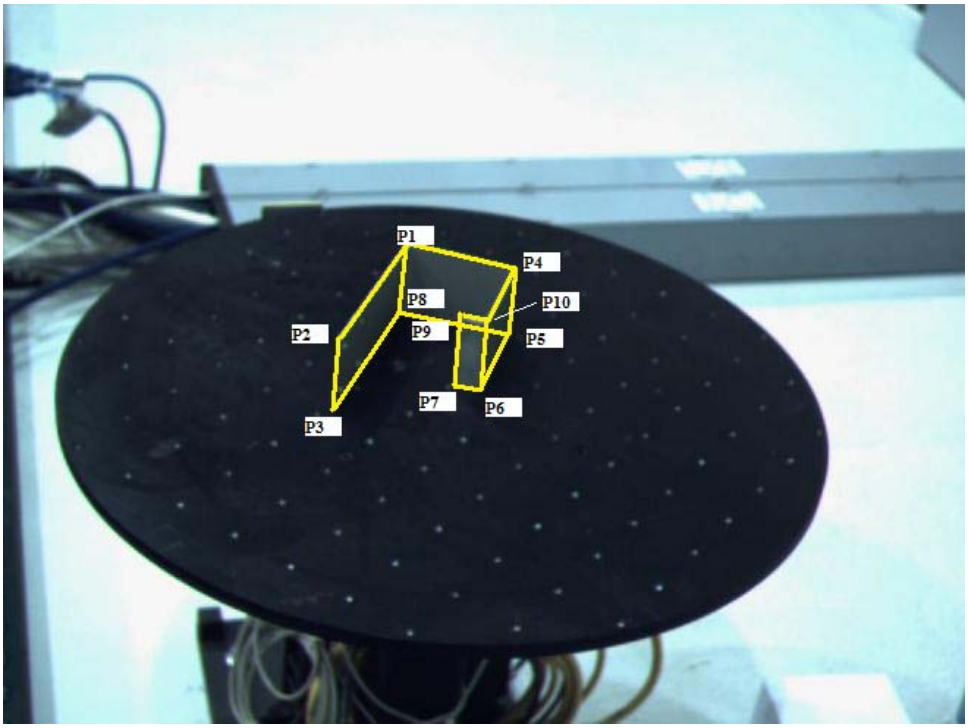

Figure 8.20. A metal bracket model projected.

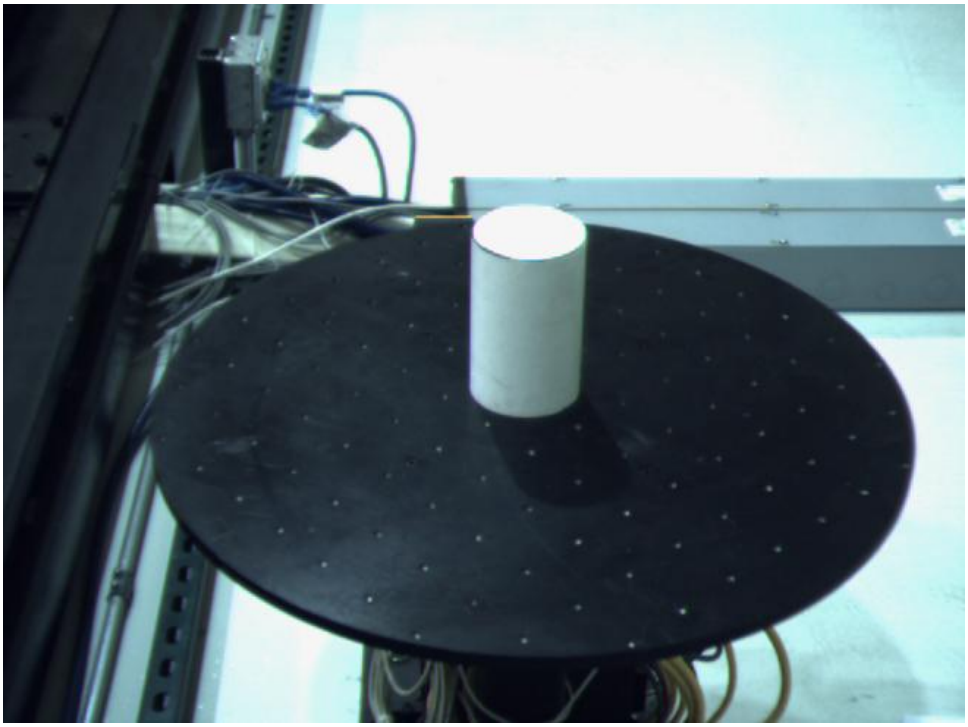

Figure 8.21. A cylindrical object on rotary table.

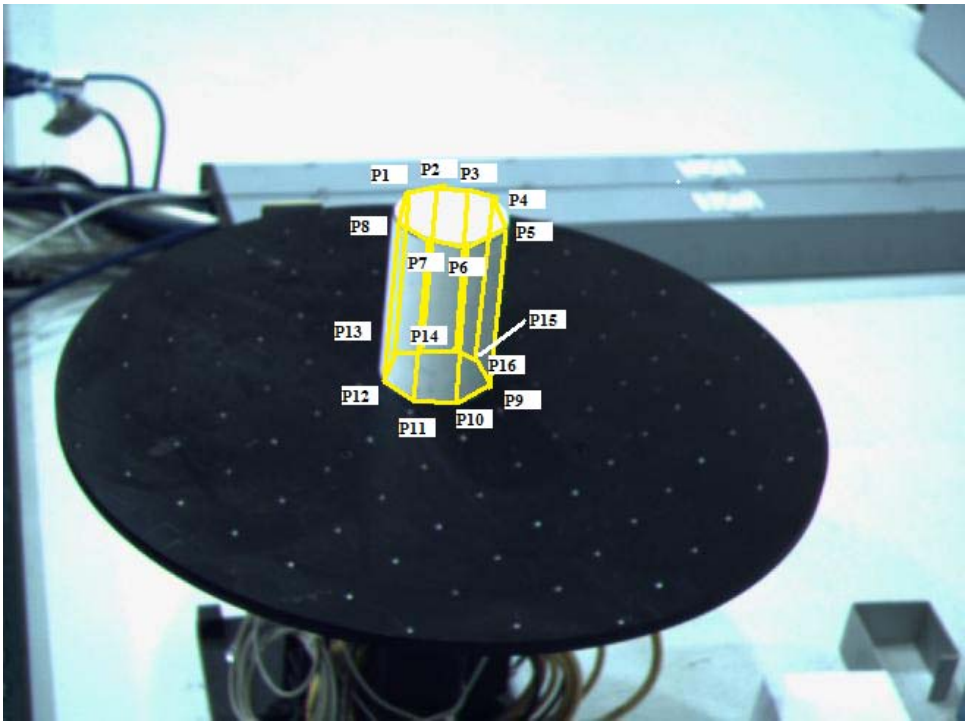

Figure 8.22. A cylindrical object model projected.

8.12.4.RMP Lane Following

A robotic mobile platform (RMP) from Segway is used and a vision system is built. The vision system consists of two cameras each mounted on a PTU. The Segway RMP is described in detail in Section 3.7.

A track is marked on the ground as shown in Figure 8.23. The track is built using straight line segments. The goal of this experiment is compute the location of the marked track and to follow the track.

For this experiment the PTUs are fixed and the cameras are pointing at the ground in front of it. The pictures of the track are captured. Figure 8.24 shows the image of the track as seen from one of the cameras. The images are processed to find the edges in them. The edge image is used and the Hough transforms are computed to locate line segments in the image. We localize the line segments to find the EMPL lines. The EMPL lines in the images are used to find the IDI points of the track. Figure 8.25 shows the EMPL lines and the IDI point found.

The cameras are calibrated individually and the transformation between them is found as before. The calibration is used in the bundle correspondence to find the corresponding entities in the images. These corresponding IDI points of the track are triangulated to find the 3D locations of the points on the road plane. This information is used by the Segway to plan its motion.

Figure 8.23 shows the RMP at the start of the trajectory. The images are used and the 3D information about the lines is computed. The RMP calculates the velocity and angular velocity need to follow the track.

The calibration information is also used to know where the future track is going to be. The RMP plans its path so as to keep the trajectory in its view all the time. It is to be noted that the cameras are looking at a distance. Thus a delay needs to be added in reacting to the trajectory. Also if the RMP waits to make a turn then by the time it reaches the intersection path it will no longer see the track.

To overcome this problem the RMP plans its motion by slowly moving in a curve from current direction to the final direction so as to keep the trajectory in view all the time. Figure 8.26, Figure 8.27 and Figure 8.28 shows the RMP following the track.

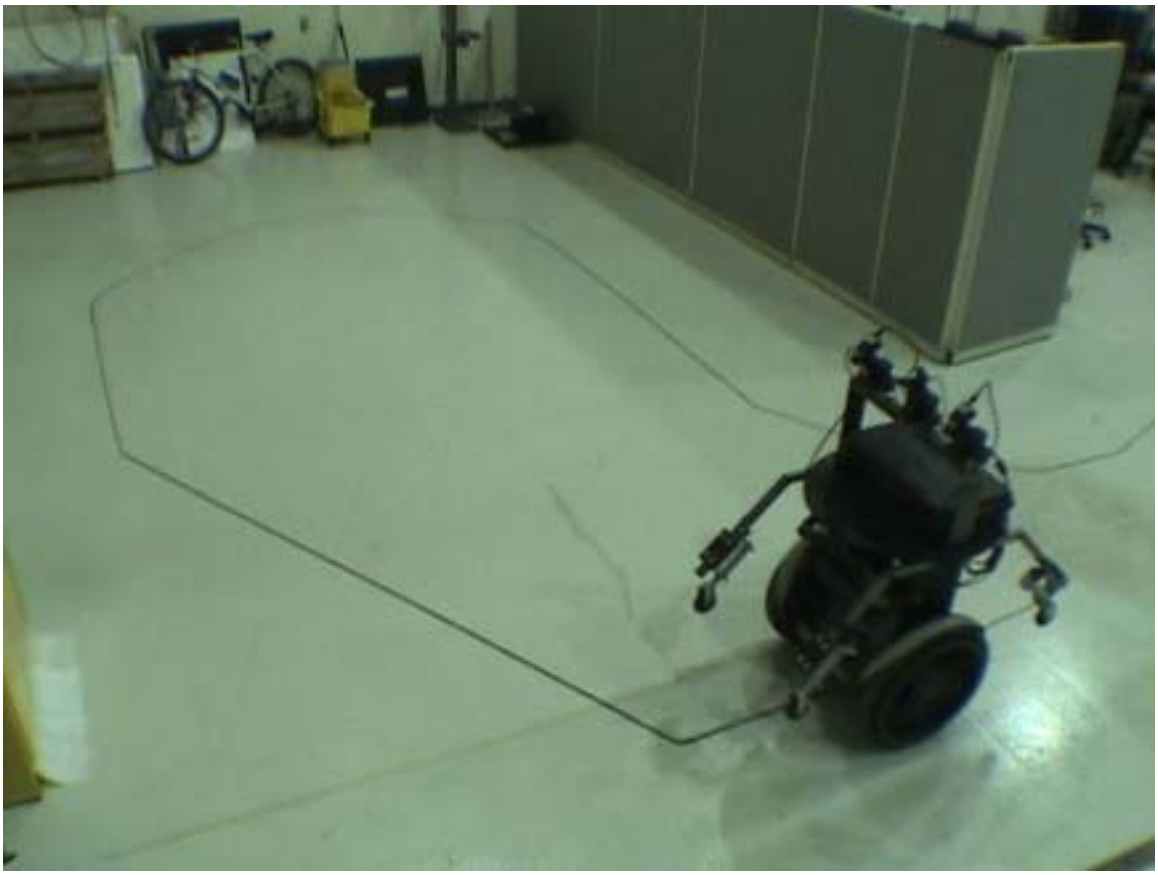

Figure 8.23. A Track marked on the ground.

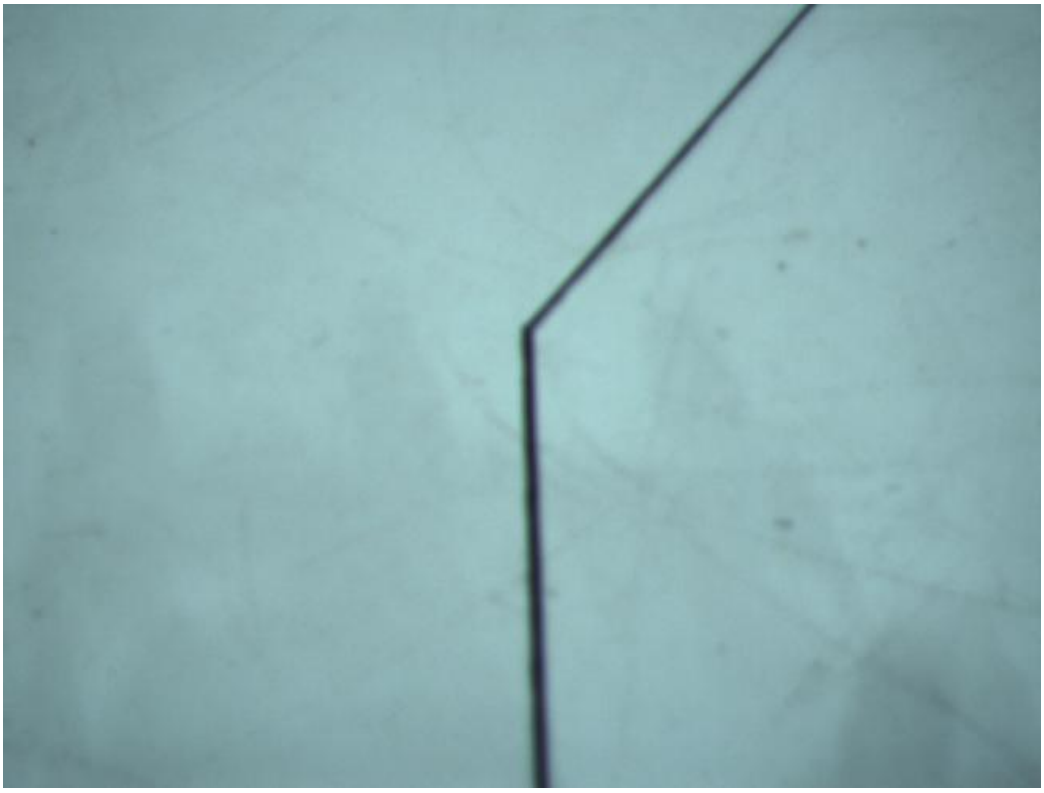

Figure 8.24. One of the images looking at the track.

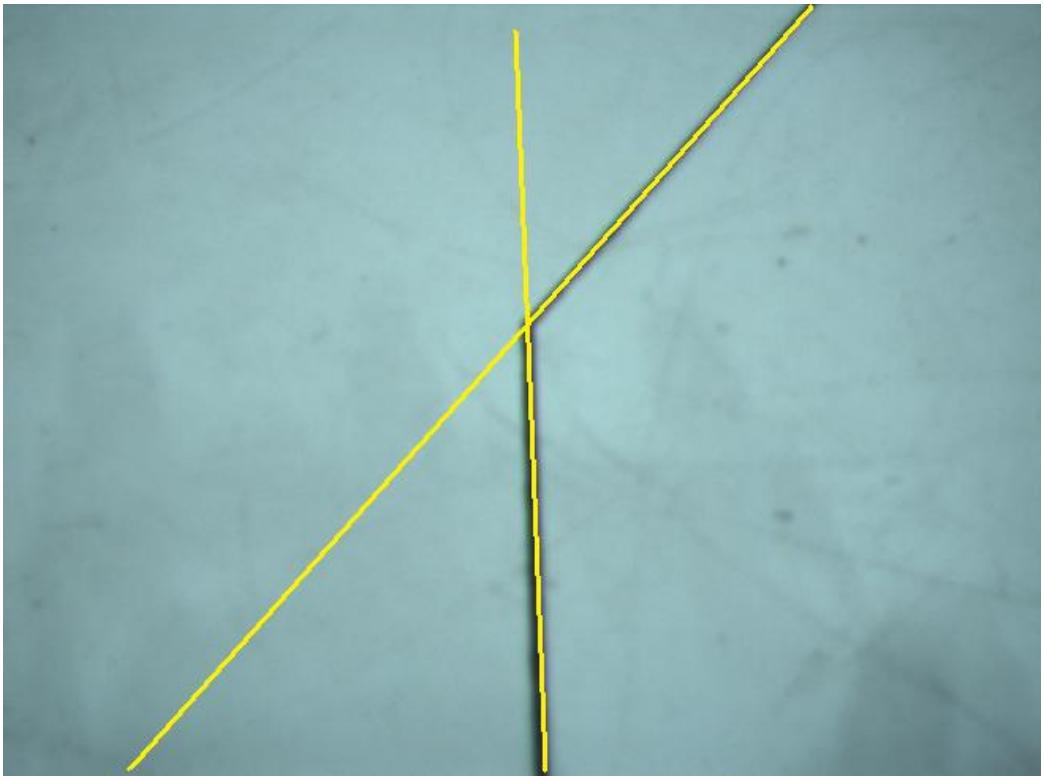

Figure 8.25. EMPL lines and the IDI point found for the track.

Figure 8.26. Segway RMP following the track.

Figure 8.27. Segway RMP following the track.

Figure 8.28. Segway RMP following the track.

8.13. Conclusion

Image processing is usually the first step for finding the 3D structure. The low level features found in the images need to be identified across various views. This is called correspondence. We have introduced a new approach, IDI points, as image features. We developed an automatic bundle correspondence algorithm that uses the calibration parameters of the image views and the topology information. The correspondence is established with this algorithm. Experiments are done to show the epipolar errors and the triangulation errors are small. Various objects are placed on a rotary table at known locations, pictures are taken, and the 3D structure of various objects is found. This structure information is transferred to another robots' space and the objects

are picked up by the second robot. Experiments are done using a mobile robot to use the features and locate a track in 3D and follow it.

Chapter 9. Conclusion

Many applications, e.g., motion planning, virtual reality, CAD, vehicle navigation, object recognition, photogrammetry, remote sensing, etc., all require a geometrical representation of the three dimensional structure of a scene. In this dissertation we studied the problem of determining the 3D structure of a scene given images of it from various views.

A robotic tool called Biclops, a two camera directed vision system is built. Each camera is mounted on a Pan-Tilt Unit which can be independently controlled. This eye in hand system is used to find the structure of the scene. There are two robots mounted on a robot transport unit.

The 3D structure of the scene is necessary to manipulate the objects in the scene. We developed a new image feature called IDI points which are intersections of a pair of EMPL lines or an EMPL line and a Curve. An automatic bundle adjustment correspondence method is developed which is used to match up the points and lines in one image to the other. The corresponding features the lines and points are used to determine the 3D structure of the scene. The 3D structure is transferred to the other robot space for it to manipulate the objects in the scene.

The accuracy of 3D structure of the scene is dependent on the accuracy of the various parameters of the underlined equipment. We go a step further and improved the accuracy of the parameters of various equipments used. We developed a new method called ViCKi (virtual close loop kinematic method) to calibrate the industrial robots and the RTU equipment. We also calibrated the pan tilt units of the Biclops. Various experiments are done to show the accuracy of the calibration methods.

The calibrated equipment is used to find the 3D structure of a scene. The robot used the 3D structure to pick up the objects.

9.1. Limitations

We have worked mostly with man-made objects that have straight edges. Occasionally we used some curved objects like cylinders. The cylinders usually do not have any straight edges but we will see straight edges in image views. The straight edges that we find usually do not correspond at all since they are not real object edges but they are only perceived as edges. To overcome this difficulty, we have to take pictures of the object from relatively close view points so the errors in the correspondence are small and we will be approximating the cylinder with multifaceted polyhedra.

We have used the curves only to locate IDI points which are at the intersection of the curve with EMPL lines. Occasionally we used maximum inflection points as IDI points but they only correspond to other images taken from closer view points.

The approach is limited to facetted objects with edges only, objects like sphere, cylinder etc which do not correspond to this form are only approximated closely.

9.2. Future Directions

We have not included the radial distortion of the cameras. Even though the distortion is not that significant we expect to see an improvement in the accuracy of the computed 3D structure when calibrated with distortion parameters. One of the limitations was that our scene was stationary. We would like to add a rigid body motion to our objects in the scene and add those parameters into optimization function. An interesting experiment would be to use our bundle correspondence online and track an object as it moves through the scene.

References

[1] O. Faugeras and B. Mourrain, “About the correspondence of points between n images,” In IEEE Workshop on Representation of Visual Scenes, MIT, Boston, MA, IEEE CS Press, 1995, pp. 37-44.

[2] O. Faugeras and B. Mourrain, “On the geometry and algebra of the point and line correspondences between n images,” Proc. 5th ICCV, Cambridge, Massachusetts, IEEE CS Press, 1995, pp. 951-956.

[3] M. Oskarsson, A. Zisserman, and K. Åström, “Minimal Projective Reconstruction for Combinations of Points and Lines in Three Views.” Proc. of the British Machine Vision Conference, 2002.

[4] A. Bartoli and P. Strum, “Multiple-View Structure and Motion from Line Correspondences,” ICCV'03 - Proc. IEEE International Conference on Computer Vision, vol. 1, Nice, France, October 2003, pp. 207-212.

[5] R. Hartley, “Projective Reconstruction from Line Correspondences,” Proc. CVPR, Seattle, Washington, USA, 1994.

[6] R.J. Holt and A.N. Netravali, “Motion and structure from line correspondences under orthographic projection,” Int. J. of Imaging Systems and Technology, 1997, 8(3):301-312.

[7] L. Quan and T. Kanade, “Affine structure from line correspondences with uncalibrated affine cameras,” IEEE Transactions on Pattern Analysis and Machine Intelligence, August 1997, 19(8):834- 845.

[8] L. Quan and T. Kanade, “A factorization method for affine structure from line correspondences,” CVPR’96, San Francisco, CA, June 1996, pp. 803-808.

[9] Taylor and D. Kriegman, “Structure and motion from line segments in multiple images,” PAMI, 1995, 17(11).

[10] Rikard Berthilsson, Kalle Åström, “Reconstruction of 3D-Curves from 2D-Images Using Affine Shape Methods for Curves,” Conference on Computer Vision and Pattern Recognition (CVPR '97), June 17 - 19, Puerto Rico, 1997.

[11] Kahl and A. Heyden, “Using Conic Correspondence in Two Images to Estimate the Epipolar Geometry,” In Proceedings of the International Conference on Computer Vision, 1998.

[12] Yermiyahu Kaminski, Michael Fryers, “Multiple View Geometry of Non-planar Algebraic Curves,” International Conference on Computer Vision (ICCV'01) Vol. 2, July 07 - 14, 2001, Vancouver, B.C., Canada.

[13] T. Papadopoulo and O. Faugeras, “Computing structure and motion of general 3d rigid curves from monocular sequences of perspective images,” Technical Report RR-2765, INRIA, 1995.

[14] Q.T. Luong and O. Faugeras, “The Fundamental Matrix: Theory, Algorithms, and Stability Analysis,” The International Journal of Computer Vision, January 1995, 17(1):43-76.

[15] O. Faugeras, Q. T. Luong, and S. J. Maybank, “Camera self-calibration: Theory and experiments,” In Proc. European Conference on Computer Vision, Santa Margherita Ligure, Italy, 1992, pages 321-334.

[16] Q.T. Luong and O. Faugeras, “Self-calibration of a moving camera from point correspondences and fundamental matrices,” IJCV, 1997, 22(3):261-289.

[17] R. K. Lenz and R. Y. Tsai, "Techniques for calibration of the scale factor and image center for high accuracy 3D machine vision metrology," in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 10, no. 5, pp. 713-720, Sept. 1988.

[18] M. Penna, "Camera Calibration: A Quick and Easy Way to Determine the Scale Factor," *in IEEE Trans. Pattern Anal. Machine Intelligence*, vol. 12, no. 12, pp. 1240-1245, 1991.

[19] Guruprasad Shivaram, Guna Seetharaman, "A New Technique for Finding the Optical Center of Cameras," *ICIP*, vol. 2, pp. 167-171, 1998.

[20] B. Caprile and V. Torre, "Using vanishing points for camera calibration," *Int. Journal of Computer Vision*, vol. 4, no. 2, pp. 127-140, March 1990.

[21] O.D. Faugeras, Q.T. Luong, and S. J. Maybank, "Camera Self-Calibration : Theory and Experiments," *In Proceedings of the Second European Conference on Computer Vision*, pp. 321-334, Santa Margherita Ligure, Italy, May. 1992.

[22] Q. T. Luong and O. D. Faugeras, "Self-calibration of a moving camera from point correspondences and fundamental matrices," *International Journal of Computer Vision*, vol. 22, no. 3, pp. 261-289, 1997.

[23] R. Hartley, "Self-calibration from multiple views with a rotating camera," *In Proc. Third European Conf. on Computer Vision*, pp. 471-478, 1994.

[24] S.J. Maybank and O.D. Faugeras, "A Theory of Self Calibration of a Moving Camera," *Int'l Journal of Computer Vision*, vol. 8, no. 2, pp. 123-151, 1992.

[25] D. C. Woo and D. W. Capson, "3D visual tracking using a network of low cost pan/tilt cameras," presented at *Canadian Conference on Electrical and Computer Engineering Conference Proceedings*, 2000.

[26] A. Basu and K. Ravi, "Active camera calibration using pan, tilt and roll," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 27, pp. 559-66, 1997.

[27] S. Fry, M. Bichsel, P. Muller, and D. Robert, "Tracking of flying insects using pan-tilt cameras," *Journal of Neuroscience Methods*, vol. 101, pp. 59-67, 2000.

[28] J. Davis and X. Chen, "Calibrating pan-tilt cameras in wide-area surveillance networks," *In Proc. of ICCV 2003*, Vol 1, pp. 144-150, October 2003.

[29] Z.S. Roth, B.W. Mooring, and B. Ravani, "An Overview of Robot Calibration," *IEEE J. of Robotics and Automation*, 3(5): pp. 377-384, 1987.

[30] J. M. Hollerbach, "A survey of kinematic calibration," in *the Robotics Review 1*, O. Khatib, J. J. Craig, and T. Lozano-Pérez, Eds. MIT Press, Cambridge, MA, pp. 207-242, 1989.

[31] B.W. Mooring, Z.S. Roth, and M.R. Driels, Fundamentals of Manipulator Calibration: John Wiley & Sons, NY, pp. 135-140, 1991.

[32] B.W. Mooring, Z.S. Roth, and M.R. Driels, Fundamentals of robotic calibration. John Willey and Sons, pp: 221-225, 1991.

[33] J.M. Hollerbach, The Calibration Index and Taxonomy for Robot Kinematic Calibration Methods. International Journal of Robotics Research, 1996. 15(12): pp. 573-591.

[34] M.R. Driels, L.W. Swayze, and L.S. Potter, Full-pose calibration of a robot manipulator using a coordinate measuring machine. Int. J. Adv. Manufacturing. Tech no l., 8: pp. 34-41, 1993.

[35] D.W. Osborn, and W.S. Newman, "A New Method for Kinematic Parameter Calibration via Laser Line," In Proceedings of International Conference on Robotics and Automation. 1993: IEEE. Vol. 2, pp. 160-165.

[36] M. Ikits, and J.M. Hollerbach., "Kinematic Calibration Using a Plane Constraint," In Proceedings of International Conference on Robotics and Automation. 1997: IEEE. pp. 3191-3196.

[37] H. Zhuang, S.H. Motaghedi, and Z.S. Roth, "Robot Calibration with Planar Constraints," *Proc. IEEE International Conference of Robotics and Automation*, Detroit, Michigan, 1999, pp. 805-810.

[38] Bennet, D.J. and J.M. Hollerbach, "Autonomous calibration, of single-loop closed kinematic chains, formed by manipulators with passive endpoint constraints," IEEE Trans. on Robotics and Automation, 1991, 7: pp. 597-606.

[39] M.A. Meggiolaro, G. Scriffignano, and S. Dubowsky, "Manipulator Calibration Using a Single Endpoint Contact Constraint." *Proceedings of the 26th Biennial Mechanisms and Robotics Conference of the 2000 ASME Design Engineering Technical Conferences*, Baltimore, MD, September 2000.

[40] L.J. Everett, and C.Y. Lin, "Kinematic Calibration of Manipulators with Closed Loop Actuated Joints," *Proc. IEEE International Conf. On Robotics and Automation*, Philadelphia, pp.792-797. 1988

[41] L. Giugovaz, and J.M. Hollerbach, "Closed loop kinematic calibration of the Sarcos Dexterous Arm," in Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, Munich, submitted, Sept. 12-16, 1994.

[42] M. Kass, A. Witkin and D. Terzopoulos, "Snakes: Active Contour Models," International Journal of Computer Vision, 1(4): 321-331, 1987.

[43] J. Ivins and J. Porrill, "Everything you always wanted to know about snakes (but were afraid to ask)," AIVRU Technical Memo #86, Artificial Intelligence Vision Research Unit, University of Sheffield, March 2000.

[44] Williams and M. Shah, "A fast algorithm for active contours and curvature estimation," CVGIP (Image Understanding), 55(1):14-26, 1992.

[45] H. Schaub and C. Smith, "Color snakes for dynamic lighting conditions on mobile manipulation platforms," Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003.

[46] P. E. Debevec, C. J. Taylor and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry and image-based approach," Proc. of SIGGRAPH, 11-20, 1996.

[47] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision. Cambridge University Press, June 2000.

[48] Richard Hartley and Peter Sturm, "Triangulation," 6th International Conference on Computer Analysis of Images and Patterns, Prague, Czech Republic pages 190-197, Sep 1995.

[49] Olivier Faugeras, Three-Dimensional Computer Vision: A geometric Viewpoint, MIT Press, 1996.

[50] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2000.

[51] Olivier Faugeras, Quang-Tuan Luong, T. Papadopoulo, "The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications," MIT Press, 2001.

[52] P. Torr and D. Murray, "The development and comparison of robust methods for estimating the fundamental matrix," International Journal of Computer Vision, 24(3):271-300, 1997.

[53] Q.-T. Luong, R. Deriche, O.D. Faugeras, and T. Papadopoulo, "On determining the fundamental matrix: analysis of different methods and experimental results," Technical Report 1894, INRIA Sophia-Antipolis, 1993.

[54] Q.-T. Luong and O.D. Faugeras, "Determining the Fundamental matrix with planes: unstability and new algorithms," In Proc. Conference on Computer Vision and Pattern Recognition, pages 489--494, New-York, 1993.

[55] M. Z. Brown, D. Burschka and G. D. Hager, "Advances in computational stereo," IEEE Trans. PAMI, vol. 25(8), pp.993-1008, 2003.

[56] T. Papadopoulo and O. Faugeras, "Computing structure and motion of general 3d rigid curves from monocular sequences of perspective images," Technical Report RR-2765, INRIA, 1995.

[57] Long Quan, "Conic reconstruction and correspondence from two views," IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(2):151-160, 1996.

[58] Kalle Åström, Roberto Cipolla, Peter Giblin, "Generalized Epipolar Constraints," In Proc of 4th European Conf. on Computer Vision, Cambridge, UK, 1996.

[59] J. J. Mor'e, "The Levenberg-Marquardt Algorithm: Implementation and Theory," *Numerical Analysis, Lecture Notes in Mathematics*, Springer Verlag, Vol. 630, pp. 105-116, 1977.

[60] J.J. Craig, Introduction to Robotics. 1986: Addison-Wesley, MA.

[61] J. Denavit, and R.S. Hartenberg, *A kinematic notation for lower-pair mechanisms based on matrices.* ASME Journal of Applied Mechanics, 1955: pp. 215-221.

[62] S.A. Hayati, "Robotic arm geometric parameter estimation," In 22nd IEEE International Conference on Decision and Control. 1983: IEEE. Vol. 3 pp.1477-1483.

[63] M.A. Branch, T.F. Coleman, and Y. Li, "A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-Constrained Minimization Problems," SIAM Journal on Scientific Computing, Vol. 21, Number 1, pp 1-23, 1999.

[64] R.H. Byrd, R.B. Schnabel, and G.A. Shultz, "Approximate Solution of the Trust Region Problem by Minimization over Two-Dimensional Subspaces," Mathematical Programming, Vol. 40, pp 247-263, 1988.

[65] www.staubli.com

[66] Chandrasekhar Gatla, Ron Lumia, John Wood, Greg Starr, "An Automated Method to Calibrate Industrial Robots Using a Virtual Closed Kinematic Chain", *IEEE Transactions on Robotics*, Volume 23, Issue 6, Dec. 2007 Page(s):1105 – 1116.

[67] Prewitt J.M.S. Object enhancement and extraction. B.S. Lipkin, A. Rosenfeld (Eds.), Picture Processing and Psychopictorics, Academic Press, New York, 1970.

[68] Canny, John, "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 6, 1986, pp. 679-698.

[69] Parker, James R., “Algorithms for Image Processing and Computer Vision”, New York, John Wiley & Sons, Inc., 1997, pp. 23-29.

[70] P.V.C. Hough, “Machine Analysis of Bubble Chamber Pictures”, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

[71] Duda, R. O. and P. E. Hart, “Use of the Hough Transformation to Detect Lines and Curves in Pictures,” *Comm. ACM, Vol. 15*, pp. 11–15 (January, 1972).

[72] R. S. Stephens, Probabilistic approach to the Hough transform, Image and Vision Computing, v.9 n.1, p.66-71, Feb. 1991.

[73] D. J. Langridge. “Curve encoding and detection of discontinuities,” Computer Vision, Graphics and Image Processing, 20(1):58-71, 1987 pages 46 and 47.

[74] G. Medioni and Y. Yasumoto. “Corner detection and curve representation using cubic b-splines,” Computer Vision, Graphics and Image Processing, 39(3):279-290, 1987.

[75] F. Mokhtarian and R. Suomela. Robust image corner detection through curvature scale space. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(12):1376-1381, 1998.

[76] R. M. Haralick and L. G. Shapiro. Computer and robot vision, volume 1. Adison-Wesley, 1993.

[77] J. Cooper, S. Venkatesh and L. Kitchen. Early jump-out corner detectors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(8):823-828, 1993.

[78] L. Kitchen and A. Rosenfeld. Gray-level corner detection. Pattern Recognition Letters, 1(2):95-102, 1982.

[79] H. Wang and M. Brady. Real-time corner detection algorithm for motion estimation. Image and Vision Computing, 13(9):695-703, 1995.

[80] H. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover .tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University, September, 1980.

[81] C. Harris and M. Stephens. A combined corner and edge detector. Alvey Vision Conference, 147-151. 1988.

[82] J. Shi and C. Tomasi. Good features to track. 9th IEEE Conference on Computer Vision and Pattern Recognition. Springer, 1994.

[83] Z. Zheng, H. Wang and E. K. Teoh. Analysis of gray level corner detection. Pattern Recognition Letters, 20(2):149-162, 1999.

[84] D. G. Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):91-110, 2004.

[85] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. 8th IEEE International Conference on Computer Vision, 525-531. Springer, Vancouver, Canada, 2001.

[86] S. M. Smith and J. M. Brady. SUSAN - a new approach to low level image processing. International Journal of Computer Vision, 23(1):45-78, 1997.

[87] M. Trajkovic and M. Hedley. Fast corner detection. Image and Vision Computing, 16(2):75-87, 1998.

[88] O. Faugeras, Three-Dimensional Computer Vision, a Geometric Viewpoint, MIT Press, 1993.

[89] B. Triggs, P. McLauchlan and R. Hartley and A. Fitzgibbon, “Bundle Adjustment — A Modern Synthesis,” ICCV '99: Proceedings of the International Workshop on Vision Algorithms. Springer-Verlag. pp. 298-372, 1999.

[90] Chandrasekhar Gatla, Ron Lumia, John Wood, Greg Starr, “An Efficient Method to Compute Three Fingered Planar Object Grasps Using Active Contour Models,” IROS 2004, IEEE/RSJ Int'l. Conf. on Intelligent Robots and Systems, Sendai, JAPAN, Sept. 28-Oct. 2, 2004.