VISUALLY GUIDED ROBOTIC ASSEMBLY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ONUR ŞERAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF MECHANICAL ENGINEERING

SEPTEMBER 2003

ABSTRACT

VISUALLY GUIDED ROBOTIC ASSEMBLY

ŞERAN, Onur
Ms., Department of Mechanical Engineering
Supervisor: Assist. Prof. İlhan E. KONUKSEVEN

September 2003, 137 pages

This thesis deals with the design and implementation of a visually guided robotic assembly system. Stereo imaging, three dimensional location extraction and object recognition will be the features of this system. This thesis study considers a system utilizing an eye-in-hand configuration. The system involves a stereo rig mounted on the end effector of a six-DOF ABB IRB-2000 industrial robot. The robot is controlled by a vision system, which uses open-loop control principles. The goal of the system is to assemble basic geometric primitives into their respective templates.

Keywords: Eye-In-Hand, Stereo Imaging, 3D Position Extraction, and Recognition

ÖZ

GÖRÜNTÜ GÜDÜMLÜ ROBOTİK MONTAJ

ŞERAN , Onur
Yüksek Lisans , Makine Mühendisliği Bölümü
Tez Yöneticisi : Yrd. Doç . Dr . İlhan E. KONUKSEVEN

Eylül 2003, 137 sayfa

Bu çalışma görüntü yönlendirmeli bir robotik montaj sisteminin tasarımı ve uygulaması üzerinedir. Stereo görüntüleme, üç boyutlu konum algılama ve nesne tanımlama bu sistemin ana özellikleridir. Bu tez çalışmasında, "elde-göz" kurulumunu kullanan bir sistem ele almıştır. Sistem, altı eksenli ABB IRB-2000 endüstriyel robotun tutucusuna monte edilmiş bir stereo-kamera düzeneği içermektedir. Robot açık-döngü kontrolü prensiplerini kullanan bir görüntü işleme sistemi tarafından kontrol edilmektedir. Sistemin amacı, temel geometrilere sahip parçaları, uygun şablonlara monte etmektir.

Anahtar Kelimeler : Elde-Göz, Ikili Görüntüleme, Üç Boyutlu Konum Algılama ,Tanımlama

To My Family

ACKNOWLEDGMENTS

I express sincere appreciation to Asst. Prof. Dr. İlhan E. Konukseven for his guidance, support and insight throughout this research. Thanks also go to Asst. Prof. Dr. Aydın Alatan for his suggestions and comments. Also I must thank Ali Osman Boyacı, Anas Abidi and Hakan Bayraktar for their support and encouragements. The METU BİLTİR - CAD/CAM & Robotics Center, the staffs of the center and especially the technical assistance Halit Şahin and mechanical engineer Özkan İlkgün are gratefully acknowledged. To my family, I really appreciate their endless support and patience.

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF SYMBOLS

$\mathbf{A}$: Camera Matrix

C : Compactness

D_u : center to center distances between adjacent sensor elements in X direction

D_v : center to center distances between adjacent sensor elements in X direction

E : Euler Number

f : focal length

GT : Geometric Trasformation Matrix

$\mathbf{H}$: Geometric Transformation Matrix

I [i , j]: Intensity Array of Image I

i : row index

j : column index

k_i : Radial Distortion parameter

$\mathbf{m}$: Image coordinate vector

$\tilde{\mathbf{m}}$: Augmented image coordinate vector

$\hat{\mathbf{m}}_i$: Maximum likelihood estimate of $\mathbf{m}$

$\mathbf{M}$: World coordinate vector

$\tilde{\mathbf{M}}$: Augmented world coordinate vector

p_i : Tangential Distortion Parameter

$\mathbf{R}$: Rotation Matrix

RT : Homogenous Transformation Matrix

s_u : Camera uncertainty parameter

$\mathbf{t}$: Translation Vector

u_0 : Image center in column direction

$\tilde{u}_i$: Projected image plane x coordinates

u'_i : Projected image column coordinates

$\breve{u}$: Non-observable distortion-free pixel image coordinates in u direction

$\breve{v}$: Non-observable distortion-free pixel image coordinates in v direction

v_0 : Image center in row direction

$\tilde{v}_i$: Projected image plane y coordinates

v'_i : Projected image row coordinates

x : x coordinate of object point

x' : x coordinate of Image plane coordinates

$\overline{x}_k$: x component of the center of the gravity of the oblect k

X : x coordinate of object coordinates

X_0 : x coordinate of the center of the object coordinates in world coordinates

X_i : x coordinate of the object Point P in image plane coordinates

y : y coordinate of object point

y' : y coordinate of Image plane coordinates

$\overline{y}_k$: y component of the center of the gravity of the oblect k

Y : y coordinate of object coordinates

Y_0 : y coordinate of the center of the object coordinates in world coordinates.

Y_i : y coordinate of the object Point P in image plane coordinates

z : z coordinate of object point

Z : z coordinate of object coordinates

Z_0 : z coordinate of the center of the object coordinates in world coordinates

Z_i : z coordinate of the object Point P in image plane coordinates

θ : Angle between the axis of elongation of the object and x axis

$\Lambda\mathbf{m}_i$: Covariance Matrix

CHAPTER 1

INTRODUCTION

The developments in electronics and computer technologies create a partnership called computer vision. This partnership proposes new developments in a wide range of applications including robotics. The combination of robotics and computer vision deals with the visual guidance of robots.

1.1 Overview

Visual guidance is a rapidly developing approach to the control of robot manipulators, which is based on the visual perception of a robot and a work-piece location. Visual guidance involves the use of one or more cameras and a computer vision system to control the position of the robot's end-effector relative to the work-piece as required by the task.

Modern manufacturing robots can perform assembly and material handling jobs with high speed and good precision. However compared to human workers, robots are at a distinct disadvantage in that they can not see what they are doing.

In industrial applications too much engineering effort is expended in providing a suitable environment for these ‘blind’ machines. This brings the design and manufacture of specialized part feeders, jigs to hold the work in progress, and special purpose end-effectors. The resulting high costs are responsible for robots failing to meet their initial promise of being versatile programmable workers able to rapidly change from one task to the next.

Once the structured work environment has been created, the spatial coordinates of all points of interest must then be taught to the robot, so manual teaching of the robot is required.

The problems of using robot manipulators conventionally may be summarized as:

1. It is necessary to provide, at considerable cost, highly structured work environments for these robots.

2. These robots need time-consuming manual teaching of robot positions.

Designing a visually guided robotic system can overcome these problems. A visually guided robot does not need to know in priori the coordinates of its work-pieces or other objects in its workspace. In manufacturing environment visual guidance would eliminate robot teaching and allow tasks that are not repetitive (such as assembly with out precise fixturing and incoming components without special orientation) [25].

1.2 History

The use of vision with robotic applications has a long history. Brad Nelson, N.P. Papanikolopoulos, and P.K. Khosla made an extensive survey on visually guided robotic assembly [30].

In this survey they concentrated on visual servoing. They claimed that visual feedback could become an integral part of the assembly process by complementing the use of force feedback to accomplish precision assemblies in imprecisely calibrated robotic assembly work-cells, and they represented some of the issues pertaining to the introduction of visual servoing techniques into the assembly process and solutions such as modeling, control and feature tracking issues in static camera servoing, sensor placement, visual tracking model and control, depth-of-field, field-of-view, spatial resolution constraints and singularity avoidance in dynamic sensor placements.

Danica Kragic and Henrik I. Christensen published a great report on visual servoing for manipulation [19]. The report concentrates on different types of visual servoing: image based and position based. Different issues concerning both the hardware and software requirements are considered and the most prominent contributions are reviewed.

Peter I. Corke made another summary about visual control of robot manipulators [26]. In this work he presented a comprehensive summary of research results in the use of visual information to control robot manipulators. Also an extensive bibliography was provided which also includes important papers.

Josef Pauli, Arne Schmidt and Gerald Sommer described an Image – based effector servoing [23]. The proposed approach is a process of perception–action cycles for handling a robot effector under continuous visual feedback. This paper applies visual servoing mechanisms not only for handling objects, but also for camera calibration and object inspection. A 6-DOF manipulator and a stereo camera head are mounted on separate platforms and are derived separately. The system has 3 phases. In calibration phase, camera features are determined.

In the inspection phase, the robot hand carries an object in to the field of view of one camera, then approaches the object along the optical axis to the camera, orients the object for reaching an optimal view, and finally the object shape is inspected in detail. In the assembly phase, the system localizes a board containing holes of different shapes, determines the hole, which fits most appropriate to the object shape, then approaches and arranges the object appropriately. The final object insertion is based on haptic sensors. The robot system can only handle cylindrical and cuboid pegs (see Fig 1.1).

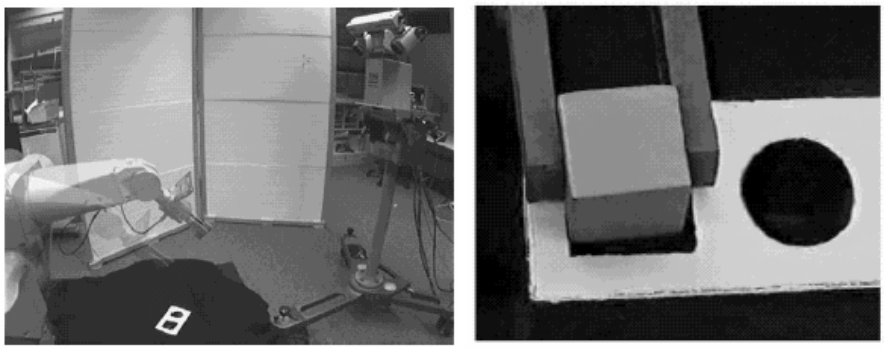


Figure1.1: Image-basedeffector servoing system

Billibon H. Yoshimi and Peter K. Allen discussed the problem of visual control of grasping [24]. They implemented an object tracking system that can be used to provide visual feedback for locating the positions of fingers and objects, as well as the relative relationships between them. They used snakes for object tracking. The center idea behind the snakes is that it is a deformable contour that moves under a variety of image constraints and object model constraints.

They claimed that this visual analysis could be used to control open loop grasping systems where finger contact, object movement, and task completion need to be monitored and controlled.

Roberto Cipolla and Nick Hullinghurst presented simple and robust algorithm which use un-calibrated stereo vision to enable robot manipulator to locate reach and grasp un-modeled objects in unstructured environments [27]. In the first stage of this work the operator indicates the object to be grasped by simply pointing at it. Then the system segments the indicated object from the background and plans a suitable grasp strategy. Finally robot manipulator reaches the object and execute the grasping (see Fig 1.2).

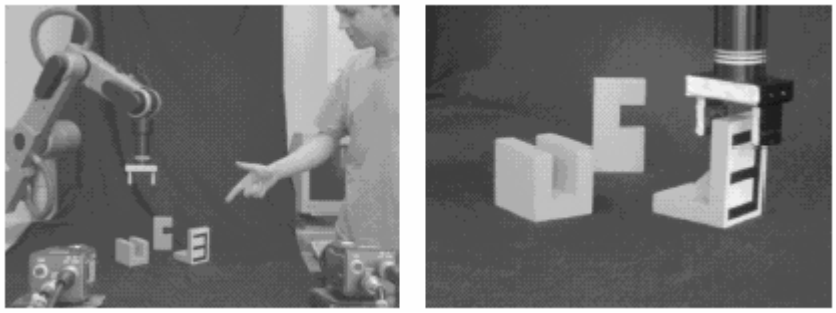


Figure 1.2: The system proposed by Roberto Cipolla and Nick Hullinghurst

Efthimia Kefelea presented a vision system for grasping in [14]. He used a binocular stereo head with pan-tilt mechanism (see Fig 1.3). His approach to object localization was based on detecting the blobs in the scene and treating them as object candidates. After having found the candidates, triangulation yielded an estimation of depth. The next task was to recognize the objects according to their size, shape, exact position orientation.

Recognition was done by comparing the image with the stored model images.

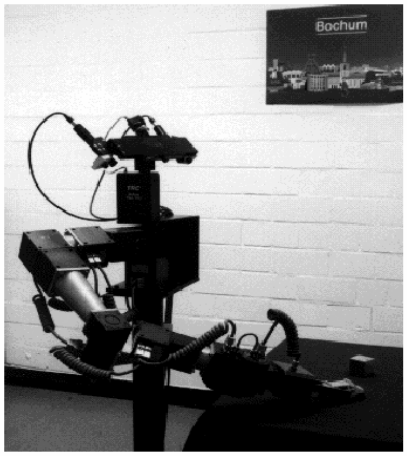


Figure 1.3: Setup of Kefelea

Gordon Wells, Christophe Venaille and Carme Torras presented a new technique based on neural learning and global Image descriptors [16]. In this work a feed-forward neural network is used to learn the complex implicit relationships between the pose displacements of a 6-dof robot and the observed variations in global descriptors of the image, such as geometric moments and Fourier descriptors. The method was shown to be capable of positioning an industrial robot with respect to a variety of complex objects for an industrial inspection application and the results also show that it could be useful for grasping or assembly applications.

Christopher E. Smith and Nikolaos P. Papanikolopoulos proposed a flexible system based upon a camera repositioning controller [21]. The controller allowed the manipulator to robustly grasp objects present in the work space .The system operated in an un-calibrated space with an un-calibrated camera.

In this paper, they discussed the visual measurements they have used in this problem, elaborated on the use of “coarse” and “fine” features for guiding grasping, and discussed feature selection and reselection.

Radu Horaud, Fadi Dornaika, Bernard Espiau presented a visual servoing approach to the problem of object grasping and more generally to the problem of aligning an end-effector with an object [28]. They considered a camera observing a moving gripper. To control the robot motion such that the gripper reaches a previously determined image position, they derived a real-time estimation of the Image Jacobian. Next, they showed how to represent a grasp or an alignment between two solids in 3D space using an un-calibrated stereo rig. For this purpose they used the relation between the specified points on object and gripper. These points might not be present in CAD representation or contact points. They also performed an analysis of the performance of the approach.

U. Büker, S.Drüe, N.Götze, G.Hartmann, B. Kalkreuter, R. Stemmer, R. Trapp presented a vision guided robotic system for an autonomous disassembly process [22]. An active stereo camera system is used as vision sensor. A combination of gray value and contour based object recognition, a position measurement approach with occlusion detection is described, and system is successfully tested on disassembly of wheels.

Rahul Singh, Richard M. Voyles, David Littau, Nikolaos P. Papanikolopoulos presented a new approach to vision-based control of robots [10]. They considered the problem of grasping different objects using a robot with a single camera mounted on the end-effector. In this approach the recognition of the object, identification of its grasp position, pose alignment of the robot with the object and its movement in depth are controlled by image morphing. The image of the all objects at a graspable pose is stored in a data base.

Given an unknown object in the work space, the similarity of the object to the ones in the database are determined by morphing its contours. From the morph a sequence of virtual images is obtained which describes the transformation between the real pose and the goal pose. These images are used as sub-goal images to guide the robot during alignment.

Christopher E. Smith and Nikolaos P. Papanikolopoulos presented a paper concentrated on a vision system for grasping of a target, which describes the several issues with respect to sensing, control, and system configuration [29]. This paper presents the options available to the researcher and the trade-offs to be expected when integrating a vision system with a robotic system for the purpose of grasping objects. The work includes experimental results from a particular configuration that characterize the type and frequency of errors encountered while performing various vision-guided grasping tasks.

1.3 Thesis Objectives

The objective of this thesis is to develop a system for visually guided robotic assembly. In the thesis ABB IRB-2000 Industrial robot, Sony Evi-D31 pan/tilt network camera, Axis Evi-D31 pan/tilt network camera, Data Translation DT 3133 frame grabber card are used. The setup is an Eye-In-Hand system with a stereo rig. The system to be developed will have the following specifications:

- To control the industrial robot on-line through a PC
- To control the cameras off– line and on – line through a PC
- To design a Graphical User Interface (GUI) to allow the user to freely control the robot, to observe the assembly process and to interfere if needed

- To design a system that can recognize and distinguish basic geometric primitives, which are specified as cylinders and rectangular prisms with different dimensions.
- The system should also be able to grasp randomly located objects on a table and assemble them to their respective templates which are also placed randomly on the table.

1.4 Limitations

Based on the scope of this thesis the following limitations apply:

- The objects to be grasped and the templates are static.
- Closed-loop control, image based servoing are not considered.
- It is assumed that the assembly parts are not occluded. Occlusion analysis / detection in the scene is also not considered.
- Graspability of the objects is not considered.

1.5 Thesis Outline

The logical structure of the system is represented as modules. In succeeding chapters these modules will be discussed. The system has four modules. A quick outline of these chapters is presented below.

In Chapter 2 - The Calibration module, the geometry of imaging, the necessity of calibration, camera model and the method of calibration used in the thesis are to be discussed. In Chapter 3 – The Imaging and Preprocessing Module, sampling and quantization will be discussed. Also the initialization of the images including color model conversion, distortion correction and image segmentation are to be discussed.

In Chapter 4 - Object Recognition Module, the object extraction (including the object and region labeling), geometric properties of an object, recognition strategies and the recognition strategy followed in this project will be discussed. In Chapter 5 - Depth Extraction and Assembly Module, the stereo imaging principles, stereo matching strategies and the matching strategy followed in this study, the depth extraction and the followed assembly strategy will be also discussed. In Chapter 6 – Error Analysis, the sources of the errors in the system will be discussed. In Chapter 7 - Discussion and Conclusion, the study is concluded by summarizing the work done and discussing future directions.

CHAPTER 2

CALIBRATION MODULE

In this chapter the geometry of imaging, the necessity of calibration, camera model and the method of calibration used in the thesis are to be discussed.

2.1 Image Geometry

The basic model for the projection of points in the scene onto the image plane is diagrammed in Fig 2.1. In this model, the imaging system's center of projection coincides with the origin of the 3-D coordinate system.

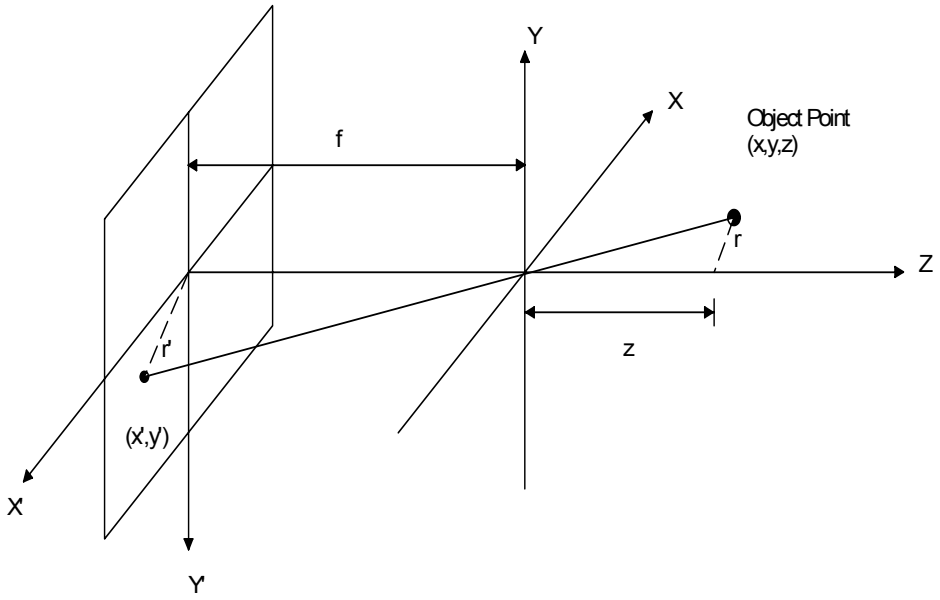


Figure 2.1: Image Geometry

A point in the scene has the coordinates (x, y, z). The ‘x’ coordinate is the horizontal position of the point in space as seen from the camera, the ‘y’ coordinate is the vertical position of the point in space as seen from the camera and the ‘z’ coordinate is the distance from the camera to the point in space along a line parallel to the Z-axis .The line of sight of a point in the scene, is the line that passes through the point of interest and the center of the projection.

The image plane is parallel to the X and Y axes of the coordinate system at a distance ‘f’ from the center of projection. The image plane is spanned by the vectors x’ and y’ to form a 2D coordinate system for specifying the position of the points in the image plane .The position of a point in image plane is specified by the two coordinates *x’* and *y’*. The point (0, 0) in the image plane is the origin of the image plane.

The point (*x', y'*) in the image plane, which is the projection of a point at position (*x, y, z*) in the scene, is found by computing the coordinates (*x', y'*) of the intersection of the line of sight passing through the scene point (*x, y, z*). The distance of the point (x, y, z) from the Z axis is $r = \sqrt{x^2 + y^2}$ and the distance of the projected point (*x', y'*) from the origin of the image plane is $r' = \sqrt{x'^2 + y'^2}$. The Z axis, the line of sight to point (x, y, z), and the line segment of length r from point (x, y, z) to the Z axis form a triangle. The Z axis, the line of sight to point (*x', y'*) in the image plane, and the line segment r' from point (*x', y'*) to the Z axis form another triangle, and these triangles are similar so the ratios of the corresponding sides of the triangles must be equal.

$$\frac{f}{z} = \frac{r'}{r} \tag{2.1}$$

$$\frac{x'}{x} = \frac{y'}{y} = \frac{r'}{r} \tag{2.2}$$

Combining equations (2.1) and (2.2) yields the equations for the perspective projection:

$$\frac{x'}{x} = \frac{f}{z} \tag{2.3}$$

$$\frac{y'}{y} = \frac{f}{z} \tag{2.4}$$

Then the position of a point (x, y, z) in the image plane can be written as:

$$x' = f\frac{x}{z} \tag{2.5}$$

$$y' = f\frac{y}{z} \tag{2.6}$$

2.2 Camera Model

Camera parameters are commonly divided into extrinsic and intrinsic parameters. Extrinsic parameters are needed to transform object coordinates to a camera centered coordinate frame. In multi-camera systems, the extrinsic parameters also describe the relationship between the cameras [1].

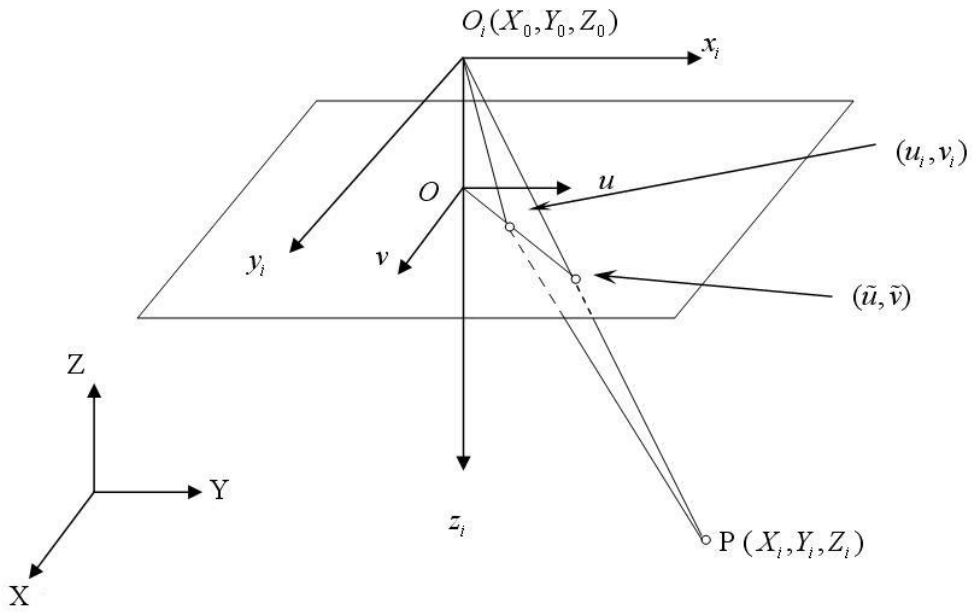


Figure 2.2: Image plane coordinates – World Coordinates Geometry

The origin of the camera coordinate system is in the projection center at the location (X_0, Y_0, Z_0) with respect to the object coordinate system, and the z-axis of the camera frame is perpendicular to the image plane (see Fig 2.2).

In order to express an arbitrary object point *P* at location (X_i, Y_i, Z_i) in image coordinates, we first need to transform it to camera coordinates (x_i, y_i, z_i).

This transformation consists of a translation and a rotation, and it can be performed by using the following matrix equation:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = R * \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + T \tag{2.2.1}$$

Where R is the rotation matrix

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{2.2.2}$$

And T is the translation vector

$$T = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} \tag{2.2.3}$$

The intrinsic camera parameters usually include the effective focal length *f*, scale factor "s_U", and the image center (u_0, v_0) also called the principal point. "s_U" is an uncertainty parameter which is due to a variety of factors such as slight hardware timing mismatch between image acquisition hardware and camera scanning hardware.

Here, usually in computer vision literature, the origin of the image coordinate system is in the upper left corner of the image array. The unit of the image coordinates is pixels, and therefore coefficients D_U and D_V are needed to change the metric units to pixels. D_U and D_V are center-to-center distances between adjacent sensor elements in horizontal (*u*) direction and vertical (*v*) direction respectively.

These coefficients can be typically obtained from the data sheets of the camera and frame grabber. By using the pinhole model, the projection of the point (x_i, y_i, z_i) to the image plane is expressed as:

$$\begin{bmatrix} \tilde{u}_i \\ \tilde{v}_i \end{bmatrix} = \frac{f}{z_i} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{2.2.4}$$

The corresponding image coordinates ($u_i^{'}$, $v_i^{'}$) in pixels are obtained from the projection by applying the following transformation:

$$\begin{bmatrix} u_i' \\ v_i' \end{bmatrix} = \begin{bmatrix} Du * su * \tilde{u}_i \\ Dv * \tilde{v}_i \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{2.2.5}$$

The pinhole model is only an approximation of the real camera projection. It is a useful model that enables simple mathematical formulation for the relationship between object and image coordinates. However, it is not valid when high accuracy is required and therefore, a more comprehensive camera model must be used. Usually, the pinhole model is a basis that is extended with some corrections for the systematically distorted image coordinates. The most commonly used correction is for the radial lens distortion that causes the actual image point to be displaced radially in the image plane.

The radial distortion can be approximated using the following expression:

$$\begin{bmatrix} \delta u^{(r)}{}_i \\ \delta v^{(r)}{}_i \end{bmatrix} = \begin{bmatrix} \tilde{u}_i (k_1 r_i^2 + k_2 r_i^4 + \ldots) \\ \tilde{v}_i (k_1 r_i^2 + k_2 r_i^4 + \ldots) \end{bmatrix} \tag{2.2.6}$$

Where $k_1, k_2, \ldots$ are coefficients of radial distortion and $r_i = \sqrt{\tilde{u}_i^2 + \tilde{v}_i^2}$.

Centers of curvature of lens surfaces are not always strictly collinear. This introduces another common distortion type, de-centering distortion which has both a radial and tangential component.

The expression for the tangential distortion is often written in the following form:

$$\begin{bmatrix} \delta u_i^{(t)} \\ \delta v_i^{(t)} \end{bmatrix} = \begin{bmatrix} 2p_1\tilde{u}_i\tilde{v}_i + p_2(r_i^2 + 2u_i^2) \\ p_1(r_i^2 + 2\tilde{v}_i^2) + 2p_2\tilde{u}_i\tilde{v}_i \end{bmatrix} \quad (2.2.7)$$

Where p_1 and p_2 are coefficients of tangential distortion.

Combining the pinhole model with the correction for the radial and tangential distortion components can derive a proper camera model for accurate calibration:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} D_u s_u(\tilde{u}_i + \delta u_i^{(r)} + \delta u_i^{(t)}) \\ D_v(\tilde{v}_i + \delta v_i^{(r)} + \delta v_i^{(t)}) \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \quad (2.2.8)$$

2.3 Camera Calibration

Camera calibration is the process of determining the internal camera geometric and optical characteristics (intrinsic parameters) and the 3-D position and orientation of the camera frame relative to a certain world coordinate system (extrinsic parameters). It is useful to determine the camera calibration matrix which relates a world coordinate system to the image plane coordinates for a given camera position and orientation. In many cases, the overall performance of the machine vision system strongly depends on the accuracy of the camera calibration. [1],[6],[9],[17],[20]

Camera calibration is the first step towards computational computer vision. Calibration is essential when metric information is required. Some applications of this capability include:

1. Dense reconstruction:

   Each image point determines an optical ray passing through the focal point of the camera toward the scene. Use of more than a view of a motionless scene permits to cross both optical rays and get the metric position of the 3D point.

2. Visual inspection:

   Once a dense reconstruction of a measured object is obtained, it can be compared with a stored model in order to detect some manufacturing imperfections as bumps, dents and cracks.

3. Object localization:

   From points of different objects, the position relation among these objects can be easily determined, which has many application in industrial part assembly and obstacle avoidance in robot navigation.

Camera modeling and calibrating is a crucial problem for metric measuring of a scene. A lot of different calibrating techniques and surveys about calibration have been presented in last years. These approaches can be classified with respect to the calibrating method used to estimate the parameters of the camera model as follows:

1. Non-linear optimization techniques.

   The camera parameters are obtained through iteration with the constraint of minimizing a determined function. The advantage of these techniques is that it can calibrate nearly any model and the accuracy usually increases by increasing the number of iterations.

2. Linear techniques, which compute the transformation matrix

These techniques use the least squares method to obtain a transformation matrix, which relates 3D points with their projections. However, these techniques cannot model lens distortion. Moreover, it's sometimes difficult to extract the parameters from the matrix due to the implicit calibration used.

3. Two-step techniques

These techniques use a linear optimization to compute some of the parameters and, as a second step, the rest of parameters are iteratively computed. These techniques permit a rapid calibration, which considerably reduces the number of iterations. Moreover, the convergence is nearly guaranteed due to the linear guess obtained in the first step.

The calibration method used in this thesis is a slightly modified version of Zhengyou Zhang's calibration method, which is proposed in [9]. In Zhang's method, only radial distortions are modeled but in the method used, tangential distortions are also modeled. The model is exactly the same model with the model that is described in previous section (Heikkila and Silven's model) [1].

Compared to classical techniques, which use expensive equipment such as two or three orthogonal planes and have many restrictions, this technique is easy to use and flexible. It only requires the camera to observe a planar pattern shown at few (at least two) different orientations. Either the camera or the planar pattern can be freely moved.

The motion needs not to be known. Computational cost due to processing at least 3 (generally for a good solution 15-20) images can be the draw back of this method but keeping in mind that the calibration is done for only once, this drawback can be ignored.

2.3.1 Theory

Let's denote a 2D point (image point) as $\mathbf{m} = [u, \quad v]^T$ and a 3D point as $\mathbf{M} = [X, \quad Y, \quad Z]^T$. The augmented vectors by adding 1 as the last element can be defined as $\tilde{\mathbf{m}} = [u, \quad v, \quad 1]^T$ and $\tilde{\mathbf{M}} = [X, \quad Y, \quad Z, \quad 1]^T$.

Then the relationship between a 3D point **M** and its image projection m is given by:

$$\tilde{\mathbf{m}} = \mathbf{A}[\mathbf{R} \quad \mathbf{t}]\tilde{\mathbf{M}} \tag{2.3.1}$$

Where **R** is the rotation matrix, **t** is the translation vector and **A** is the camera intrinsic matrix given by:

$$\mathbf{A} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3.2}$$

With u_0 and v_0 the coordinates of principal point, $\alpha = f * Du * su$ and $\beta = f * Dv$ the scale factor in image *u* and *v* axes and γ the skew coefficient defining the angle between the *u* and *v* axes.

It is assumed that the model plane is on Z = 0 of the world coordinate system. If the i^{th} column of the rotation matrix R is denoted by r_i, then the equation (2.3.1) can be written as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix}\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

$$= \mathbf{A}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \qquad (2.3.3)$$

Since Z = 0 $\tilde{\mathbf{M}} = \begin{bmatrix} X, & Y, & 1 \end{bmatrix}^T$. Therefore, a 3D model point **M** and its image point **m** is related by a geometrical transform **H**:

$$\tilde{\mathbf{m}} = \mathbf{H}\tilde{\mathbf{M}} \qquad (2.3.4)$$

With $\mathbf{H} = \mathbf{A}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}$

As it can be seen, **H** is a 3x3 matrix and it can be estimated by a technique based on maximum likelihood criterion.

Let $\mathbf{M}_i$ and $\mathbf{m}_i$ be the model and image points respectively. Although they should satisfy $\tilde{\mathbf{m}} = \mathbf{H}\tilde{\mathbf{M}}$, in practice because of the noise in the extracted image points, they don't. It is assumed that $\mathbf{m}_i$ is corrupted by Gaussian noise with mean 0 and covariance matrix $\Lambda\mathbf{m}_i$. The maximum likelihood estimation of **H** can be obtained by minimizing the functional:

$$\sum_i (m_i - \hat{m}_i)^T \Lambda_{m_i}^{-1} (m_i - \hat{m}_i) \qquad (2.3.5)$$

If $\mathbf{H}=\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$ and $\bar{\mathbf{h}}_i = \begin{bmatrix} h_{i1} \\ h_{i2} \\ h_{i3} \end{bmatrix}$ subjected to $\tilde{\mathbf{m}} = \mathbf{H}\tilde{\mathbf{M}}$, it can be seen that $\mathbf{m}_i$ will be equal to $\frac{1}{\bar{\mathbf{h}}_3^T \mathbf{M}_i}\begin{bmatrix} \bar{\mathbf{h}}_1^T \mathbf{M}_i \\ \bar{\mathbf{h}}_2^T \mathbf{M}_i \end{bmatrix}$.

Then it can be written as $\hat{\mathbf{m}}_i = \frac{1}{\bar{\mathbf{h}}_3^T \mathbf{M}_i}\begin{bmatrix} \bar{\mathbf{h}}_1^T \mathbf{M}_i \\ \bar{\mathbf{h}}_2^T \mathbf{M}_i \end{bmatrix}$. Also it can be assumed that $\Lambda m_i = \sigma^2 \mathbf{I}$ since all image coordinates are extracted independently with the same procedure. With this assumption the above problem becomes a nonlinear least squares one:

$$\min_{\mathbf{H}} \sum_i \left\| \mathbf{m}_i - \hat{\mathbf{m}}_i \right\|^2 \tag{2.3.6}$$

The problem can be solved by using Levenberg-Marquardt Algorithm. This requires an initial guess.

To obtain initial guess:

Let $\mathbf{x} = \left[\bar{\mathbf{h}}_1^T, \ \bar{\mathbf{h}}_2^T, \ \bar{\mathbf{h}}_3^T \right]^T$. Then $\tilde{\mathbf{m}} = \mathbf{H}\tilde{\mathbf{M}}$ can be written as:

$$\begin{bmatrix} \tilde{\mathbf{M}}^T & \mathbf{0}^T & -u\tilde{\mathbf{M}}^T \\ \mathbf{0}^T & \tilde{\mathbf{M}}^T & -v\tilde{\mathbf{M}}^T \end{bmatrix} \mathbf{x} = \mathbf{0} \tag{2.3.7}$$

For n points, there will be n above equations and can be written in matrix equation:

$$\mathbf{L}\mathbf{x} = \mathbf{0} \tag{2.3.8}$$

Where **L** is a 2nx9 matrix. In this equation **x** will be the right singular vector of **L** associated with the smallest singular value.

After having estimated the homography **H,** it can be written as:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} \tag{2.3.9}$$

and from (2.3.4):

$$\begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \lambda \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \tag{2.3.10}$$

where λ is an arbitrary scalar.

Using the knowledge that $\mathbf{r}_1$ and $\mathbf{r}_2$ are orthonormal, two constraints on intrinsic parameters can be extracted:

$$\mathbf{h}_1^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_2 = 0 \tag{2.3.11}$$

$$\mathbf{h}_1^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_1 = \mathbf{h}_2^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_2 \tag{2.3.12}$$

Then let

$$\begin{aligned} \mathbf{B} = \mathbf{A}^{-T}\mathbf{A}^{-1} &\equiv \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} \\ \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} & \frac{(v_0\gamma - u_0\beta)^2}{\alpha^2\beta^2} + \frac{v_0^2}{\beta^2} + 1 \end{bmatrix} \end{aligned} \tag{2.3.13}$$

Note that **B** is symmetric, defined by a 6D vector:

$$\mathbf{b} = \begin{bmatrix} \mathbf{B}_{11}, & \mathbf{B}_{12}, & \mathbf{B}_{22}, & \mathbf{B}_{13}, & \mathbf{B}_{23}, & \mathbf{B}_{33} \end{bmatrix}^T \tag{2.3.14}$$

Let the i^{th} column vector of **H** be $\mathbf{h}_i = \begin{bmatrix} h_{i1} & h_{i2} & h_{i3} \end{bmatrix}^T$, then:

$$\mathbf{h}_i^T \mathbf{B} \mathbf{h}_j = \mathbf{v}_{ij}^T \mathbf{b} \tag{2.3.15}$$

Where

$$v_{ij} = \begin{bmatrix} h_{i1}h_{j1}, & h_{i1}h_{j2} + h_{i2}h_{j1}, & h_{i2}h_{j2}, & h_{i3}h_{j1} + h_{i1}h_{j3}, & h_{i3}h_{j2} + h_{i2}h_{j3}, & h_{i3}h_{j3} \end{bmatrix}^T \tag{2.3.16}$$

Therefore, the constraints (2.3.11) and (2.3.12) can be written as:

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix} \mathbf{b} = 0 \tag{2.3.17}$$

For n images of the model plane, it becomes:

$$\mathbf{V}\mathbf{b} = \mathbf{0} \tag{2.3.18}$$

Where V is a 2n x 6 matrix. For $n \geq 3$ there will be a general unique solution **b** .The solution is well known as the right singular vector of V associated with the smallest singular value.

Once **b** is computed, all parameters of intrinsic camera matrix **A** can be computed

$$\begin{aligned} v_0 &= (B_{12}B_{13} - B_{11}B_{23}) / (B_{11}B_{22} - B_{12}^2) \\ \lambda &= B_{33} - \left[B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23}) \right] / B_{11} \\ \alpha &= \sqrt{\lambda / B_{11}} \\ \beta &= \sqrt{\lambda B_{11} / (B_{11}B_{22} - B_{12}^2)} \\ \gamma &= -B_{12}\alpha^2\beta / \lambda \\ u_0 &= \gamma v_0 / \alpha - B_{13}\alpha^2 / \lambda \end{aligned} \tag{2.3.19}$$

After having found matrix A, the extrinsic parameters for each image are readily computed. From (2.3.4):

$$\begin{aligned}\mathbf{r}_1 &= \lambda \mathbf{A}^{-1}\mathbf{h}_1 \\ \mathbf{r}_2 &= \lambda \mathbf{A}^{-1}\mathbf{h}_2 \\ \mathbf{r}_3 &= r_1 \times r_2 \\ \mathbf{t} &= \lambda \mathbf{A}^{-1}\mathbf{h}_3\end{aligned} \tag{2.3.20}$$

This solution can be refined by maximum likelihood estimation. For n images and m points in the model plane, the maximum likelihood estimate can be obtained by minimizing the following functional:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|\mathbf{m}_{ij} - \hat{\mathbf{m}}(\mathbf{A}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{M}_j)\right\|^2 \tag{2.3.21}$$

This minimization problem can be solved with Levenberg-Marquardt Algorithm. The results of the technique described above can be used as initial guess.

Up to now lens distortions are not included in the solution. To get a proper solution lens distortions have to be included. Let (u, v) be the ideal (non-observable distortion-free) pixel image coordinates and $(\breve{u}, \breve{v})$ be the corresponding observed pixel image coordinates.

The ideal points are the projections of the model points according to pinhole model. If (x, y) and $(\breve{x}, \breve{y})$ are the ideal (distortion-free) and real (distorted) normalized image coordinates. Then:

$$\begin{aligned}\breve{x} &= x + x\left[k_1 r^2 + k_2 r^4\right] + 2p_1 xy + p_2(r^2 + 2x^2) \\ \breve{y} &= y + y\left[k_1 r^2 + k_2 r^4\right] + p_1(r^2 + 2y^2) + 2p_2 xy\end{aligned} \tag{2.3.22}$$

Where $r^2 = x^2 + y^2$ and k_1, k_2, p_1 and p_2 are radial and tangential distortion parameters respectively.

From $\breve{u} = u_0 + \alpha\breve{x} + \gamma\breve{y}$ and $\breve{v} = v_0 + \beta\breve{y}$, assuming the skew is 0:

$$\begin{aligned} \breve{u} &= u + (u - u_0)[k_1 r^2 + k_2 r^4 + p_1 y + p_2(\frac{r^2}{x} + 2x)] \\ \breve{v} &= v + (v - v_0)[k_1 r^2 + k_2 r^4 + p_1(\frac{r^2}{y} + 2y) + p_2(2x)] \end{aligned} \tag{2.3.23}$$

The intrinsic parameters are estimated reasonable well by simply ignoring the distortion parameters. One strategy is to estimate distortion parameters after having estimated other parameters. To estimate the distortion parameters for each point in each image there are two equations and they can be written as:

$$\begin{bmatrix} (u-u_0)r^2 & (u-u_0)r^4 & (u-u_0)y & (u-u_0)(\frac{r^2}{x} + 2x) \\ (v-v_0)r^2 & (v-v_0)r^4 & (v-v_0)(\frac{r^2}{y} + 2y) & (v-v_0)2x \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} \breve{u} - u \\ \breve{v} - v \end{bmatrix} \tag{2.3.24}$$

For given m points in n images, totally there are 2mn equations, so distortion parameters can be estimated. The solution is well known as the right singular vector associated with the smallest singular value.

After having estimated distortion parameters, the solution can be refined by Maximum Likelihood Estimation including all the parameters. The estimation is a minimization of the following functional:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \left\| \mathbf{m}_{ij} - \breve{\mathbf{m}}(\mathbf{A}, k_1, k_2, p_1, p_2, \mathbf{R}_i, \mathbf{t}_i, \mathbf{M}_j) \right\|^2 \tag{2.3.25}$$

This is a non-linear minimization problem and can be solved with the Levenberg-Marquardt Algorithm.

2.3.2 Implementation

Intel Open Computer Vision Library (Open CV) is used for calibration of the cameras. The library has its own functions for the method described in the previous section.

There is not a ready program for calibrating cameras, so one must write his/her own code using OpenCV calibration functions. The calibration code in this thesis has three steps.

Step 1:

Code automatically reads the calibration images, and extracts the corners of the calibration pattern. The OpenCV library uses chessboard for calibration pattern (see Fig. 2-3). To achieve more accurate calibration results, the pattern should be printed with high resolution on high-quality paper and this paper should be put on a hard (preferably glass) substrate.

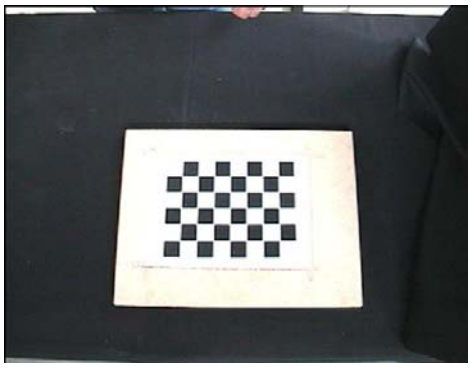


Fig 2.3: Calibration Pattern

The corner extraction function in OpenCV Library "cvFindChessBoardCornerGuesses" is a specialized function for extracting the corners of the chessboard pattern. After having found the image coordinates of corners, the code refines these corners in sub-pixel accuracy with the function "cvFindCornerSubPix".

Step 2:

The code arranges all the corners in each image and assigns their 3D coordinates. These 3D coordinates are independent from any other coordinate system. The origin of the coordinate system is the first corner. For each image assuming z = 0, x and y coordinates are incremented according to the chessboard parameters (see Fig 2.4).

Default parameters are:

# of columns: 8

# of rows : 6

increment in x direction dx = 30 mm

increment in y direction dy = 30 mm

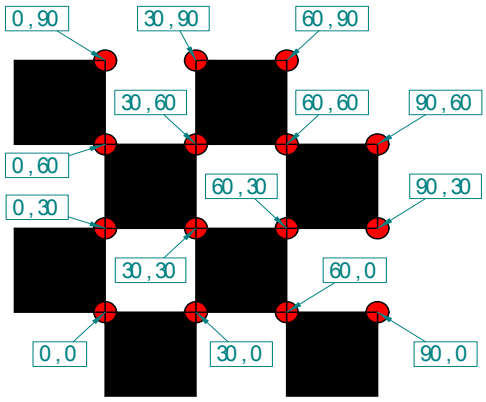


Figure 2.4: 4x4 of 6x8 Chessboard

Step 3:

The extracted image point (2D) and the model point (3D) coordinates for each image are fed to the calibration function of OpenCV ”cvCalibrateCamera_64d”. The function outputs the camera matrix, camera distortion parameters and extrinsic parameters (rotation and translation vectors for each image).

The accuracy of the calibration depends on the number of images, number of the points in the calibration pattern and size of the calibration pattern. Larger calibration board with larger number of model points will improve the calibration.

In this work an A4 size 35 point-chessboard pattern is used. The number of calibration images is 20. Experiment results in [9] show that the most accurate calibrations are obtained with 15-20 images.

During capturing calibration images, the lightning conditions are extremely important. The pattern should not be too illuminated due to the shining effect on the pattern. This effect will increase the error during corner extraction, even might cause the system to miss some of the corners. This will result in an ill-conditioned calibration.

CHAPTER 3

IMAGING AND PREPROCESSING MODULE

In this chapter imaging and preprocessing module will be discussed. This module involves image capturing, converting images to digital format (so they can be analyzed in computer environment), distortion correction and preprocessing operations.

Image Processing can be considered as a sub-field of machine vision. Image processing techniques usually transform images in to other images; the task of information recovery is left to human. This field includes topics such as image enhancement, image compression, image correction, image segmentation. [12]

3.1 Sampling and Quantization

A television signal is an analog waveform whose amplitude represents the spatial image intensity $\boldsymbol{I}$ (x, y). This continuous Image function cannot be represented exactly in a digital computer. There must be a digitizer between the optical system and the computer to sample the image at a finite number of points and represent each sample within the finite word size of the computer. This is called sampling and quantization. The samples are referred to as picture elements or pixels.

Many cameras acquire an analog image, which is sampled and quantized to convert it to a digital image The sampling rate determines how many pixels the digital image have (the image resolution), and quantization determines how many intensity levels will be used to represent the intensity value at each sample point. [12]

In general each pixel is represented in the computer as a small integer. Frequently Monochrome (gray-scale) are used for image processing and the pixel is represented as an unsigned 8-bit integer in the range of [0,255] with 0 corresponding to black, 255 corresponding to white, and shades of gray distributed over the middle values.

There are also other image models with different color channels and different pixel sizes such as absolute color 24-bit RGB (Red-Green-Blue) image. In RGB images three bytes (24 bits) per pixel represent the three channel intensities. Or MSI (Multi Spectral Image). An MSI image can contain any number of color channels; they may even correspond to invisible parts of the spectrum.

3.2 Image Architecture

The relationship between the geometry of the image formation and the representation for images in the computer is very important. There must be a relation ship between the mathematical notation used to develop image processing algorithms and the algorithmic notation used in programs.

A pixel is a sample of the image intensity quantized to an integer value. Then it is obvious that an image is a two-dimensional array of pixels. The row and column indices [i, j] of a pixel are integer values that specify the row and column in the array of pixel values.

Pixel [0, 0] is located at the top left corner of the image. The index i (row) increases from top to bottom of the image, and the index j (column) increases from left to right of the image (see Fig 3.1).

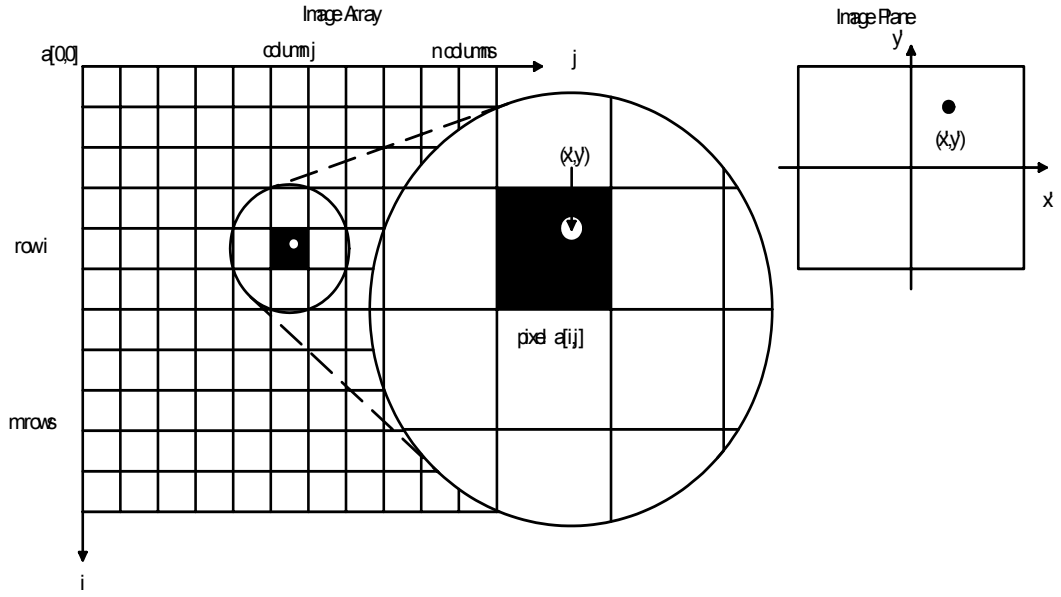


Figure 3.1: Sketch of an image array

In the image plane points have x and y coordinates. And this coordinate system is slightly different from array coordinate system. The origin (0, 0) is at the center of the image plane and y increases from center to top, x increases form center to right.

It can be seen that the axes of image array are the reverse of the axes in the image plane. Assuming that the image array has m rows and n columns then the relationship between the image plane's coordinates and image array coordinates can be written as:

$$x = j - \frac{m-1}{2} \tag{3.1}$$

$$y = -(i - \frac{n-1}{2}) \tag{3.2}$$

3.3 Preprocessing

The aim of the preprocessing is to make the captured images ready for analysis. The initialization process starts with image reading from files. The captured images are saved to computer hard disk as 24-bit bitmap files by the digitizer. To process these images, fist the system should convert these images from file format to 2D array format that can be reached and manipulated.

For this purpose the code uses OpenCV (Intel Open Computer Vision) Library's "`cvLoadImage`" function. This function uses an image file as input and outputs a 2D array in "`IplImage`" format. This format is a variable type for coding which is developed in Intel Image Processing Library. When "`cvLoadImage`" function is called by the code, the function will return a 24-bit RGB, top left centered 2D array IplImage. At this point the image is introduced to computer completely and properly.

Then initialization process continues with two steps. These steps can be written as:

1. Image Initialization
2. Image Segmentation

3.3.1 Image Initialization

This step involves two sub-steps. These sub-steps can be written as:

1. Color format conversion
2. Distortion correction

Step 1 - Color Format Conversion:

As mentioned before the captured image is read in a 24-bit RGB format, so the image has three-color plane. This means that each pixel in the image has three values corresponding to red, green and blue in the range of [0,255] for each.

Color processing is another field of image processing and machine vision, which is not needed for the designed system. The image should be converted to a Monochrome (gray-scale) image for processing. This conversion can be done by applying a simple formula on the pixel values of red, green and blue.

Let R, *G* and *B* be the intensity values of the Red, Green and Blue planes respectively, and GR the gray-scale intensity values. Then the formula can be written as:

$$GR = 0.212675 * R + 0.715160 * G + 0.072169 * B \quad (3.3)$$

With this formula a three-channel (colored) image can be converted to a gray-scale image by applying it to all pixels in the image.

For this purpose the code uses a function, which needs a colored image as an input and returns another image in gray-scale format by applying equation (3.3) to all pixels of input image. In Fig 3.3.1 and Fig 3.3.2 there are images as an example for conversion. The image in Fig 3.3.1 is fed to the function and the image in Fig 3.3.2 is got as an output.

Figure 3.3.1: A 24-bit RGB Image

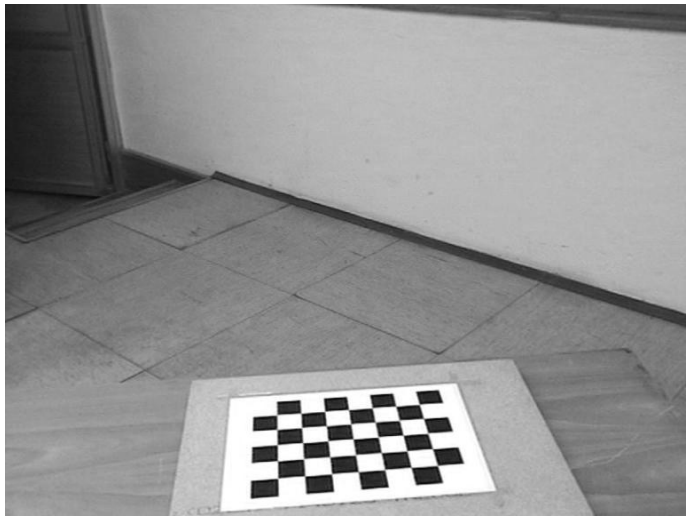


Figure 3.3.2: 8-bit Gray-Scale Image

Step 2 - Distortion Correction:

As discussed in Chapter 2, lenses have radial and tangential distortions. These distortions must be corrected before using some algorithms that extracts pixel coordinates.

For this purpose the code uses a ready function of OpenCV Library called "`cvUnDistort`". This function uses the image to be corrected, the distortion parameters and the camera matrix as input and gives a corrected image as output. See Fig (3.3.3) and Fig (3.3.4). In the first figure note that at the corners of the image, the floor can be seen. In the second image (corrected) the floor cannot be seen.

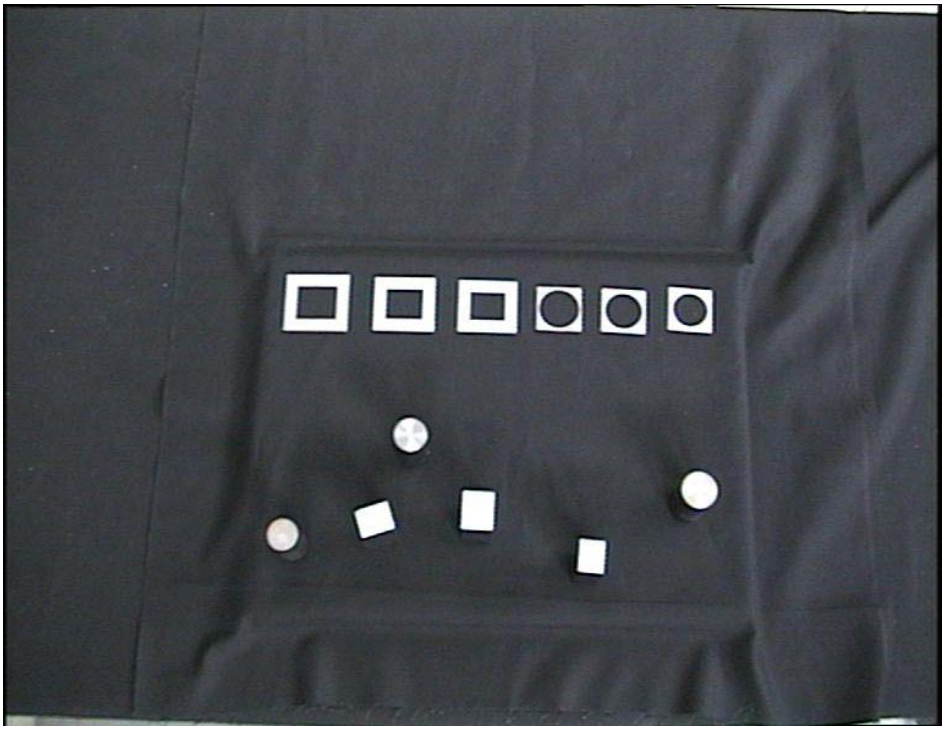


Figure 3.3.3: Distorted image

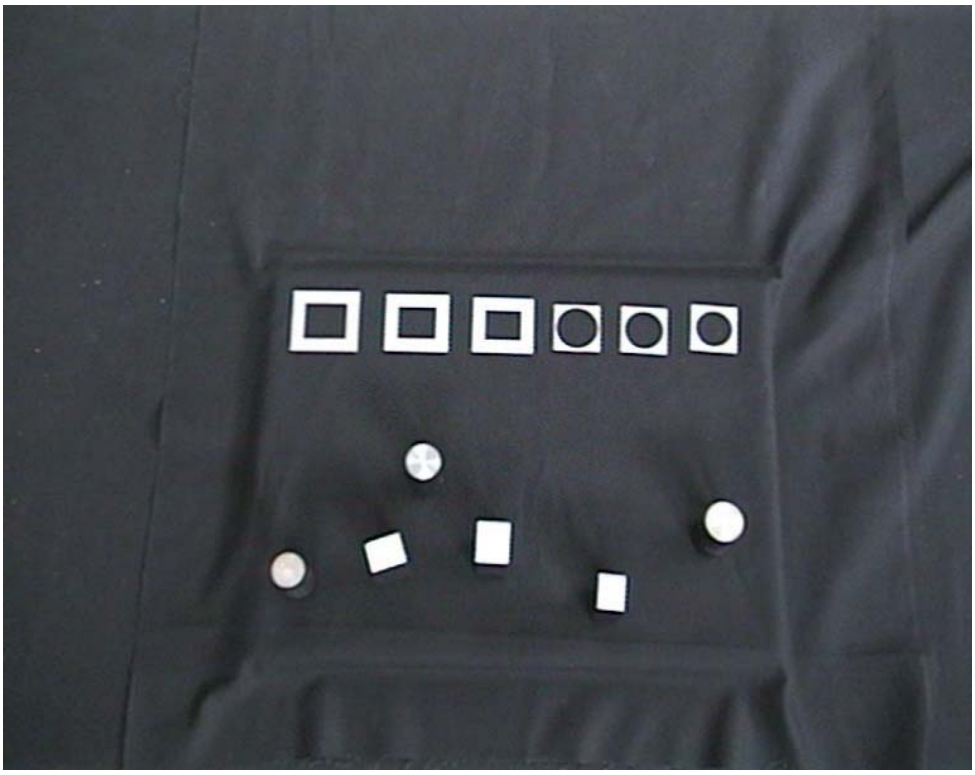


Figure 3.3.4: Corrected Image

3.3.2 Image Segmentation

One of the most important problems in a computer vision system is to identify the sub images that represent objects. This operation, which is so natural and so easy for people, is surprisingly difficult for computers. The partitioning of an image in to regions is called segmentation. Segmentation can be defined as a method to partition an image, Im [i, j], in to sub images called regions such that each sub image is an object candidate. Segmentation is a very important step in understanding images.

A binary image is obtained using an appropriate segmentation of a gray-scale image.

If intensity values (pixel values) of an object in the scene are in a interval and the intensity values (pixel values) of the background of the scene is out of the interval, using this interval value, binary image or segmented regions can be obtained. This method is called “Tresholding “.

Tresholding method converts a gray-scale image in to a binary image to separate the object of interest from the background. To make an effective tresholding, the contrast between the object and background must be sufficient.

Let G [i, j] is a gray scale image to be segmented and B [i, j] is the binary image to be obtained. Then tresholding method can be defined mathematically as:

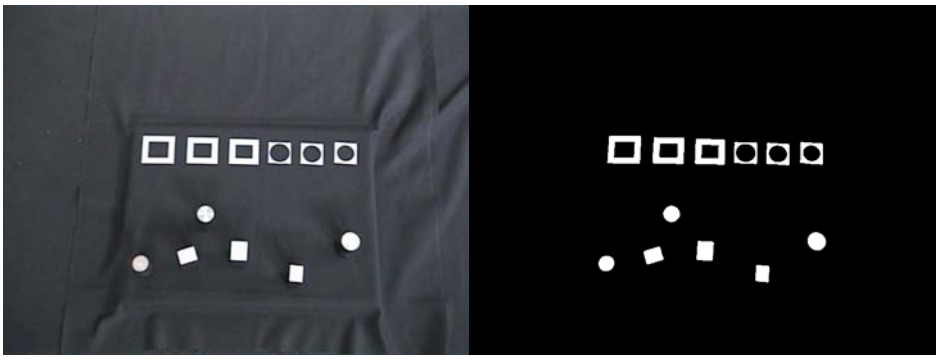
$$B[i,j] = \begin{cases} 1 \text{ if } \mathrm{G}[i,j] \le T \\ 0 \quad \text{otherwise} \end{cases}$$

If it is known that the object intensity value is between T1 and T2 then:

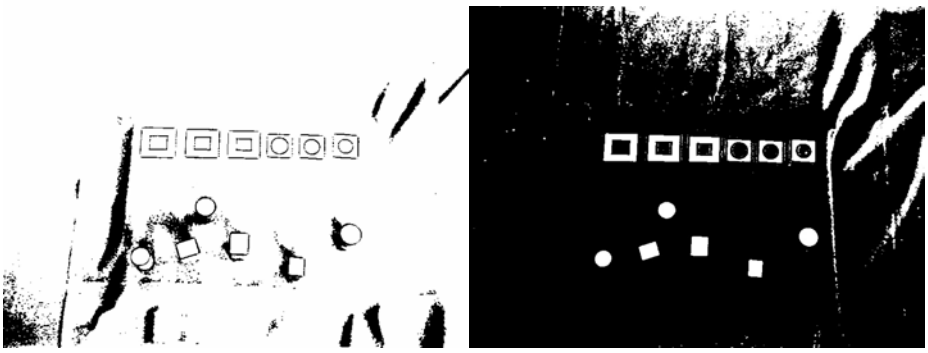
$$B[i,j] = \begin{cases} 1 \text{ if } T_1 \le G[i,j] \le T_2 \\ 0 \quad \text{otherwise} \end{cases}$$

The tresholding function used in the code is fed with the gray-scale image to be tresholded and the threshold value. The function gives a second image, which corresponds to binary image of the input image, without altering the input image.

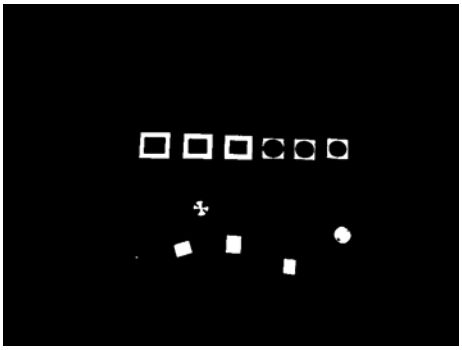
The “1” which represents the objects is exchanged with 255 and the “0” which represents the background remains same. As discussed before “255” corresponds to white and “0” corresponds to black. It is obvious that after tresholding operation, the resultant image will be a gray-scale image, which has only two gray intensities; black for background and white for objects (see Fig 3.3.5).



Original Image T=140



T= 50 T= 80



T= 240

Figure 3.3.5: Tresholded Images with Different Treshold Values

In this image set the best segmentation is obtained with T=140. In the images, which have lower threshold, values cannot separate the objects from the background. Due to illumination and reflectance, some regions of background are also defined as object.

In the last image which has a “240” threshold value, although there is no confusion about background, it can be seen that some of the objects can not be separated properly and even there is an object which is completely defined as background.

As it can be seen form Fig 3.3.5, the threshold value is extremely important in image segmentation. In the designed system, the program leaves the determination of the threshold value to user and allows user to check if it is the right threshold value by showing the tresholded images.

CHAPTER 4

OBJECT RECOGNITION MODULE

In this chapter, the object extraction (including the object and region labeling), geometric properties of an object, recognition strategies and the recognition strategy followed in this project are discussed.

The object recognition problem can be defined as a labeling problem based on models of known objects. Given an image containing one or more objects including their background and a set of labels corresponding to a set of models known to the system, the system should assign correct labels to correct objects or regions in the image.

To design a proper object recognition system, the components below should be considered (Fig 4.1 shows the relationship between these components):

- Model database
- Feature Detector
- Classifier

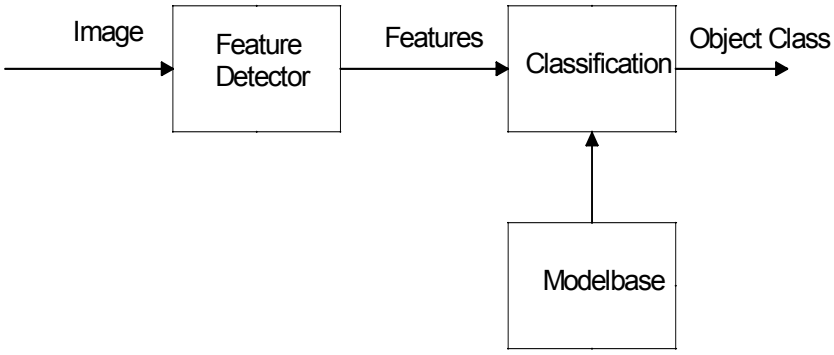


Fig 4.1: The Relationship Between the Components

The model base contains all the models known to the system. The information in the model database is mostly the set of features of the objects. A feature is some attribute of the object that is considered important in describing and recognizing the object among other objects. Color, shape and size are most commonly used features.

The goal of a feature detector is to characterize an object to be recognized by measurements whose values are very similar for objects in the same category and very different for objects in different categories. This brings an idea of “distinguishing features”. A feature detector applies a set of operators or algorithms to extract the required features defined by the designer. Feature detector produces the inputs of the Classifier. Choosing the right features and detectors to extract features reliably is the heart of recognition process. The features should describe the objects clearly also emphasize the difference between the object classes. The detector should be able to extract features as precise as possible also should not be sensitive to noise or environmental conditions.

The Classifier is the decision tool of a recognition system. The task of the classifier component is to use the feature information provided by feature extractor to assign object to a category (Object class).

Actually a hundred percent performance is generally impossible. The difficulty of the classification problem depends on the variation of the features for objects in the same category relative difference between the feature values for objects in different categories. This variety may be due to the complexity of the system or the noise in features [15].

4.1 Feature Extraction

In a recognition system, to recognize the objects in an image, some attributes of the object called “features” must be calculated. In most industrial applications, using simple geometric properties of objects from an image, the three-dimensional location of the objects can be found. Moreover in most applications the number of different objects is not large. If the objects are different in size and shape, the size and shape features of the objects may be very useful to recognize the objects. Many applications in industry have utilized some simple geometric properties of objects (like size, position, orientation) for determining the location and recognizing.

The task of recognition starts with object extraction. As discussed in Chapter 3, initialization process results with a tresholded binary image. The objects in the image can be extracted from this binary image.

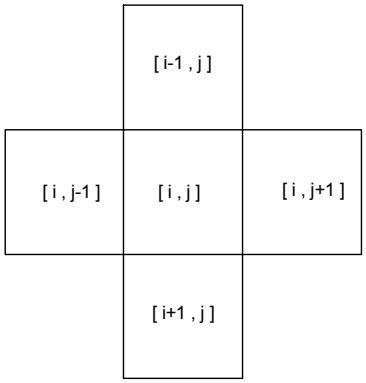
4.1.1 Object Extraction

A connected component or a region represents an object. This means that the system should search for connected components or regions. In order to achieve this task, the system applies some binary algorithms to the image. The fundamental concept of binary algorithms and the algorithm structures will be discussed below [12].

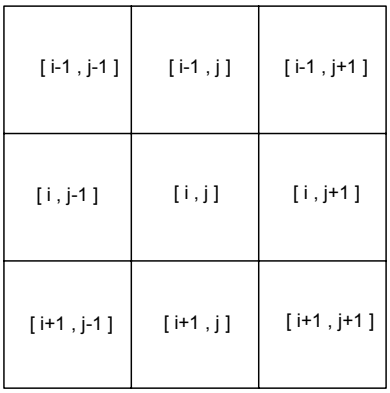
4.1.1.1 Fundamental Concepts

Neighbors :

In a digital image represented in a square grid, a pixel has a common boundary with 4 pixels and shares a corner with four additional pixels. It is called that two pixels are 4-neighbors if they *share* a common boundary. Similarly two pixels are 8-neighbors if they share at least one corner (see Fig 4.2).



4-neighbors of pixel [i, j]



8-neighbors of pixel [i, j]

Figure 4.2: 4 and 8 Neighbor pixels

Foreground:

The set of all 1 pixels in a binary image is called the foreground and is denoted by S.

Connectivity:

A pixel $p \in S$ is said to be connected to $q \in S$ if there is a path between p and q consisting entirely of pixels of S.

Connected Components:

A set of pixels in which each pixel is connected to all other pixels is called a connected component. A connected component also represents an object in a binary image.

Background and Holes:

The set of all connected components of complement of S that have pixels on the border of an image is called background. All other components are called as holes.

Boundary:

The boundary of S is the set of pixels of S that have 4-neighbors in complement of S.

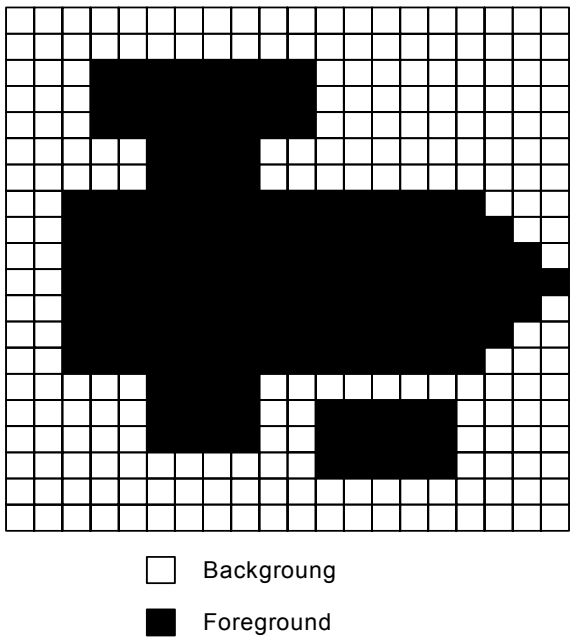


Figure 4.3: Example to Background and Foreground concepts

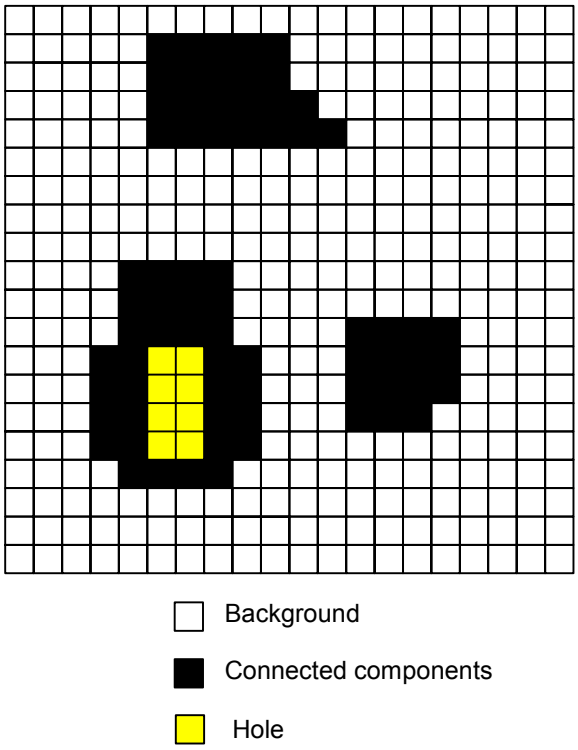


Figure 4.4: Example to Background, Connected component and Hole concepts.

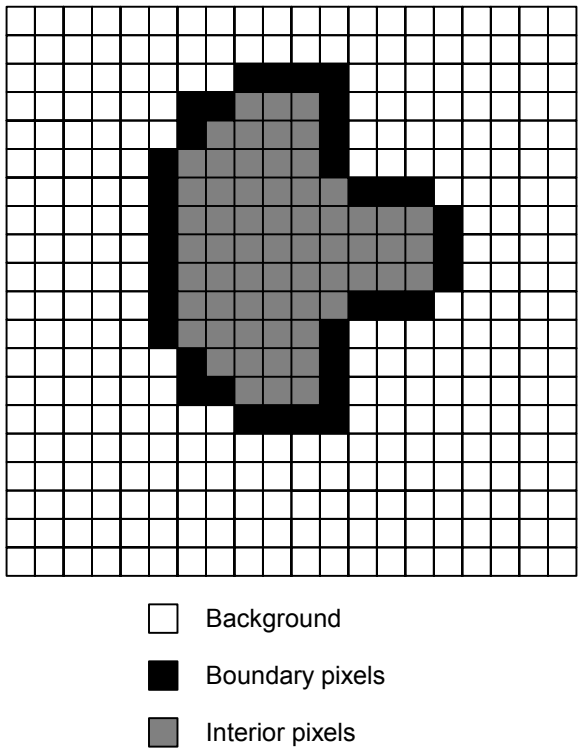


Figure 4.5: Example to Boundary and Interior pixels concept

4.1.1.2 Connected Component Labeling

One of the most common operations in machine vision is finding the objects in an image .The points in a connected component form a candidate region for representing an object. If there is only one object in an image, then there may not be a need for finding the connected components. But if there is more than one object in an image then the properties and locations should be found. In this case connected components must be determined.

A sequential algorithm can be used for labeling. This algorithm looks at the neighborhood of a pixel and tries to assign already used labels to a 255 pixel. In case of two different labels in the neighborhood of a pixel, an equivalent table is prepared to keep track of all labels that are equivalent. This table is used in the second pass to refine the labels in the image.

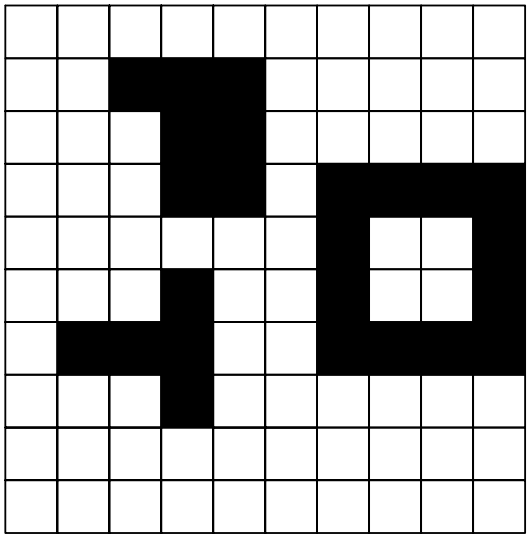
The algorithm scans image left to right, top to bottom and looks at only the neighbors at the top and the left of the pixel of interest. These neighbor pixels have already been seen by the algorithm. If none of these pixels' value is 255 then the pixel of the interest needs a new label. If only one of its neighbors is labeled, then the pixel is labeled with the same label. If both neighbors have the same label then the pixel is labeled with the same label. In the case where both neighbors are labeled with different labels than the algorithm assigns the smaller label to the pixel and record these two labels to the equivalence table [12].

After having finished scanning, the algorithm rescans the array to refine the labels according to the equivalence table. Usually the smallest label is assigned to a component. The algorithm is given in the Algorithm 4.1 below.

Sequential Connected Components Algorithm

1. Scan the image left to right, top to bottom.
2. if pixel is 255 , then
    a) If only one of its upper and left neighbors has a label, then copy the label.
    b) If both have the same label, then copy the label.
    c) If both have different labels, then copy the upper's label and enter the labels in the equivalence table as equivalent labels.
    d) Otherwise assign a new label to this pixel
3. If there are more pixels to consider, then go to step 2.
4. Find the lowest label for each equivalent set in the equivalence table.
5. Scan the array. Replace each label by its lowest equivalent label.

An example result for sequential algorithm can be seen from Figure 4.6



Original Image

		1	1	1					
			1	1					
			1	1		2	2	2	2
						2			2
			3			2			2
	3	3	3			2	2	2	2
			3						

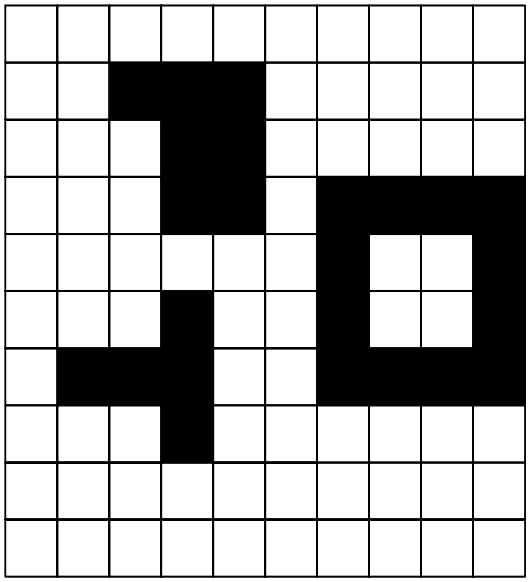
Labelled Array

Figure 4.6: Object Labeling

Region Labeling is very similar to connected component labeling. As mentioned earlier, an object in an image is a region. But note that regions can represent also the background or holes of the objects in an image.

In some applications whether they are object or not, the region information may be required.

The algorithm of this operation is very similar to the algorithm described above. The main difference between them is that the algorithm is not searching all the 255 pixels. This means that the algorithm labels also the background with the label ”0” (see Fig 4.7 and Fig 4.8).



Original Image

Figure 4.7: Input Image to the Algorithm

0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0
0	0	0	1	1	0	2	2	2	2
0	0	0	0	0	0	2	3	3	2
0	0	0	4	0	0	2	3	3	2
0	4	4	4	0	0	2	2	2	2
0	0	0	4	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Labelled Array

Figure 4.8: The Output of the algorithm

4.1.2 Geometric Properties

Geometric properties of an object are important hints about its geometry. In most industrial applications the 3D location of the objects can be found using 2D locations in the image. Generally the number of different objects is not large. If the objects are different in size and shape, the size and shape features of the objects may be determined from their images to recognize them.

After having extracted objects in an image these features can easily be found. These geometric properties can be written as:

1. Area (Size) of the object
2. Position of the object
3. Orientation of the object
4. Compactness of the object.
5. Euler Number of the object.

4.1.2.1 Area

The Area Feature is a measure of the size of the object in an image. Once the objects are labeled in an image, all objects in the image can be analyzed individually, and the area feature can be easily obtained by computing the number of the pixels in the object of interest. Mathematically it can be written as:

Let B [i, j] is the i th row, j th column element of labeled image array B which has n rows and m columns.

If B [i, j] is a pixel element of object O_k then the area of the object O_k is:

$$A_{O_k} = \sum_{i=1}^{n}\sum_{j=1}^{m} B[i,j] \quad \text{where} \quad B[i,j] = \begin{cases} 1 \text{ if } B[i,j] \in O_k \\ 0 \quad \text{otherwise} \end{cases} \tag{4.1}$$

This is also the zeroth-order moment.

		1	1	1					
			1	1					
			1	1		2	2	2	2
						2			2
			3			2			2
	3	3	3			2	2	2	2
			3						

Labelled Array

Figure 4.8: A Labeled Array

For instance in an image like the one in Fig. 4.8 the areas of the objects will be: A1 = 7 pix, A2 = 12 pix, A3 = 5 pix.

4.1.2.2 Position

The position of an object in an image plays an important role in many applications. In industrial applications, usually objects are on a known surface, such as a table. If the cameras are calibrated, the 3D position of the object can be computed by using the 2D position of the object in image.

The position of an object in an image can be computed using the center of area of the object in an image. The center of the area in binary images is the same as the center of mass, if all pixels are assumed as mass. With this assumption the position of the object "k" in an image can be written as:

$$\overline{x}_k = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} jB[i,j]}{A_k} \tag{4.2}$$

$$\overline{y}_k = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} iB[i,j]}{A_k} \tag{4.3}$$

Where $B[i,j] = \begin{cases} 1 \text{ if } \mathrm{B[i,j]} \in \mathrm{k} \\ 0 \end{cases}$ and A_k is the area of the object "k".

4.1.2.3 Orientation

The calculation of orientation of the object is rather complex compared to the calculation of other features.

The logic behind the calculation of orientation of the object is to use the orientation of the axis of elongation of the object. Usually the axis of second moment is used as the axis of elongation.

The axis of second moment for an object image is the line, which the sum of the squared distances between the object points and the line is least.

This minimization can be written as:

$$X^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij}^2 B[i, j] \tag{4.4}$$

Where r_{ij} is the perpendicular distance between the object points and the line. If the line is represented in polar coordinates (see Fig. 4.9).

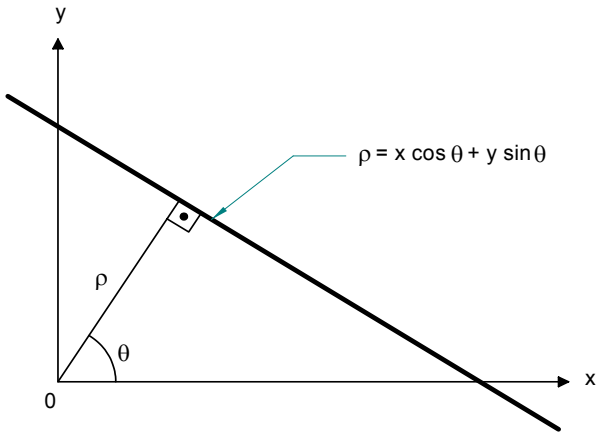


Figure 4.9: Representation of a line

$$\rho = x\cos\theta + y\sin\theta \tag{4.5}$$

Then the distance of a point (x, y) is obtained by plugging the coordinates of the point in to the equation for the line

$$r^2 = (x\cos\theta + y\sin\theta - \rho)^2 \tag{4.6}$$

Plugging this equation in to the minimization criterion leads:

$$X^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij}\cos\theta + y_{ij}\sin\theta - \rho)^2 B[i,j] \tag{4.7}$$

Setting the derivative of X^2 with respect to ρ to zero and solving for ρ yields:

$$\rho = \overline{x}\cos\theta + \overline{y}\sin\theta \tag{4.8}$$

Then substitute this value to equation (4.7) and replace:

$$x' = x - \overline{x}$$
$$y' = y - \overline{y}$$

Then the minimization becomes:

$$X^2 = a\cos^2\theta + b\sin\theta\cos\theta + c\sin^2\theta \tag{4.9}$$

Where

$$a = \sum_{i=1}^{n}\sum_{j=1}^{m}(x'_{ij})^2 B[i,j]$$
$$b = 2\sum_{i=1}^{n}\sum_{j=1}^{m}x'_{ij}y'_{ij}B[i,j]$$
$$c = \sum_{i=1}^{n}\sum_{j=1}^{m}(y'_{ij})^2 B[i,j]$$

the second order moments

The minimization criterion then can be written as:

$$X^2 = \frac{1}{2}(a+c) + \frac{1}{2}(a-c)\cos 2\theta + \frac{1}{2}b\sin 2\theta \tag{4.10}$$

Differentiating X^2, setting the result to zero, and solving for θ yields:

$$\tan 2\theta = \frac{b}{a-c} \tag{4.11}$$

4.1.2.4 Compactness

Compactness is simply the measure of circularity of an object. A circle is the most compact figure (the smallest compactness value- 4π) according this measure and it can be computed by the formula:

$$Compactness = \frac{P^2}{A} \geq 4\pi \tag{4.12}$$

Where P is the perimeter of the object and A is the area of the object.

In a digital image perimeter of an object can be easily defined as the number of the boundary pixels in the object of interest. In order to calculate the perimeter of an object, boundary pixels should be extracted first. After having extracted boundary pixel, the number of these pixels must be counted according to the Euclidean Distance Measure. By using this measure, the 4-neighbor of a pixel is counted as “1” pixel, on the other hand the 8-neighbor of a pixel is counted as “$\sqrt{2}$”pixel. Below in Algorithm 4.2 a Boundary-Following Algorithm is given.

Algorithm 4.2 Boundary-Following Algorithm

1. Find the starting pixel s $\in$ S for the region using systematic scan (from left to right, from top to bottom of the image).
2. Let the current pixel in boundary tracking be denoted by *c*. Set *c* = *s* and let the 4-neighbor to the west of s be $b \notin S$.
3. Let the 8-neighbors of *c* starting with *b* in clockwise order be $n_1, n_2, \ldots, n_8$. Find n_i, for first *i* that is in *S*.
4. Set $c = n_i$ and $b = n_{i-1}$.
5. Repeat steps 3 and 4 until c = s

For the result of the Boundary-Following Algorithm see Fig. 4.10

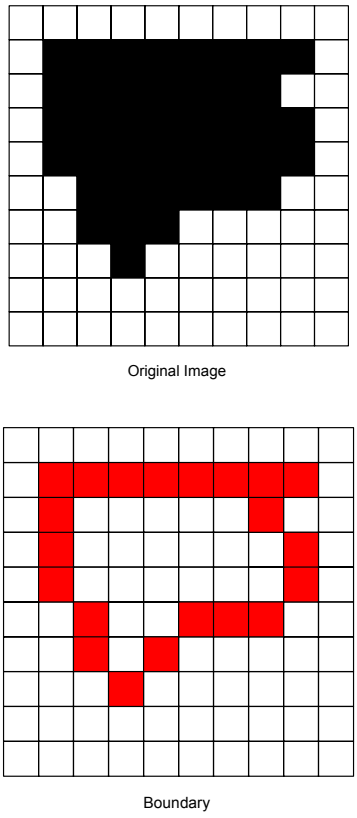


Figure 4.10: Input and Output of Boundary Following Algorithm

4.1.2.5 Euler Number

In many applications, the genus or Euler Number is used as a feature of an object. Euler Number is defined as the number of the components minus the number of holes. Thus,

$$E = C - H$$

Where E is the Euler Number, C is the number of components and H is the number of holes.

4.2 Classification

Classification is simply assigning the input in to one of pattern classes based on features. A pattern class can be defined as a set of patterns known to originate from the same source, sharing some common attributes.

Classification process needs some hints about the objects to be classified or recognized. These hints are called features. This means that a classifier needs a feature space to make decision. So the goal for a classifier is to determine a decision boundary in the feature space.

Assume that a system is required to classify objects in to object classes 1 and 2 (see Fig 4.12).

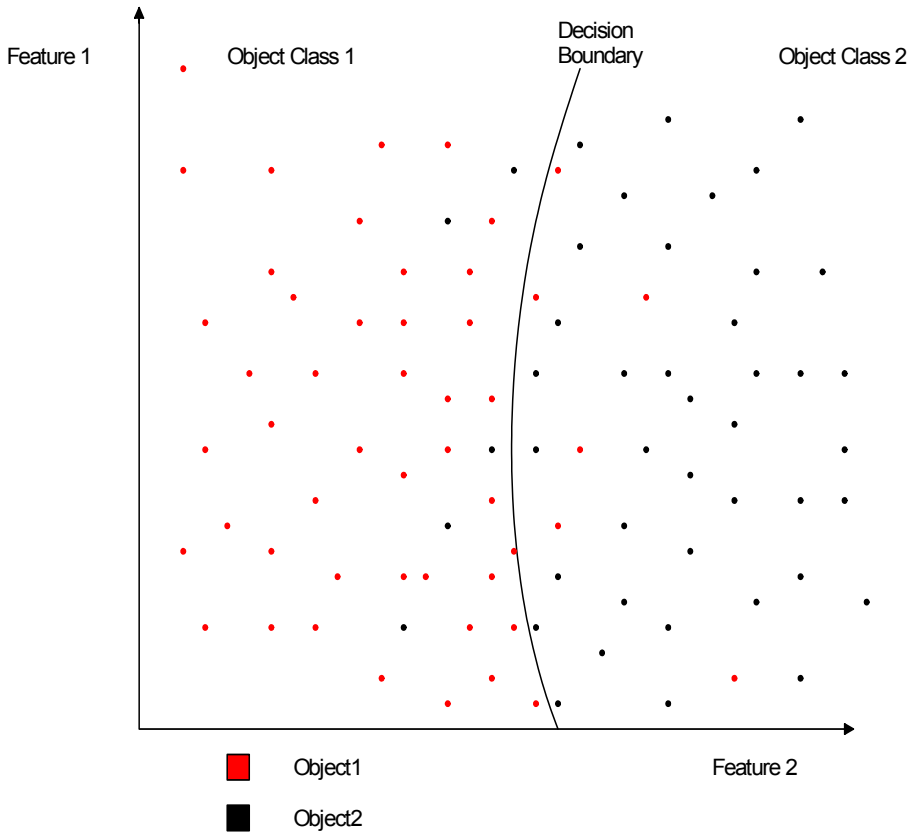


Figure 4.12: Sample Feature Space and Decision Boundary

As it can be seen in Fig. 4.12 an optimal decision boundary is shown according to the feature space. There are some misclassified objects in this classification but it should be noticed that a proper decision boundary, which classifies the objects completely correct for above objects, would definitely misclassify the new-coming objects.

There are different approaches in pattern classification. These approaches can be grouped in to three groups [15]. These are:

1. Statistical Approach:

   The Statistical approaches mainly concentrate on the probabilities of the features, objects and classes .The structure of the objects to be classified is ignored.

2. Structural Approach:

   The Structural Approaches depend on the structural information that describes the patterns. For example sides, top surface, bottom surface of a cylinder.

3. Neural Approach:

   The Neural Approach is a relatively new approach in Pattern Recognition involving interconnected networks of nonlinear units called neural nets. This approach is applied by training the network as an initialization.

The approach used in the designed system is rather different and is simpler than the approaches discussed above. As mentioned earlier the system is designed to separate geometric primitives (metal parts) to “Cylinder”, ”Cylinder Template”, ”Prism” and ”Prism Template” pattern classes.

Statistical approach is based on the probability density function of features and object classes as mentioned above. In order to apply this approach efficiently and properly, the number of the objects to be recognized should be large. And sampling is needed to form an initial model base of the classes.

Structural approach deals with the structural features of the objects to be recognized. There is no need to have large number of objects or sampling because the objects, which can be classified to the same class using this approach, will definitely have same or similar structural features. But in practice it is really hard to describe the objects in a structural way. Neural Approach is a really nice and new solution for a pattern recognition problem. Neural Networks are used in many artificial intelligence applications but it is also not a reliable or efficient approach like other approaches to classify limited number of geometric primitives in to 4 classes, due to the requirement of training. Training procedure also needs sampling. But in the designed system the parts to be grasped are invariant. The number of the parts, the shape of the parts and the size of the parts are constant.

In such system Statistical Approaches or Neural Approach will work. However they will make the system not learning one but memorizing one. Memorization is the main inefficiency in these methods.

4.3 Implementation

As discussed earlier, the goal of a classifier is to determine a decision boundary in the feature space, and the features that are used in the designed system are some geometrical properties of the object in a binary image including area, position, orientation, perimeter, compactness and Euler Number. It is very important to select the features, which will be the most dominant ones in recognition. The measure of being dominant is of course the ability of characterizing each object class and to distinguish them.

In order to classify cylinders, cylinder templates, prisms and prism templates, the system uses Euler number and compactness features of the objects.

After the initialization of both images captured by the cameras, the tresholded images are read to the corresponding 2D unsigned arrays for labeling by using a function, which takes the tresholded image and an empty array and then returns the array filled with the pixel values of the tresholded image as an output. After having read the tresholded images, both arrays are fed in to the labeling function. This function labels the objects in the tresholded images and gives the number of the objects in the image and a 2D array with the same size of the image including the labeled objects by using the method described in section (4.1.1.2).

Now the system is ready to extract features. The code takes the number of the objects in the image, the labeled array as inputs and returns the area, x component of the center and y component of the center of the each object in the labeled array.

To calculate the compactness of the objects in the image, the system first finds the perimeter of the objects. The code takes the labeled array as input and returns an array, which can be viewed as boundary pixel image and the perimeters of each object in the image by using the method described in section (4.1.2.4).

After having computed the perimeters of the objects, the system easily computes compactness. Then the system finds the Euler number of the objects in the image. The code takes the tresholded image as an input and returns the Euler number of each object as output.

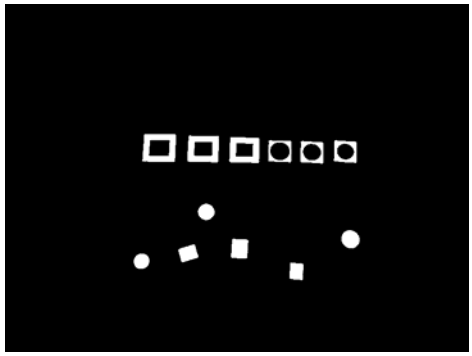
The first step of classification starts here. The system separates the parts including cylinders and prisms from their respective templates by using the Euler number.

The objects that has Euler number of 1 are assigned to "*Part Class*" and that has Euler Number of '0' are assigned to "*Template Class*" due to the holes.

The next step is to separate the *Part Class* to "*Cylinder Class*" and "*Prism Class*". For this purpose the system compares the compactness of the objects in the Part Class with a threshold given by the programmer.

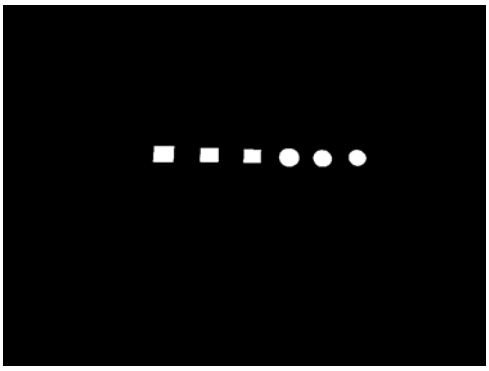
As discussed earlier compactness is the measure of the circularity. It is expected that the compactness values of cylinders must be smaller than prisms and the minimum compactness value to be 4π (12.56). It is tested that a compactness threshold of 13 is completely successful in separating the cylinders from prisms correctly. So the system assigns the members of *Part Class* which has smaller compactness value than 13 to *Cylinder Class* and which has bigger compactness value than13 to *Prism Class*. Now the system knows exactly which part is a cylinder and which part is a prism.

After having recognized the parts, the system concentrates on the "*Template Class*" because the system knows nothing about the members of *Template Class* except that they have holes. Actually the system should not interest in the features of the templates but in the features of the holes of the templates because the all templates have rectangular shape. Also it is more reliable to use the hole instead of template during the assembly process due to insertion of the parts. Due to this requirement the system extracts the holes in the tresholded images, which comes from initialization process (see Fig 4 .13).



Original Tresholded Image

Figure 4.13: Hole Extraction



Hole extracted Image

Figure 4.13(cont.): Hole Extraction

After extracting the holes the system labels this image and calculates all the objects' (holes') features .The same procedure is followed during the operation as described above for the parts. Again the compactness values of the objects (holes) are compared with the same threshold and assigned to "*Cylinder-Template Class*" if compactness is lower than 13, else it is assigned to "*Prism-Template Class*". Then the members of the *Template Class members* are updated according to the corresponding holes to relate the holes with the *Template Class members*.

The flowchart of the classifier is given in Fig. 4.14

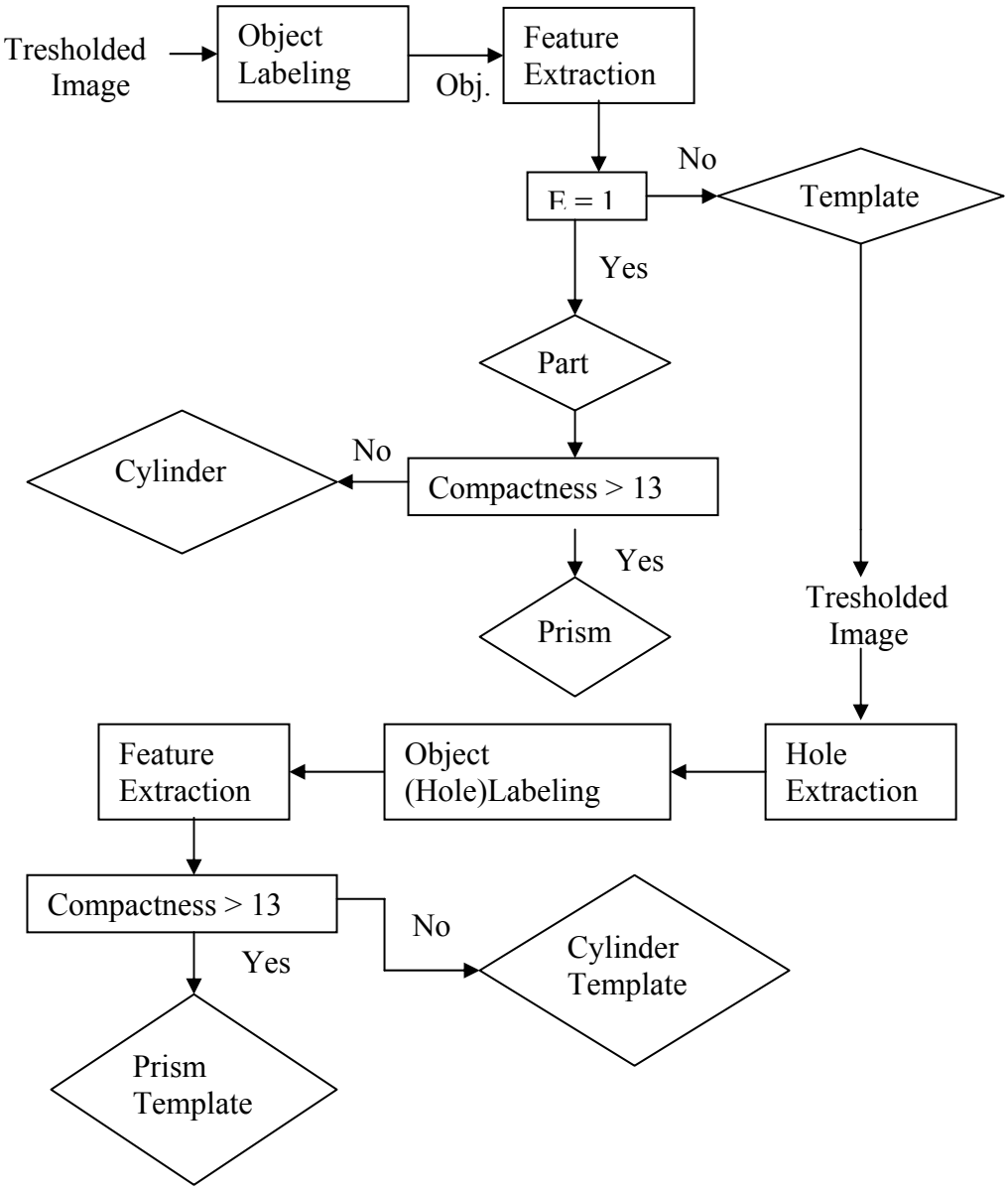


Figure 4.14: Recognition Flow Chart

With this approach the decision boundary of the classifier successfully classifies the objects in the image (see Fig 4.15).

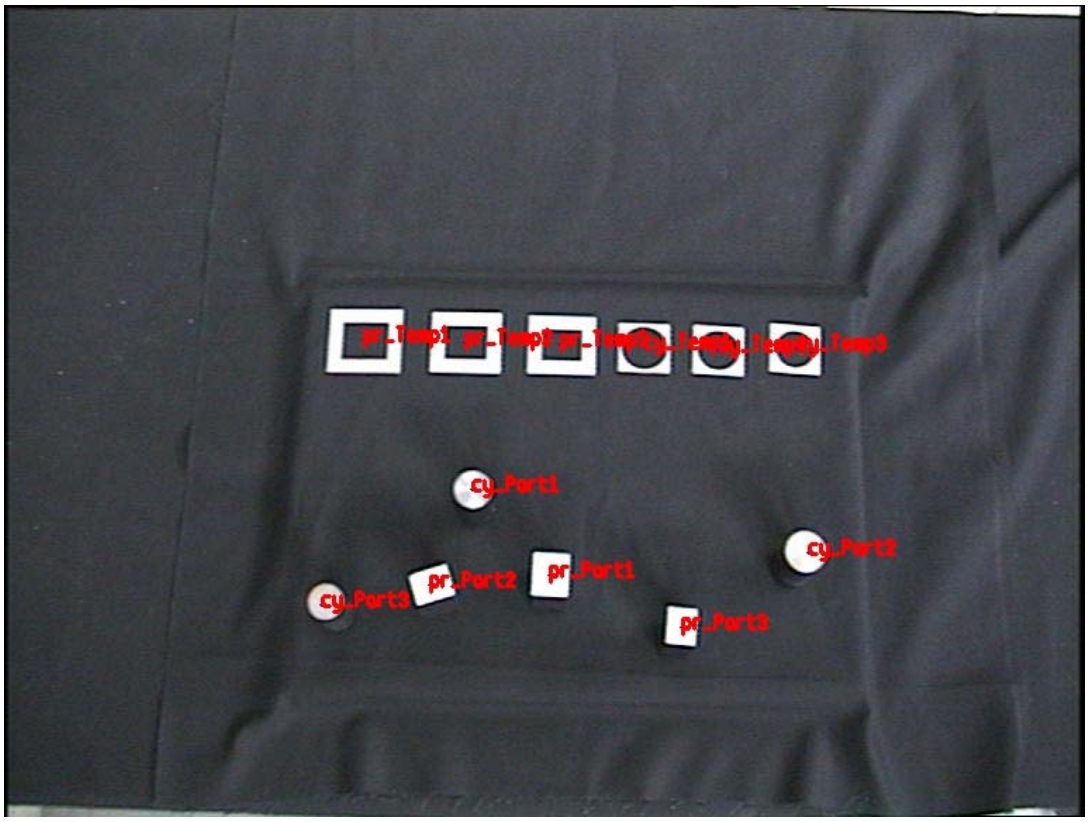


Figure 4.15: Result of Recognition Module

CHAPTER 5

DEPTH EXTRACTION AND ASSEMBLY MODULE

In this chapter extraction of the three dimensional coordinates of the objects and the assembly strategy will be discussed. Calculating the distance of various points in the scene relative to the position of the camera is one of the most important tasks for a computer vision system. A common method for depth extraction is to use a stereo system. A stereo system involves two cameras for image acquisition. There are also different methods like capturing two or more images from a moving camera or using range-imaging systems. The 3D information can also be extracted from the clues that can be obtained from 2D images such as shading and texture.

5.1 Stereo Imaging

The geometry of a stereo system can be figured as in Fig 5.1 and Fig 5.2.

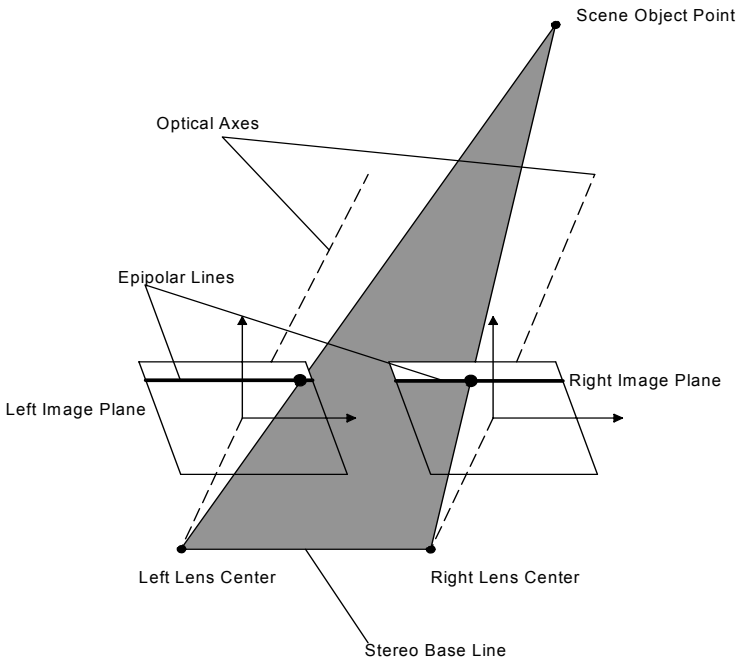


Fig 5.1: Simple Stereo Imaging Geometry

The image planes are coplanar in Fig. 5.1. A feature in the scene is viewed by two cameras at different positions in the image plane .The displacement between the locations of the two features in the image plane is called disparity. The plane passing through the camera centers and the feature point is called the epipolar plane. The intersection of the epipolar plane with the image plane defines the epipolar line. In this case every feature in one image will lie on the same row in the second image [12].

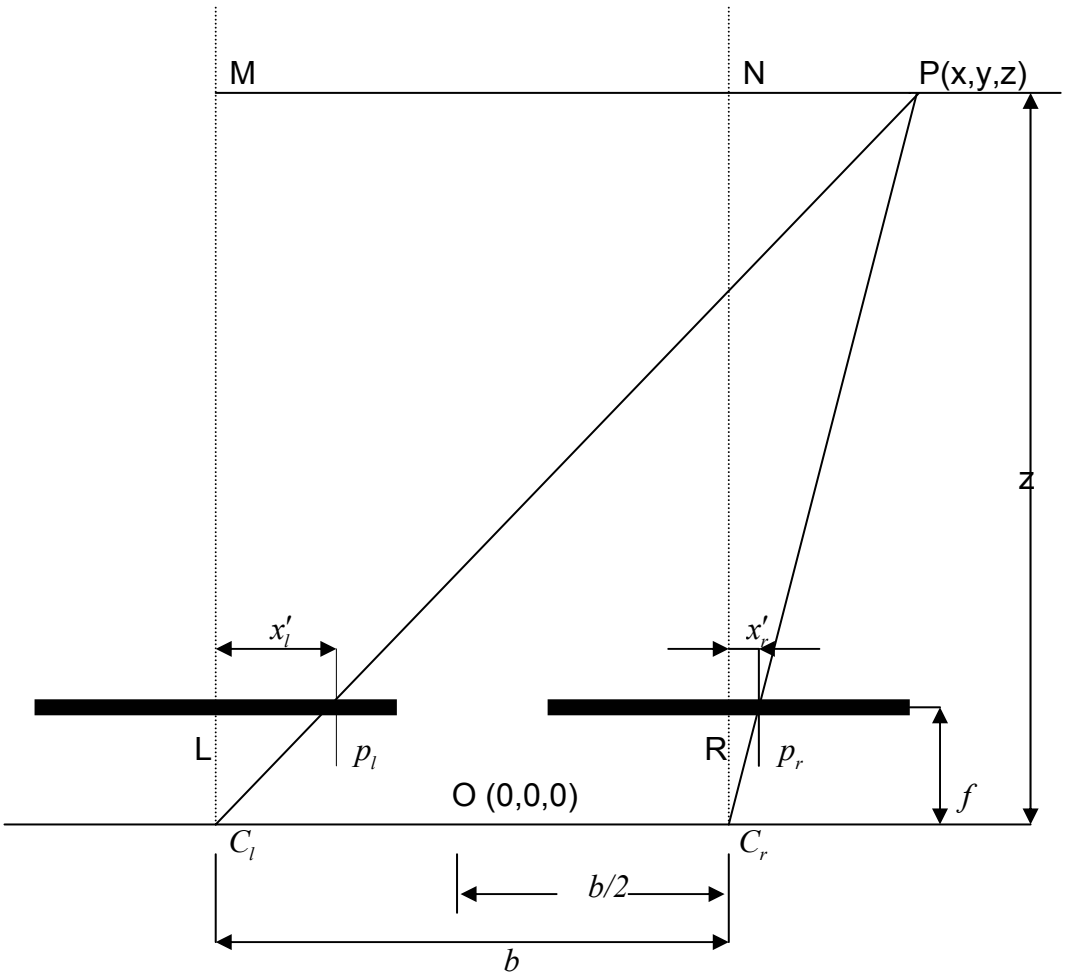


Figure 5.2: Simple Stereo Imaging Geometry

In Figure 5.2 the scene point is observed at points p_l and p_r in the left and right image planes. If it is assumed that the origin of the coordinate system is on the middle point of lens centers, by comparing the similar triangles P-M-C_l and p_l-L-C_l, it can be written as:

$$\frac{x+b/2}{z}=\frac{x_l^{'}}{f} \tag{5.1}$$

Similarly, from the similar triangles P-N-C_r and p_r-R-C_r;

$$\frac{x_r^{'}}{f} = \frac{x - b/2}{z} \tag{5.2}$$

$$\frac{y_l^{'}}{f} = \frac{y_r^{'}}{f} = \frac{y}{z} \tag{5.3}$$

Then;

$$x = b\frac{(x_l^{'} + x_r^{'})/2}{x_l^{'} - x_r^{'}} \tag{5.4}$$

$$y = b\frac{(y_l^{'} + y_r^{'})/2}{x_l^{'} - x_r^{'}} \tag{5.5}$$

$$z = \frac{bf}{x_l^{'} - x_r^{'}} \tag{5.6}$$

Thus, the depth at various scene points can be extracted by calculating the disparities of the corresponding image points.

As it can be seen above, to extract the 3D coordinates of an object or a point, it must be ensured that the object point has projections in both image planes. In other words, both cameras should see the object. The projections of an object point in image planes are called conjugate pairs. With out extracting conjugate pairs of an object in the image planes, the 3D coordinate extraction of the object of interest is not possible. The problem of determining the conjugate pairs of an object point in the image planes is called correspondence problem.

5.2 Stereo Matching

The correspondence problem can be stated as: for each point in the left image, find the corresponding point in the right image. It is necessary to measure similarity of these points of interest, to define them as the conjugate pairs. The candidate points should be distinctly different from the surrounding pixels; otherwise all pixels would be matches. So defining good and matchable features is very important. Both edge features and region features are used in stereo matching.

The epipolar constraint is one of the most useful information in stereo matching. As shown in Figure 5.1 the projections of an object point will be on the epipolar lines in left and right images. So the knowledge of the epipolar lines will reduce the correspondence problem to a 1D search. However due to measurement errors and other uncertainties in camera position and orientation, matching points may not occur exactly on the estimated epipolar lines in the image plane. In this situation a small neighborhood of the estimated line must be searched. The stereo matching methods can be grouped in to two [2], [18]:

5.2.1 Region Based

These methods utilize the image domain similarity metrics in the correspondence process. Given any two views of the same scene it can be seen that, at some scale, there is a degree of similarity between two images. The coarser the scale is, the more similar the images become.

Because the disparity of a point has a proportionally smaller effect at large image scales and at coarser image scales there is a reduction in the quantity of discernible features, only dominant features exist. If a view is divided to small sub regions then any sub-region will begin to look more similar to the corresponding sub-region of the other view. Thus it is possible

to apply a region based similarity metric to find the correspond of a sub-region in the other view by searching through the epipolar line.

5.2.2 Feature Based

These methods perform stereo matching with high-level parameterization called image features. Many areas of computer vision like object recognition or tracking are feature based applications. Since stereovision involves extracting three-dimensional data from the scene, the features, which are useful, are the features, which describe the 3D structure of the scene. Generally the manmade environments can be described with edges and edge intersections called corners. Feature based stereo matching algorithms uses edges and corners. The epipolar constraint is also applicable but horizontal edges cannot be matched. Edges and their possible matches are compared according to orientation and strength. Orders of the edges in the images and the intensity information along the edge across two sides are also used for matching the edges.

Circles, ellipses and polygonal regions are also used in indoor environments as features for matching. The areas of these features in pixels and coordinates of centers can be used to match these features. Ordering is also another criterion for these features

In their most simplistic form region based approaches involve subdividing the whole view in to sub regions and applying a photometric similarity measure to all regions.

The aim of this type of algorithm is generally to return a dens depth map of the view, where depth is calculated at every pixel in the scene. But feature based approaches give sparser depth information. Feature Based approaches can detect conjugate pairs robustly and faster than region based approaches. If a depth map of a scene is needed, feature based approaches

are not suitable because these approaches requires interpolation to get a depth map.

The designed system works in an indoor manmade environment and the aim is to grasp the parts and assemble them to the corresponding templates. The system is working with geometric primitives. The scene that the system percepts, includes just these geometric primitives and a large uniform black background (see Figure 5.3).

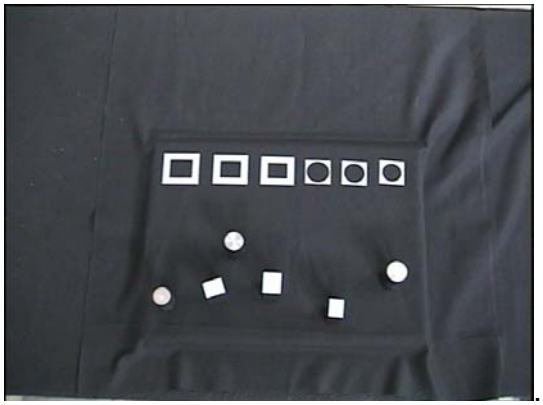


Figure 5.3: The Operating Environment

Due to the structure of the environment and the purpose of the operation a feature-based approach is applied. The approach can be called as Object-based stereo matching. This approach uses the objects itself as region features for matching. For similarities the geometric properties of the objects are used.

Since the main idea is to find such points that can be easily used to align with the robot's tip point for grasping, pixels which can clearly describe

the location of the objects and which can be aligned with the tip point of the robot are required.

Obviously, the most suitable points are the center of gravity points of the objects' top surfaces. Using real center of gravity of the objects will cause mismatches due to the effects of the side portions of the objects projected on the image (see Figure 5.4).

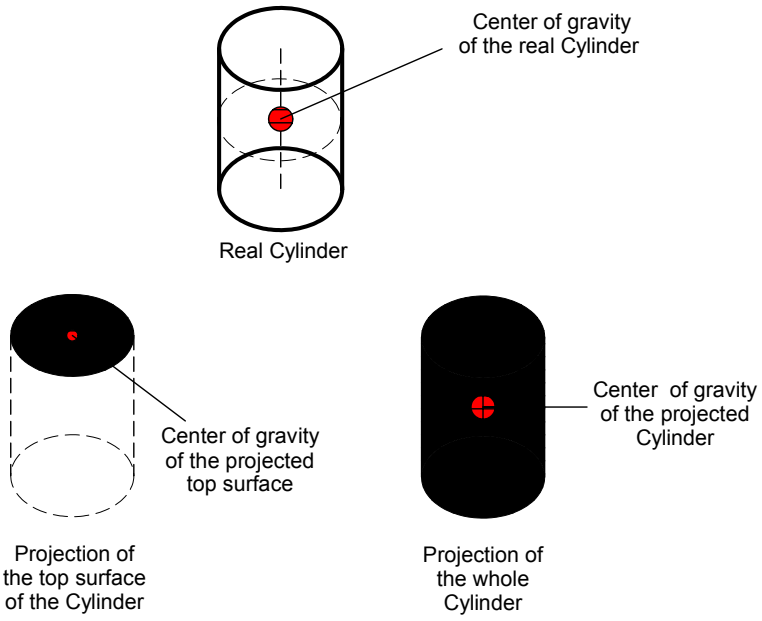


Figure 5.4: Center of Gravity Representations

As it can be seen from Fig. 5.4 center of the projection of the cylinder has no physical meaning due to the projection of an unseen point. The system will not be able to distinguish this point from a point that is on the side surface of the cylinder. Their projections will be the same. However the center of the projection of the top surface can easily be used due to being the same point in 3D.

The system takes the labeled arrays of both images and the epipolar tolerance as inputs and gives an array, which holds the matching results. The matching code searches the center of the gravity of the objects in the left image in the labeling order (appearance order in the image). After having found the center, the function specifies the epipolar line, which corresponds to the y or the row component of the center. Using this value and the epipolar tolerance, the function creates a region, which is the +/- neighborhood for the epipolar line extracted from left image, in the right image. After testing the system by comparing the image pixel coordinates of the centroids of the same objects in the both images, it is concluded that ± 15 pixel is sufficient for epipolar tolerance during matching. This region is the search space for the possible matches in the right image. The function register all possible matches – the objects in the right image, which has center in this region – to an array except the ones that are matched before. Then using a similarity function, the system calculates the similarity between the region in the left image and the candidates in the right image. The candidate with the highest similarity value is assigned as “the match” to an array.

The similarity function can be written as:

$$s(L_i, R_j) = \sum_{p=1}^{N} w_p (s_p(L_i^{\,p}, R_j^{\,p}) \tag{5.7}$$

Where N is the number of features compared, i and j are the indexes of the objects in the left and right images respectively, w_p is the weight of the p th feature defined by the designer and :

$$s_p(L,R) = 1 - \frac{\min(A_p(L), A_p(R))}{\max(A_p(L), A_p(R))} \tag{5.8}$$

Where A_p is the value of the p th feature. In the similarity function, area, euler number, and compactness features are used [11].

This approach is applied to both parts and templates. These two types of objects are matched separately due to the need of concerning the features of the holes in the templates. Since the parts will be assembled in to the holes of the templates, the centers of the holes are the points of interest. The holes are matched instead of template itself. Euler number feature is ignored in matching function while processing holes.

With this approach the objects in the scene are successfully matched. In Fig 5.5 first image is the left image and the second is the right image. The red numbers in the left image indicate the object indexes and the blue numbers in the right image indicate the index of the object that is matched with.

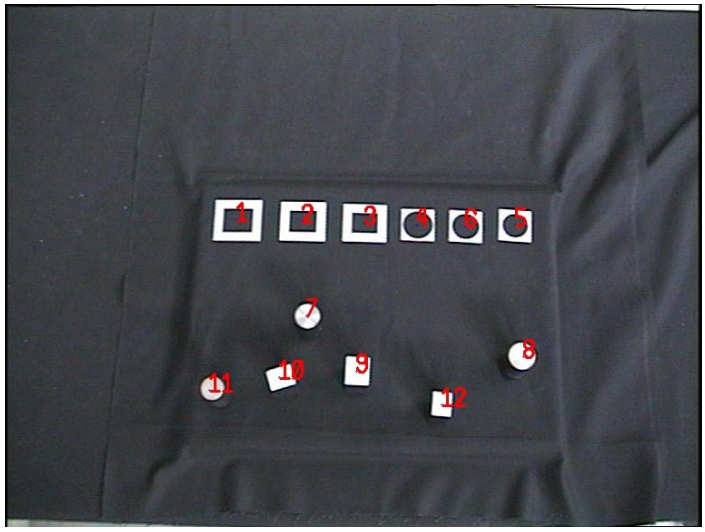


Figure 5.5: Matching Results

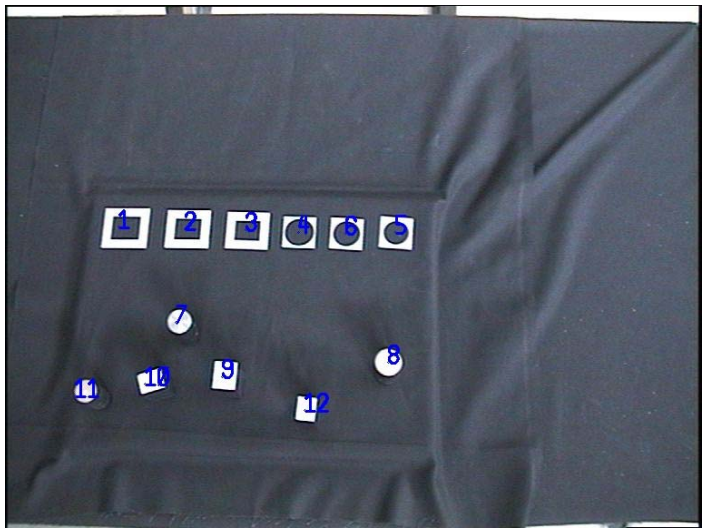


Figure 5.5 (cont.): Matching Results

5.3 Depth Extraction

As discussed earlier in Chapter 2, the transformation between the image coordinates and 3D model coordinates can be written as;

$$\tilde{\mathbf{m}} = \mathbf{A}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\tilde{\mathbf{M}} \tag{5.9}$$

If A is written in the form of:

$$\mathbf{A} = \begin{bmatrix} \alpha & \gamma & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.10}$$

And the term [R t] in the form of:

$$\mathbf{RT} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.11}$$

Where $\mathbf{0} = [0 \quad 0 \quad 0]$, R is the 3 by 3 rotation matrix and t is the 3 by 1 translation vector.

Also $\tilde{\mathbf{m}} = [u, \quad v, \quad 1]^T$ and $\tilde{\mathbf{M}} = [X, \quad Y, \quad Z, \quad 1]^T$. Then the equation 5.9 can be written as:

$$\tilde{\mathbf{m}} = \mathbf{GT}\tilde{\mathbf{M}} \tag{5.12}$$

Where $\mathbf{GT} = \mathbf{ART} = \begin{bmatrix} GT_{11} & GT_{12} & GT_{13} & GT_{14} \\ GT_{21} & GT_{22} & GT_{23} & GT_{24} \\ GT_{31} & GT_{32} & GT_{33} & GT_{34} \end{bmatrix}$

GT is the geometric transformation including the camera intrinsic and extrinsic parameters.

Substituting $\tilde{m}$ and $\tilde{M}$ to the equation 5.12 yields:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{GT} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.13}$$

From above equation u and v can be written as:

$$u = GT_{11}X + GT_{12}Y + GT_{13}Z + GT_{14} / GT_{31}X + GT_{32}Y + GT_{33}Z + GT_{34} \tag{5.14}$$

$$v = GT_{21}X + GT_{22}Y + GT_{23}Z + GT_{24} / GT_{31}X + GT_{32}Y + GT_{33}Z + GT_{34} \tag{5.15}$$

If equation 5.14 is re-arranged;

$$(GT_{31}u - GT_{11})X + (GT_{32}u - GT_{12})Y + (GT_{33}u - GT_{13})Z = -(GT_{34}u - GT_{14}) \tag{5.16}$$

And same for equation 5.15 yields:

$$(GT_{31}v - GT_{21})X + (GT_{32}v - GT_{22})Y + (GT_{33}v - GT_{23})Z = -(GT_{34}v - GT_{24}) \quad (5.17)$$

Note that after the matching operation, the system has the conjugate pairs. By definition the conjugate pairs represent ***the same point*** in 3D.

Let the conjugate pairs (u_1, v_1) and (u_2, v_2) are obtained from the matching process, the intrinsic and extrinsic parameters of the cameras are known and given in the form of $\mathbf{GT}1$ and $\mathbf{GT}2$, and it is known that the conjugate pairs are the projections of the point $\mathbf{M} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$ in 3D.

Then re-writing equation 5.12 for both cameras yields:

$$\tilde{\mathbf{m}}_1 = \mathbf{GT}1\tilde{\mathbf{M}} \quad (5.18)$$

$$\tilde{\mathbf{m}}_2 = \mathbf{GT}2\tilde{\mathbf{M}} \quad (5.19)$$

If so, the equations 5.16 and 5.17 can also be written for both cameras:

$$\begin{aligned}&(GT1_{31}u_1 - GT1_{11})X + (GT1_{32}u_1 - GT1_{12})Y + (GT1_{33}u_1 - GT1_{13})Z \\ &= -(GT1_{34}u_1 - GT1_{14})\end{aligned} \quad (5.20)$$

$$\begin{aligned}&(GT1_{31}v_1 - GT1_{21})X + (GT1_{32}v_1 - GT1_{22})Y + (GT1_{33}v_1 - GT1_{23})Z \\ &= -(GT1_{34}v_1 - GT1_{24})\end{aligned} \quad (5.21)$$

$$\begin{aligned}&(GT2_{31}u_2 - GT2_{11})X + (GT2_{32}u_2 - GT2_{12})Y + (GT2_{33}u_2 - GT2_{13})Z \\ &= -(GT2_{34}u_2 - GT2_{14})\end{aligned} \quad (5.22)$$

$$\begin{aligned}&(GT2_{31}v_2 - GT2_{21})X + (GT2_{32}v_2 - GT2_{22})Y + (GT2_{33}v_2 - GT2_{23})Z \\ &= -(GT2_{34}v_2 - GT2_{24})\end{aligned} \quad (5.23)$$

These equations can be written in a matrix form:

$$\begin{bmatrix} GT1_{31}u_1 - GT1_{11} & GT1_{32}u_1 - GT1_{12} & GT1_{33}u_1 - GT1_{13} \\ GT1_{31}v_1 - GT1_{21} & GT1_{32}v_1 - GT1_{22} & GT1_{33}v_1 - GT1_{23} \\ GT2_{31}u_2 - GT2_{11} & GT2_{32}u_2 - GT2_{12} & GT2_{33}u_2 - GT2_{13} \\ GT2_{31}v_2 - GT2_{21} & GT2_{32}v_2 - GT2_{22} & GT2_{33}v_2 - GT2_{23} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$= \begin{bmatrix} -(GT1_{34}u_1 - GT1_{14}) \\ -(GT1_{34}v_1 - GT1_{24}) \\ -(GT2_{34}u_2 - GT2_{14}) \\ -(GT2_{34}v_2 - GT2_{24}) \end{bmatrix} \tag{5.24}$$

The 3D coordinates of the point M, can be found by solving equation 5.24 for X, Y and Z.

Note that in this approach, there is no need to calculate the baseline b or the disparity d. All parameters are known to the system after calibration and stereo matching operations. For extracting the 3D coordinates the system takes the intrinsic and extrinsic parameters of the cameras, the image coordinates of the conjugate pairs as input and gives an array of 3D coordinates as output. The system calculates the 3D coordinates of the objects one by one using the recognition and matching results.

5.4 Assembly

After having extracted the 3D coordinates of the parts and templates, the system assigns the parts to the templates according to a sequential assembly strategy.

This strategy forces the system to start assembly from the cylinder parts. There is also a sequence among the cylinders according to their sizes. The system orders the cylinders according to their areas.

The same operation is performed on the cylinder templates. After assigning the cylinders and their templates, same procedure is followed for the prisms and their templates.

Note that the sorting operation is based on areas of the objects. Every object has a different size so the system should never face an ambiguous situation in principle. However the difference between the sizes of the objects is very small and due to perspective principles the closer parts seem larger while farther objects seem smaller. This situation creates errors in sorting operation and causes wrong assignments between the parts and their templates.

To avoid such an error, the areas should be carried to a reference and be compared at this reference. The plane of the farthest object can be used as the reference. Also the angle between the surface normals and the rays passing through the center points of the surfaces and their projections must be taken in to account (see Figure 5.6).

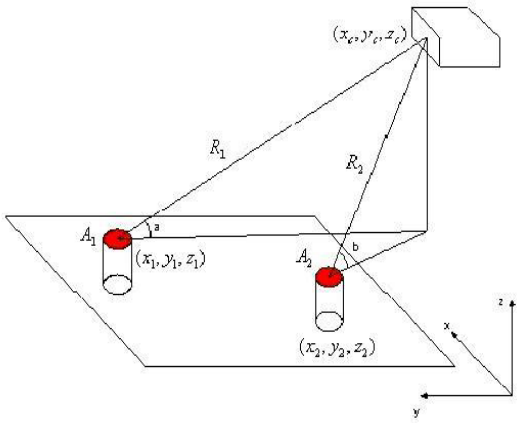


Figure 5.6: Area Correction

The distance between the parts 1 and 2 are denoted as R_1 and R_2, the A_1 and A_2 are the areas of the surfaces and "a" and "b" are the angles between the surface and the image plane respectively. The magnitude of the translation vectors R_1 and R_2 are given as:

$$R_1 = \sqrt{(x_c - x_1)^2 + (y_c - y_1)^2 + (z_c - z_1)^2} \tag{5.25}$$

$$R_2 = \sqrt{(x_c - x_2)^2 + (y_c - y_2)^2 + (z_c - z_2)^2} \tag{5.26}$$

To make the areas perpendicular to image planes, the sines of the angles "a" and "b" are needed. They can be written as:

$$\mathrm{Sin}(a) = \frac{\sqrt{(x_c - x_1)^2 + (y_c - y_1)^2}}{(z_c - z_1)} \tag{5.27}$$

$$\mathrm{Sin}(b) = \frac{\sqrt{(x_c - x_2)^2 + (y_c - y_2)^2}}{(z_c - z_2)} \tag{5.28}$$

If the farthest area is the reference, then the areas must be transformed to:

$$A_1' = A_1 * \sin(a) \tag{5.29}$$

$$A_2' = A_2 * \frac{R_1}{R_2} * \sin(b) \tag{5.30}$$

After assignment of the objects is completed, the system sends the assembly coordinates to the robot (see Fig 5.7).

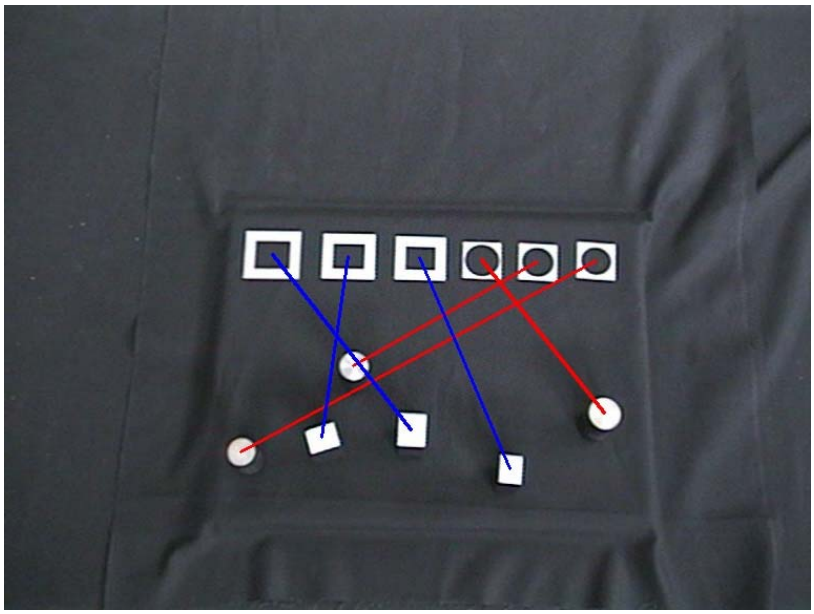


Figure 5.7: Assembly Results

The coordinates sent to the robot are in object coordinate system. The origin of the system is on the table. The system adds offsets to the coordinates that are defined in the calibration process.

Since the axes of the object coordinate system are parallel to the axes of robot coordinates there are no rotation components. The offsets include only translation values.

The system is able to understand the three-dimensional location of an object in any height, during assembly, the robot use the z component of the coordinates by detecting the height of the part and defining the grasping height according to this extracted information. The default grasping heights are 860 mm and 875mm for 65mm and 80mm long cylinders in z direction due to safety of the robot and it is also enough to perform a stable grasp.

In this stage the default heights are 862 mm and 877 mm in z direction. The 2 mm difference between grasping and releasing values is another safety restriction due to the errors in the planarity of the table and to avoid the collusion between part and the template in possible positioning error situation.

The grasping and releasing stages of the prisms are quiet different. The system works in the same procedure until the robots tool tip point reaches 900 mm in z coordinate. At that point robot moves in x-y plane in the robot coordinate system by an offset, which will align the left camera's lens with the prism and capture an image of the prism. This image is used immediately for the extraction of the orientation of the prism. Then this orientation information is fed to the robot. The robot continues the grasping process with the old coordinates and an oriented end-effector. The default grasping height is decreased to 835 mm due to avoid any collisions between the grasped part and the other parts in the environment while the robot positions itself for orientation extraction of the template at 900 mm height.

In the releasing stage the orientation of the templates are extracted with the same procedure. The default releasing height is 837 mm in z coordinate due to the same safety constraint.

CHAPTER 6

ERROR ANALYSIS

In this chapter, the errors and the sources of the errors in the system will be discussed. The main error source is the calibration process. The calibration errors are the most dominant effects in the assembly. Also there are errors caused by the numerical errors and accuracy of the coordinates in the image coordinate extraction process. The combination of these errors can be called as vision system errors.

Vision system errors can be directly calculated by extracting the 3D coordinates of known points. The scenes of the calibration pattern are taken from both cameras (Fig.6.1), which show the same position of calibration pattern. The image coordinates of the corners of the chessboard pattern are fed to the depth extraction algorithm. The coordinates extracted from the depth extraction algorithm, original three-dimensional coordinates and the errors are found as follows:

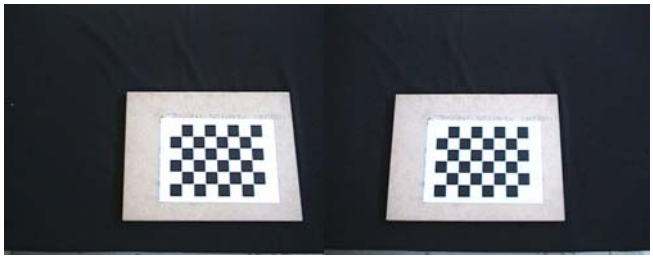


Figure 6.1: The calibration images used in depth extraction.

Table 6.1: The 3D Coordinates of 35 points in chessboard known to the system.

Point No:	X (mm)	Y (mm)	Z (mm)
1	0.00E+00	1.20E+02	0.00E+00
2	3.00E+01	1.20E+02	0.00E+00
3	6.00E+01	1.20E+02	0.00E+00
4	9.00E+01	1.20E+02	0.00E+00
5	1.20E+02	1.20E+02	0.00E+00
6	1.50E+02	1.20E+02	0.00E+00
7	1.80E+02	1.20E+02	0.00E+00
8	0.00E+00	9.00E+01	0.00E+00
9	3.00E+01	9.00E+01	0.00E+00
10	6.00E+01	9.00E+01	0.00E+00
11	9.00E+01	9.00E+01	0.00E+00
12	1.20E+02	9.00E+01	0.00E+00
13	1.50E+02	9.00E+01	0.00E+00
14	1.80E+02	9.00E+01	0.00E+00
15	0.00E+00	6.00E+01	0.00E+00
16	3.00E+01	6.00E+01	0.00E+00
17	6.00E+01	6.00E+01	0.00E+00
18	9.00E+01	6.00E+01	0.00E+00
19	1.20E+02	6.00E+01	0.00E+00
20	1.50E+02	6.00E+01	0.00E+00
21	1.80E+02	6.00E+01	0.00E+00
22	0.00E+00	3.00E+01	0.00E+00
23	3.00E+01	3.00E+01	0.00E+00
24	6.00E+01	3.00E+01	0.00E+00
25	9.00E+01	3.00E+01	0.00E+00
26	1.20E+02	3.00E+01	0.00E+00
27	1.50E+02	3.00E+01	0.00E+00
28	1.80E+02	3.00E+01	0.00E+00
29	0.00E+00	0.00E+00	0.00E+00
30	3.00E+01	0.00E+00	0.00E+00
31	6.00E+01	0.00E+00	0.00E+00
32	9.00E+01	0.00E+00	0.00E+00
33	1.20E+02	0.00E+00	0.00E+00
34	1.50E+02	0.00E+00	0.00E+00
35	1.80E+02	0.00E+00	0.00E+00

Table 6.2: Extracted coordinates of the points in the pattern

Point No:	X (mm)	Y (mm)	Z (mm)
1	-0.083065	119.829336	0.331038
2	30.068682	119.774476	1.477480
3	59.830392	119.489979	1.226306
4	89.862937	120.062704	0.877668
5	119.946986	119.722455	0.688689
6	150.289875	120.678743	-1.501603
7	180.367325	120.255254	-0.791599
8	-0.207134	90.687653	-2.261260
9	29.873758	89.785217	0.034954
10	59.905453	89.955973	0.520547
11	89.887218	89.457209	1.196609
12	119.922877	89.889712	0.519650
13	150.113196	89.760192	0.098940
14	180.185716	90.211299	-0.848230
15	-0.022552	60.138434	-1.067134
16	29.963104	60.215932	-0.209946
17	59.895377	59.265897	2.237329
18	89.945315	59.787956	1.029571
19	119.797076	59.490503	0.899702
20	149.922756	59.677761	0.997546
21	179.957502	59.518849	0.925174
22	0.050071	30.512137	-1.169644
23	30.083530	30.144220	-0.655355
24	60.087818	30.331149	-0.885615
25	89.910344	29.597184	1.444911
26	120.117118	30.250719	-0.499927
27	149.921220	29.857688	0.812227
28	180.326263	30.782220	-2.242518
29	0.080115	0.376357	-2.181633
30	30.077837	0.467923	-1.061879
31	60.046639	0.025646	-0.374343
32	90.008609	0.092834	0.457033
33	119.926245	-0.018860	-0.101807
34	149.979069	0.263189	-0.199479
35	180.065247	0.257582	-1.153713

Table 6.3: The errors in 3D for images in Figure 6.1

Point No:	X (mm)	Y (mm)	Z (mm)
1	0.083065	0.170664	0.331038
2	0.068682	0.225524	1.477480
3	0.169608	0.510021	1.226306
4	0.137063	0.062704	0.877668
5	0.053014	0.277545	0.688689
6	0.289875	0.678743	1.501603
7	0.367325	0.255254	0.791599
8	0.207134	0.687653	2.261260
9	0.126242	0.214783	0.034954
10	0.094547	0.044027	0.520547
11	0.112782	0.542791	1.196609
12	0.077123	0.110288	0.519650
13	0.113196	0.239808	0.098940
14	0.185716	0.211299	0.848230
15	0.022552	0.138434	1.067134
16	0.036896	0.215932	0.209946
17	0.104623	0.734103	2.237329
18	0.054685	0.212044	1.029571
19	0.202924	0.509497	0.899702
20	0.077244	0.322239	0.997546
21	0.042498	0.481151	0.925174
22	0.050071	0.512137	1.169644
23	0.083530	0.144220	0.655355
24	0.087818	0.331149	0.885615
25	0.089656	0.402816	1.444911
26	0.117118	0.250719	0.499927
27	0.078780	0.142312	0.812227
28	0.326263	0.782220	2.242518
29	0.080115	0.376357	2.181633
30	0.077837	0.467923	1.061879
31	0.046639	0.025646	0.374343
32	0.008609	0.092834	0.457033
33	0.073755	0.018860	0.101807
34	0.020931	0.263189	0.199479
35	0.065247	0.257582	1.153713

These errors are extracted using calibration points. For an ordinary point the errors will be larger, so it can be stated that the errors in the above table are the minimum errors.

The repeatability of the robot is ± 0.1 mm, and the displacement of the right finger of the gripper is larger than the left one due manufacturing errors. This error varies with the pressure of the air fed in to the gripper and the dimension of the parts. This creates an offset from the center of the gripper that directly affects the assembly process.

In the orientation extraction process, the robot positions itself according to the extracted coordinates of the objects, in order to get an image of the prism or the template, which causes error in orientation extraction. The orientation error can't be calculated directly due to the unstable errors caused by calibration but it can be estimated by comparing the real orientation of the objects to orientations extracted by the vision system. For this purpose 18 real orientation values of the objects are measured and compared to the orientation values, which are extracted by the vision system. The results can be seen in Table 6.4:

Table 6.4: Orientation Errors

	Measured Orientation (degrees)	Extracted Orientation (degrees)	Error (degrees)
1	90	87.157193	2.84807
2	110	106.252595	3.747405
3	70	69.249336	0.740664
4	60	56.580771	3.419229
5	90	87.959992	2.040008
6	120	116.459684	3.540316
7	50	48.427586	1.572414
8	89	86.769914	2.230026
9	115	113.354699	1.645301
10	135	131.335905	3.664095
11	28	28.289723	-0.289723
12	89	95.620500	-6.6205
13	179	174.627207	4.37274
14	60	58.191635	1.808635
15	121	118.812260	2.18774
16	55	51.611906	3.388087
17	124	129.367412	-5.397412
18	37	33.386768	3.613232

The angle errors are not systematic. As seen in table above, for the same angle the system can extract different angles due to positioning errors created by the vision system. Also during orientation of the gripper with the angle provided by the orientation extraction algorithm about x axis of the gripper, the gripper rotates not about the center of the tip point of the gripper, but about the original tip point attached to 6'th joint (see Fig.6.2).

This creates an offset between the real position of the object's top surface centroid and the oriented gripper center. To align the oriented gripper to the object a correction is required. This correction involves positioning and orientation. This means that the correction brings an error to the grasping or releasing coordinates due to positioning and orientation errors. The most erroneous assemblies occurred in prisms due to the usage of these corrections.

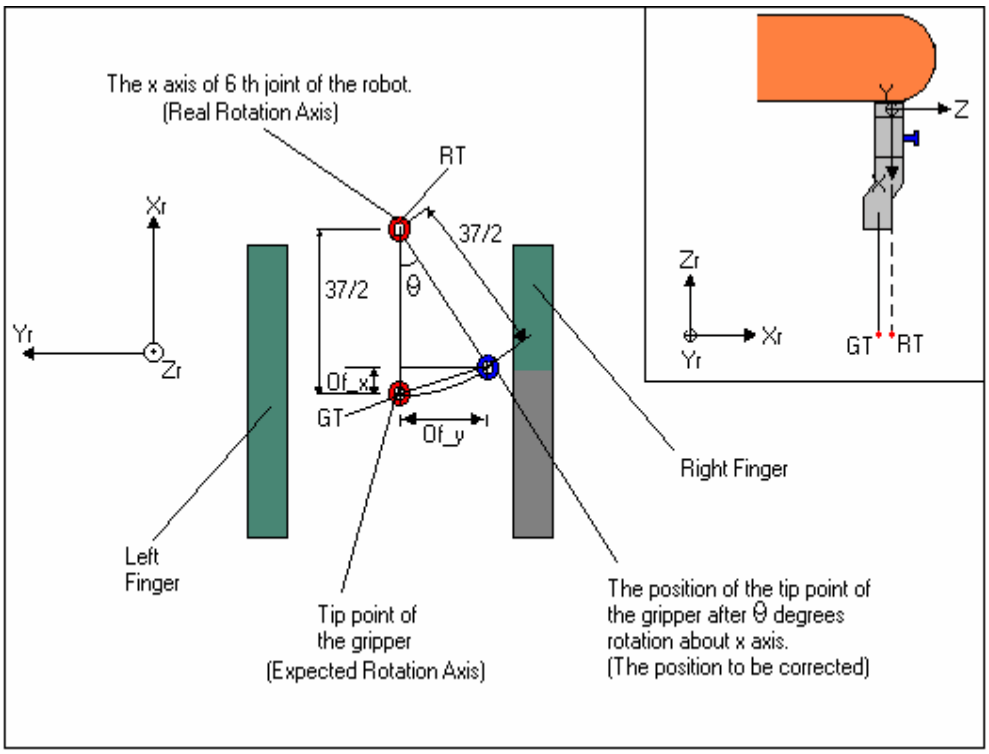


Figure 6.2: Offset Correction

The offsets used to correct this positioning error can be written as:

$$Of\_x = (37/2) * (1 - \cos(\theta)) \quad (6.1)$$

$$Of\_y = (37/2) * \sin(\theta) \quad (6.2)$$

The flowcharts of the operation for cylinders and prisms are shown in Figure 6.3 and 6.4. In these flowcharts the effects of errors on the three-dimensional coordinates after each operation are shown.

The notations RE, GE, RLE, OE represent robot positioning error, grasping error, releasing error and orientation error.

The vision system error varies between 0.020931 mm and 0.169608 mm in X, 0.018860 mm and 0.734103 mm in Y, 0.034954 mm and 2.261260 mm in Z.

The orientation error (OE) varies between -6.620 degrees and 4.37274 degrees.

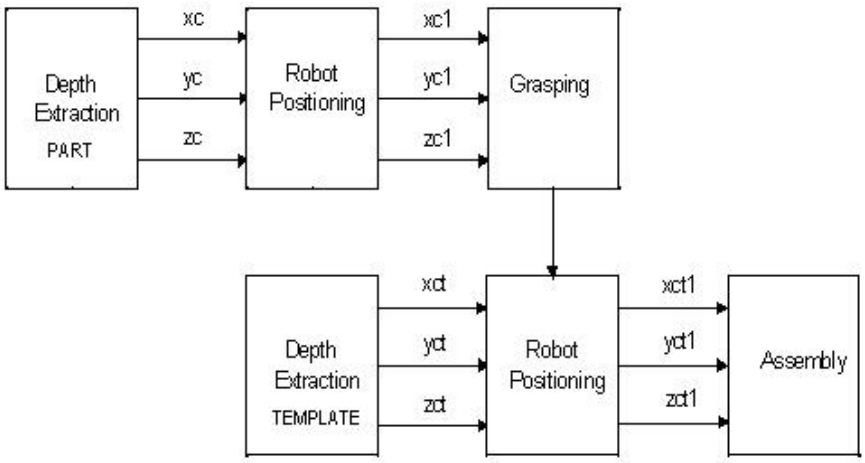


Figure 6.3: Coordinate flowchart for cylinders

Where;

xc = Extracted x coordinate of the cylinder by the vision system including the vision system error.

yc = Extracted y coordinate of the cylinder by the vision system including the vision system error

zc = Extracted z coordinate of the cylinder by the vision system including the vision system error

xct = Extracted x coordinate of the cylinder template by the vision system including the vision system error

yct = Extracted y coordinate of the cylinder template by the vision system including the vision system error

zct = Extracted z coordinate of the cylinder template by the vision system including the vision system error

Then;

$$xc1 = xc \pm RE \tag{6.3}$$

$$yc1 = yc \pm RE \tag{6.4}$$

$$zc1 = zc \pm RE \tag{6.5}$$

$$xct1 = xct \pm RE \tag{6.6}$$

$$yct1 = yct \pm RE \pm GE \tag{6.7}$$

$$zct1 = zct \pm RE \tag{6.8}$$

After assembly is done, the final coordinates of the cylinder have also a releasing error:

$$xac = xct1 \pm RLE \tag{6.9}$$

$$yac = yct1 \pm RLE \tag{6.10}$$

$$zac = zct1 \tag{6.11}$$

For prismatic parts the error model is rather different due to orient tation errors.

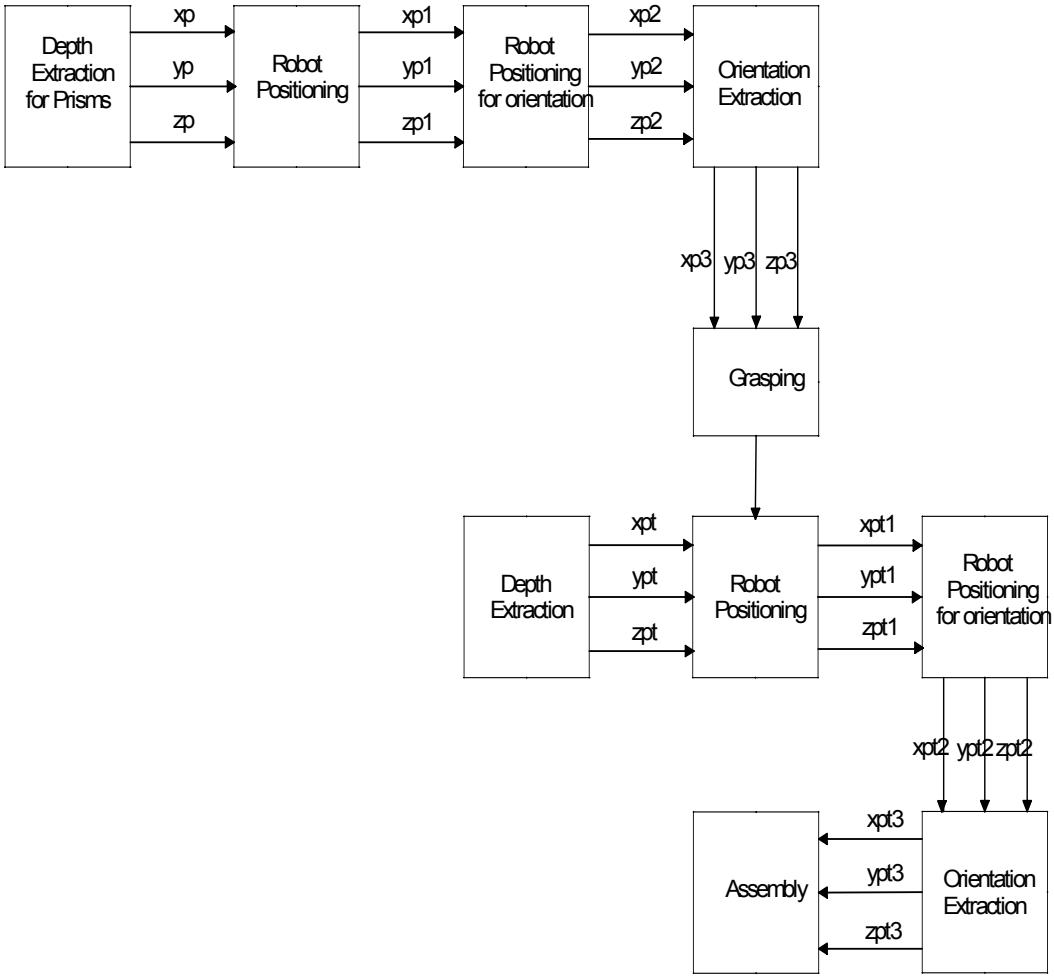


Figure 6.4: Coordinate flowchart for prisms

Where;

xp = Extracted x coordinate of the prism by the vision system including the vision system error

yp = Extracted y coordinate of the prism by the vision system including the vision system error

zp = Extracted z coordinate of the prism by the vision system including the vision system error

xpt = Extracted x coordinate of the prism template by the vision system including the vision system error

ypt = Extracted y coordinate of the prism template by the vision system including the vision system error

zpt = Extracted z coordinate of the prism template by the vision system including the vision system error

Then;

$$xp1 = xp \pm RE \tag{6.12}$$

$$yp1 = yp \pm RE \tag{6.13}$$

$$zp1 = zp \pm RE \tag{6.14}$$

$$xp2 = xp1 \pm RE \tag{6.15}$$

$$yp2 = yp1 \pm RE \tag{6.16}$$

$$zp2 = zp1 \pm RE \tag{6.17}$$

$$xp3 = xp - \left(\frac{37}{2}(1 - \cos(\theta \pm OE)) \right) \pm RE \tag{6.18}$$

$$yp3 = yp - \left(\frac{37}{2}\sin(\theta \pm OE) \right) \pm RE \tag{6.19}$$

$$zp3 = zp \pm RE \tag{6.20}$$

$$xpt1 = xpt \pm RE \tag{6.21}$$

$$ypt1 = ypt \pm RE \tag{6.22}$$

$$zpt1 = zpt \pm RE \tag{6.23}$$

$$xpt2 = xpt1 \pm RE \tag{6.24}$$

$$ypt2 = ypt1 \pm RE \tag{6.25}$$

$$zpt2 = zpt1 \pm RE \tag{6.26}$$

$$xpt3 = xpt - \left(\frac{37}{2}(1 - \cos(\theta \pm OE)) \right) \pm (GE * \sin(\theta \pm OE)) \pm RE \tag{6.27}$$

$$ypt3 = ypt - \left(\frac{37}{2}\sin(\theta \pm OE) \right) \pm (GE * \cos(\theta \pm OE)) \pm RE \tag{6.28}$$

$$zpt3 = zpt \pm RE \tag{6.29}$$

After Assembly is done, the final coordinates of the prism have also a releasing error:

$$xap = xpt1 \pm RLE \tag{6.30}$$

$$yap = ypt1 \pm RLE \tag{6.31}$$

$$zap = zpt1 \tag{6.32}$$

During assembly process of the application the prismatic parts have greater errors compared to cylindrical parts. This model proves that the assembly error is greater in assembly of the prism parts. After a sample assembly, the following errors were obtained by measuring the offsets between the parts' boundaries and templates' inner boundaries via vernnier caliber. Table 6.5 shows the errors in centroid positioning extracted from offset measurements.

Table 6.5: Assembly Errors

Parts	X (mm)	Y (mm)
Cylinder1	2.3	1.8
Cylinder2	2.56	1.9
Cylinder3	2.56	2
Prism 1	2.7	2.5
Prism 2	2.4	2.1
Prism 3	2.2	1.3

CHAPTER 7

DISCUSSION AND CONCLUSION

7.1 Summary

The objective of the thesis was to design a system, which involves a vision system combined with an industrial robot to recognize, understand 3D position and grasp some geometric primitives, specified as cylinders and prisms. It is aimed that the system should have a graphical user interface, which would allow the user to observe the process and interfere if needed. The designed system has reached these objectives successfully.

The system has four main logical modules as described in Chapters 2, 3, 4 and 5. In the first module called “Calibration Module”, both cameras are calibrated using a modified version of Zhang’s Method. Intel Open Computer Library (Open CV) is used for calibrating the cameras. Although this method is very flexible, after its implementation, it is observed that the calibration data extraction process is very sensitive to the lightning conditions. An A4 size chessboard pattern printed with a laser printer is used as the calibration pattern. The pattern is attached to an A3 size wooden block to have a stable and rigid calibration board. The calibration results are satisfactory enough for the system to achieve the expected goals.

In the second module called “Imaging and Preprocessing Module”, the images are first captured from both cameras and then converted to gray scale images. The distortions in the images are corrected using the distortion parameters provided by calibration. The distortion correction is followed by image segmentation.

Tresholding method is used for segmentation. Automatic tresholding methods are ignored due to unconstrained lightning environment. It is observed that a traditional tresholding method that allows user to define the treshold value during the process is more robust. This approach ensures that the image is perfectly segmented to objects and background under the supervision of the user.

In the third module called “Object Recognition Module”, objects and their geometrical features are extracted. Utilizing these extracted features the recognition of the parts is done. It is obvious that in such a system, which uses geometric primitives, the most powerful information about the parts is their geometry of the cross-sections. For this purpose the upper surfaces of the objects are considered. It is observed that due to uncontrolled lightning conditions, there is a reflection problem in the side surfaces of the parts to be grasped. Due to some environmental problems, this undesired lightning condition couldn’t be overcome. To solve this problem the side surfaces of the parts are covered with black.

In the fourth module called “Depth Extraction and Assembly Module”, the 3D robot coordinates of the objects are extracted, and then according to sizes and the object classes provided by recognition, the objects are assigned to templates. The positions of the objects are sent to the robot and the assembly is done.

It is observed that there were some offsets in expected coordinates of grasping and releasing in the object space. These errors are caused by vision system errors, robot and gripper.

7.2 Future Work

The system can be developed to be less sensitive to lightning conditions. Different approaches may be applied to overcome the reflection problem in the side surfaces of the parts to be grasped. Taking images form different positions and extracting the top surfaces from these images can be a solution. The camera holder has 380 mm length in the positive x axis from the robot's tip. This narrows the working range of the robot drastically. In order to apply this approach first the narrowing effect of the camera holder in the robot working range, the limits of the common field of view of the cameras, and the occlusion detection should be considered.

The system may extract the 3D coordinates from any position of the robot. In this approach, the narrowing effect of the camera holder on the robot working range plays a critical role; again the limits of the common field of view of the cameras must be sufficient enough to work on this issue. Taking images from random positions of the robot will result in variation of the scenes of the parts in which the top surfaces of the parts are missing or parts are occluding each other. This means that the recognition strategy should be changed completely and occlusion detection may be required.

The error analysis can be improved by using proximity sensor. The real coordinates of the part can be measured using a laser proximity sensor. This will increase the accuracy of the measurements. Also the system may be converted to an image-based real-time robot control system.

REFERENCES

[1] Janne Heikkila & Olli Silven (1997). A Four-step Calibration Procedure with Implicit Correction. Infotech Oulu and Department of Electrical Engineering University of Oulu. http://www.vision.caltech.edu/bouguetj/calib_doc/papers/heikkila97.pdf

[2] Richard Szelinski & Ramin Zabih (2000). An Experimental Comparison of Stereo Algorithms. http://www.research.microsoft.com/ szelinski / stereo

[3] Gregory D.Hager (1997). A Modular System for Robust Positioning Using Feedback from Stereo Vision. IEEE Transactions on Robotics and Automation, Vol.13, No. 4 pp 582-595 , August 1997.

[4] Gabriel Fielding & Moshe Kam (1999). Applying the Hungarian Method to Stereo Matching. Proc. Of the 36th IEEE CDC San Diego, Ca. Dec 1999.

[5] F. Janabi-Sharifi & W.J. Wilson (1996). Automatic Selection of Image Features for Visual Servoing.IEEE Transactions on Robotics and Automation , Vol 13, No 6 , pp 890-903, December(1997).

[6] Roger Y.Tsai (1985). A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf Cameras and Lenses. IEEE Journal of Robotıcs and Automation, Vol.RA-3, No. 4, pp 323-344 August 1987.

[7] Hong Suh & Tae Won Kim (1999). A visual servoing algorithm using fuzzy logics and fuzzy-neural networks. Mechatronics 10 (2000) 1-18.

[8] Christopher E.Smith, Scott A. Brandt & Nikolaos P.Papanikolopoulos (1996). Eye-In-Hand RoboticTasks In Uncalibrated Environments. IEEE Transactions on Robotics and Automation, Vol. 13 , No. 6 pp 903-914. December 1997

[9] Zhengyou Zhang (1998). A Flexible New Technique for camera Calibration.Technical Report MSR-TR-98-71,Microsoft Research, December 1998, http:// research.microsoft.com/zhang/Calib.

[10] Rahul Singh,Richard M.Voyles David Littau Nikolaos P. Papanikolo poulos(1999). Grasping Real Objects Using Virtual Images. htpp://citeseer.nj.nec.com

[11] Peter T.Sander, Laurent Vinet, Laurent Cohen & Andre Gagalowicz (1989). Hierarchical Region Based Stereo Matching. htpp://citeseer.nj.nec.com

[12] Ramesh Jain, Rangachar Katsuri, Brian G.Schunk (1995). Machine Vision. McGraw-Hıll International Editions Computer Science Series.

[13] Rajeev Sharma & Seth Hutchinson (1996).Motion Perceptibility and its Application to Active Vision-Based Servo Control. IEEE Transac tions on Robotics and Automation Vol 13, No.4, pp 607-617. August1997

[14] Efthimia Kefelea(1998).Object Localization and Recognition for a Grasping Robot.htpp://citeseer.nj.nec.com

[15] Richard O. Duda, Peter E.Hart, David G.Stock (2001). Pattern Classification Second Edition, John Wiley &Sons.

[16] Gordon Wells, Chistophe Venaille, Carme Torras (1996). Promising research Vision-based robot positioning using neural networks.Image and Vision Computing 14 (1996) pp 715-732

[17] Jose Mauricio S.T. Motta,Guilherme C. de Carvalho, R.S McMaster (2001). Robot calibration using 3D vision-based measurement system with a single camera.Robotics and Computer Integrated Manufactu ring 17 (2001) pp 487-497

[18] R.A Lane & N.A.Thacker (1996).Stereo Vision Research: An Algorithm Survey. htpp://citeseer.nj.nec.com

[19] Danica Kragic & Henrik I Christensen(2002). Survey on Visual Servoing for Manipulation.Report from Computational Vision and Active Perception Labratory Stockholms Unıversıtet. http://www.nada.kth.se/~danik/publications.html

[20] Reimar K. Lenz & Roger. Y. Tsai (1987).Techniques for Calibration of the Scale Factor and Image Center forHigh Accuracy 3D Machine Vision Metrology. IEEE Transactions on Pattern Analysis And Machine Intelligence, Vol. 10 No. 5 pp 713-720,September 1988.

[21] Christopher E. Smith & Nikolaos P.Papanikolopoulos(1995).Theory and experiments in Vision-Based Grasping. 34th IEEE Conference on decision and Control (1995).

[22] U.Büker,S.Drüe, N. Götze,G. Hartmann, B. Kalkreuter, R. Stemmer, R. Trapp (2001).Vision-Based control of an autonomous dissasembly station. Robotics and Autonomous Systems 35 (2001) pp 179-189.

[23] Josef Pauli,Arne Schmidt,Gerald Sommer(2001).Vision-based integrated system for object inspection and handling.Robotics and Autonomous Systems 37 (2001) pp 297-309.

[24] Billibon H. Yoshimi and Peter K. Allen (1994). Visual Control of Grasping and Manipulation Tasks. Center for Research in Intelligent Systems, Department of Computer Science. htpp://citeseer.nj.nec.com, ftp.cs.columbia.edu/pub/vision/iuw-cdrom/yoshimi/iuw94.ps

[25] Peter I. Corke (1997).Visual Control of Robots:high-performance visual servoing.John-Wiley and Sons INC.

[26] Peter I. Corke (1997).Visual Control of Robot Manipulators-A Review. htpp://citeseer.nj.nec.com

[27] Roberto Cipolla & Nick Hollinghurst (1997). Visually Guided Grasping in Unstructured Environment. htpp://citeseer.nj.nec.com

[28] Radu Horaud, Fadi Dornaika & Bernard Espinau (1998). Visually Guided Object Grasping. IEEE Transactions on Robotics and Automation, Vol 14, No 4 pp 525-532, August 1998.

[29] Christopher E. Smith & Nikolaos P.Papanikolopoulos (1996). Vision-Guided Robotic Grasping : Issues and Experiments. htpp://citeseer.nj.nec.com

[30] Brad Nelson, N.P. Papanikolopoulos & P.K. Khosla (1993).Visual Servoing For Robotic Assembly.In Visual Servoing-Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback,ed. K.Hashimoto, World Scientific Publishing , pp 139-164.

APPENDİX A

SONY/AXIS EVI-D31 CAMERAS

Figure A.1: Sony EVI-D31 Camera

EVI-D30/D31 is a pan/tilt video camera with highly sophisticated image processing technology, which enables target objects to be recognized.

Features:

High speed, Wide Range Pan/tilter
X12 Optical Zoom, High Speed Auto-Focus Lens
6 Position Preset
Auto Tracking / Motion Detector
RS-232C Serial Control (VISCATM)
IR remote Commander

AT(Auto Tracing)Mode

AT is a function, which continually extracts a subject that the user pre-defines. After picking up pixels of similar color and brightness around the selected subject, EVI-D30/D31 extracts the target by using the subject model based on light reflection and nonlinear camera processing. There are four modes for pre-defining the subject. AT-PAN/TILT function follows the moving subject automatically by controlling the pan & tilt motors without the use of special sensors. AUTO ZOOM function automatically controls the zoom lens to ensure that the size of the subject remains constant. The EVI-D30/D31 employs the auto exposure and advanced backlight compensation systems to ensure that the subject remains bright even in harsh backlight conditions. Because the subject position is known a comparison can be made between its brightness and that of the background and the camera subsequently adjusted to compensate for the conditions.

MD(Motion Detector)Mode

MD basically detects the difference between the initial reference image and the current image. The conventional technique employed in MD uses only the brightness of the video signal. The EVI-D30/D31 uses both the brightness and color, which enables even an object of the same brightness

as the background to detected. Changes in light conditions are constantly monitored and the data in the reference image adjusted.

A user can set two rectangular detection areas of any size and in any position of the scene. Once motion is detected within the preselected windows an alarm signal is output, which, for example, can commence the recoding on a VCR. This mode is made even more versatile by the ability to adjust the detection brightness, color and area.

VISCA

EVI-D30/D31 can be controlled by RS-232C serial control using VISCA™. VISCA™ is an acronyrn of Video System Control Architecture. It is a network protocol designed to interface a wide variety of video equipment to computer. Under VISCA™, up to 7 EVI-D30/D31 can be connected to one controller using RS-232C communication. RS-232C parameters are communications speed of 9600 baud, data length of 8 bits, 1 stop bit, and no parity.

Pan/Tilt Range

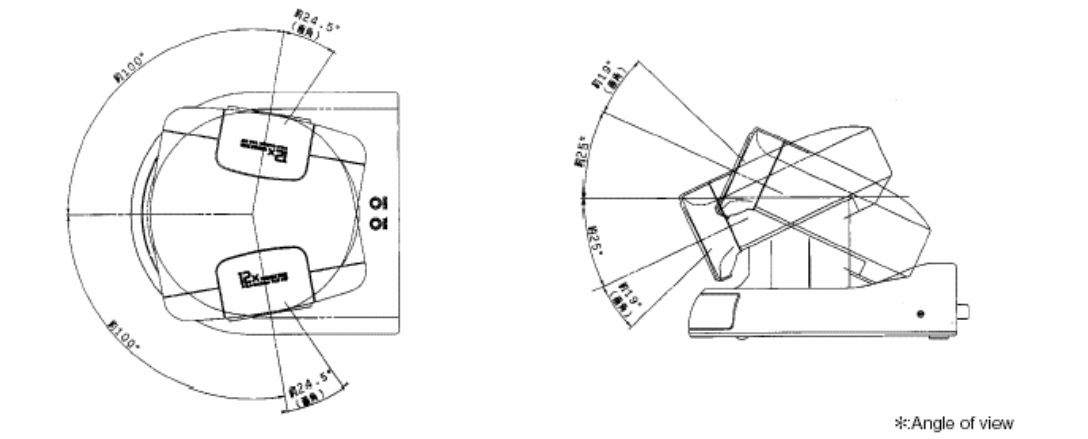


Figure A.2: Pan/Tilt Range

Specifications

System

Video signal:	EVI-D30: NTSC Color, EIAJ standards
	EVI-D31: PAL Color, CCIR standards
Picture element:	1/3 inch color CCD
	Total picture element number:
	EVI-D30: Approx. 410,000
	EVI-D31: Approx. 470,000)
Effective picture element number:	
	EVI-D30: Approx. 380,000
	EVI-D31: Approx. 440,000
Lens :	Electromotion twelve fold zoom lens
	f=5.4 to 64.8mm, F1.8 to F2.7
	Horizontal angle: 4.4° to 48.8°
Point-blank range:	WIDE end : 10mm
	TELE end : 800mm
Minimum illumination:	7 lux (F1.8)/with 50IRE
	Illumination range: 7 to 100,000 lux
Shutter speed:	EVI-D30: 1/60 to 1/10,000 (VISCA control)
	EVI-D31: 1/50 to 1/10,000 (VISCA control)
Gain selector :	Automatic/manual
Horizontal resolution:	NTSC : 460 TV
	PAL : 450 TV
Video S/N:	48 dB
Pan/tilt action Horizontal:	100°, Vertical: 25°

Input/output terminals

Video output:	RCA pin jack (1), 1Vpp, 75 ohm unbalanced
	Synchronization: negative
S video output:	4 pin mini DIN (1)

Audio output:	RCA pin jack, monaural (1)
	Rated output: 327 mV
	Output impedance: less than 2.2 kilohms
Input/output	
control terminals:	RS232C (input: 1, output: 1), 8 pin mini DIN,9600bps
	Data: 8 bit
	Stop bit: 1
Microphone input terminal:	Mini jack (monaural) (1) (ø 3.5)
	Rated input 0.775 mV
	DC 3V for low impedance microphone
	Input impedance: more than 10 kilohms
Power terminal :	EIAJ type4

General

Input voltage:	DC 12 to 14 V
Power consumption:	11 W
Operating temperature:	0° to 40° (32° to 104°F)
Storage temperature:	– 20° to 60° (– 4° to 140°F)
Dimensions Video camera:	Approx 142 × 109 ×164 mm
	(5 5/8 × 4 3/8 × 6 1/2 in.)
	(w/h/d)
Remote commander:	Approx. 56 × 26 × 210 mm
	(2 1/4 × 1 1/16 × 3/8 in.)
	(w/h/d)
Mass Video camera:	Approx. 1,200 g
	(42.3 oz.)
Remote commander:	Approx. 109 g (3.8 oz)

APPENDİX B

DT3133 FRAME GRABBER CARD

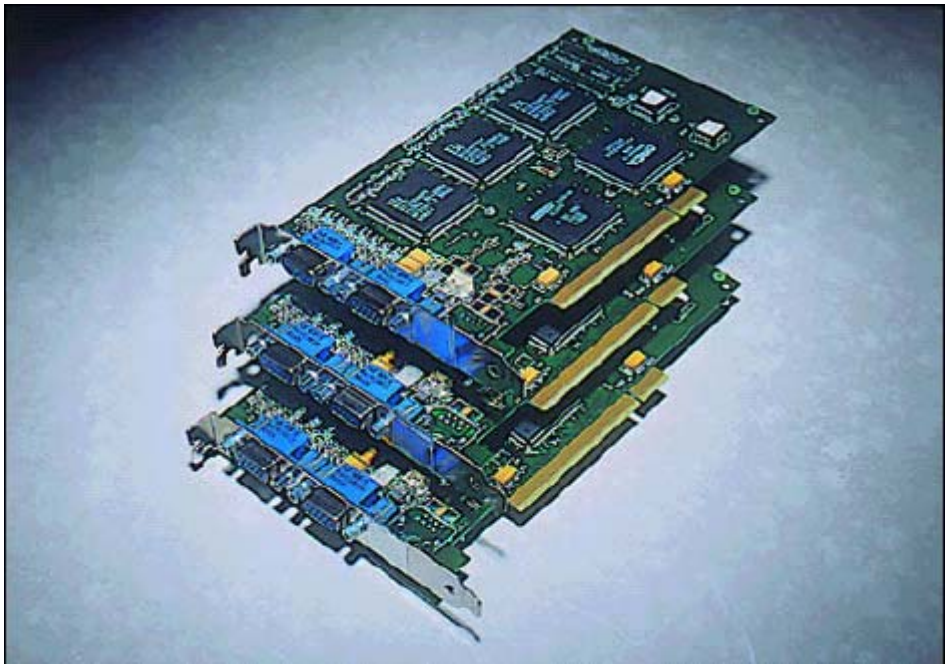


Figure B.1: DT3133/32/31 Frame Grabber Cards

Ideal for applications requiring simultaneous image acquisition from multiple sources, DT3130 Series frame grabbers contain the functionality of up to three individual frame grabbers, all on one half-size PCI board. An option adds isolation from the hazards of an industrial environment.

The DT3133 includes three active inputs or nine muxed, for input of up to nine RS-170/CCIR monochrome, NTSC/PAL color cameras, or three Svideo and six RS-170/CCIR, NTSC/PAL cameras.

Key Features

- Contains the functionality of up to three frame grabbers on one PCI short card, enabling multiple image acquisition.
- Handles monochrome, composite color, and S-video input sources.
- Available option adds isolation from the hazards of an industrial environment.
- Programmable strobe outputs for precise camera control.
- 12 volt camera power connection.
- PCI Bus Master and Scatter/Gather architecture for intelligent image data management; enables acquisition and transfer to host memory at 30 fps (RS-170/NTSC), 25 fps (CCIR/PAL).

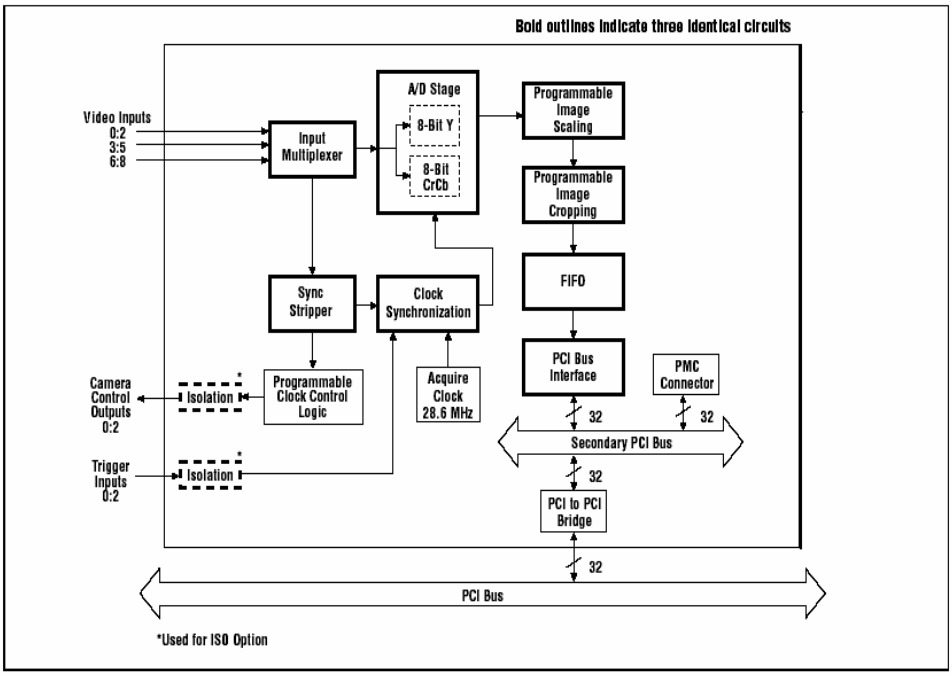


Figure B.2:Block Diagram of DT3133 framegrabber card

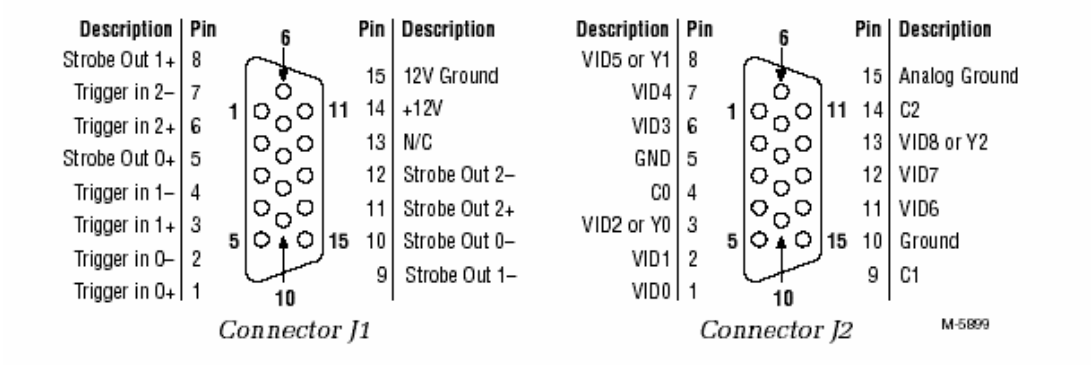


Figure B.3: User Connection Of DT3133

Video Input

Video Format*:* Composite video and S-video (Y/C) formats; RS-170, RS-330, and NTSC (60 Hz) or CCIR and PAL (50 Hz); interlaced; software selectable

Timing Format*:* Standard 60 Hz and 50 Hz timing formats are supported; software selectable

Inputs: DT3133: 3 simultaneously active inputs at any one time; 3 active composite out of 9 multiplexed composite, or 3 S-video and 6 composite; all inputs ac coupled

Video Signal: 1 volt peak to peak, 75 ohms

Spatial Resolution*:* 640 x 480 (60 Hz); 768 x 576 (50 Hz)

Acquisition

Digitization: Twin 8-bit A/Ds, one for monochrome, and one for chroma; data derived to YCrCb format.

Pixel Jitter*:* ±6 nsec maximum

Aspect Ratio*:* 1:1 Square pixels, depending on scaling factors

Frame Grab Speed*:* 1/30 s (60 Hz), 1/25 s (50 Hz)

Modes: Interlaced (start on next even, next odd, or next field), single frame or continuous operation; all software selectable.

On-Board Processing

Region Of Interest*:* Programmable ROI window defines video data to be transferred to memory; pixels outside window are discarded.

Scaling: Images scaleable to 4 pixels by 4 lines, Performed using linear phase interpolation; software selectable

Data Formats

Image data can be output in 32, 24, 16, and 15-bit RGB, 16-bit YUV, or 8-bit monochrome formats

Control Signals

External trigger inputs*:* DT3133: 3 total
TTL levels—one per active video input

Camera strobe outputs*:* DT3133: 3 total—one per active video input; individually controllable;TTL levels; Programmable HSYNC and VSYNC counts; Strobe output pulse-width programmable from 3.3 to 427 msec with selectable polarity.
Control Signal Isolation: Available via the ISO option.

Video Display

Uses PC's graphics card and monitor for display. Real-time video display and non-destructive, real-time animated overlays performed using DirectDraw (DDI)

Video Transfer Rate

55 MB/s typical, 132 MB/s max. Board operates as a Bus Master using Burst Mode for data transfer to host memory. Intelligent Scatter/Gather architecture used for image data management in host memory.

Power Requirements

+5 V @ 1 A typical

+12 V @ 1.5 A max (for camera power) via CPU power supply harness

Physical and Environmental

Form:	Half-size PCI bus board (short card)
Dimensions:	10.7 cm x 17.5 cm (4.2 in. x 6.875 in.)
Weight*:*	150 g (5.3 ounces)
Operating Temperature*:*	0° to 50° C(32° to 122° F)
Storage Temperature*:*	–25° to 70° C(–13° to 158° F)
Relative Humidity*:*	Up to 90%, non-condensing

System Requirements

Pentium-III class processor

32-bit/33MHz PCI bus and supporting BIOS

At least one available PCI Bus slot

Microsoft Windows 2000/XP

256 MB of system RAM minimum

CD-ROM drive (for software installation)

Graphics controller with DirectX driver

APPENDIX C

PROGRAM STRUCTURE

The designed program was written in MS VC++ 6.0 and contains approximately 9000 lines of code all written by the author except The Intel OpenCV Library functions and Robot Control Library.

The Robot Control Library for ABB IRB 2000 industrial robot was designed by Ali Osman Boyacı and Anas Abidi in MS Visual Basic 6.0, converted and integrated to VC++ by the author.

The Intel Open Computer Vision Library is a free library that includes ready functions for many computer vision operations, and it can be downloaded from internet.

In the succeeding pages the operational structure of the program will be illustrated by giving screen-shots from the program.

Figure C.1: A Screen Shot of the Program

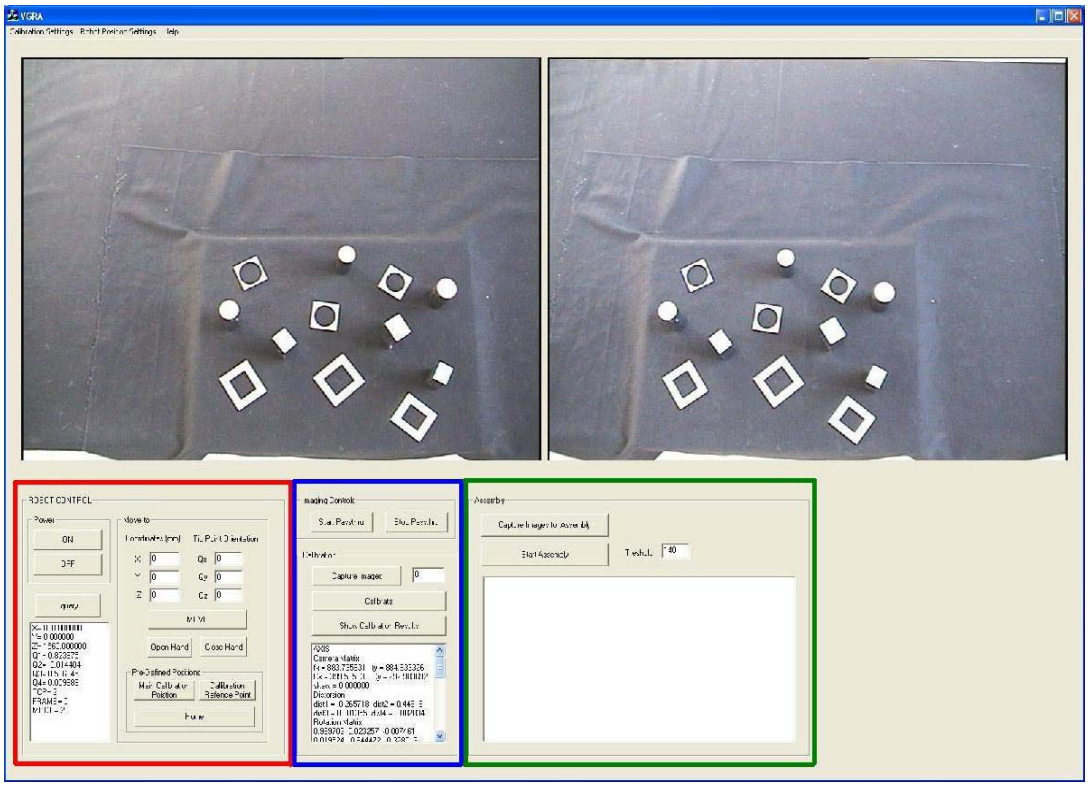


Figure C.2: Partitions for Different Operations Done by the Program

As it can be seen from Fig. C.2, the different operations done by the program are categorized as Robot Control, Imaging Controls, Calibration and Assembly. The buttons, which belongs to same category are grouped together and bounded with colored boxes in the above figure. Red box indicates the Robot Control category, blue box indicates the Imaging Controls and Calibration Controls, and green box indicates the Assembly Controls.

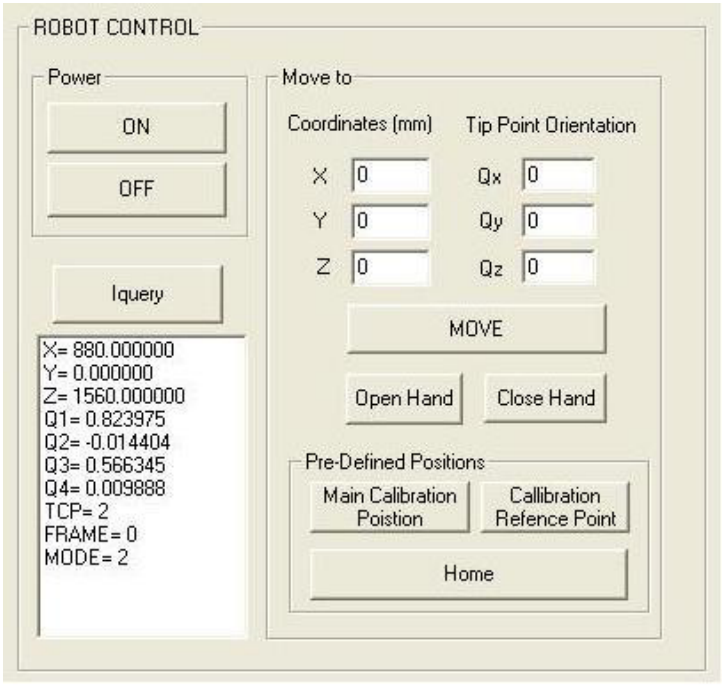


Figure C.3: Robot Control Buttons
(The Red Box)

In Figure C.3 the robot control buttons can be seen, The “ON” button opens the RS232 port and sends a message in order to turn on the robot. The “OFF “ button turns of the robot and closes the port. The “Inquiry” button returns the pose of the end effector, the TCP used, the frame used and the operation mode in to a list box below this control button (see Fig. C.3). The “Move” button is used with the coordinate and orientation inputs, which are entered to text boxes above this button. The program will give an error message if nothing is entered to the input text boxes or if “Z” coordinate exceeds the safety limit 830. The ”Open Hand” and “Close Hand” buttons are designed to control the gripper’s actions. The “Home” button sends the robot to the home position (X = 950, Y = 0, Z = 1585, θ_z = 0, θ_y = 0, θ_x = 0) of the robot. For “Main Calibration Position” button and ”Calibration Reference Position” see Appendix D.

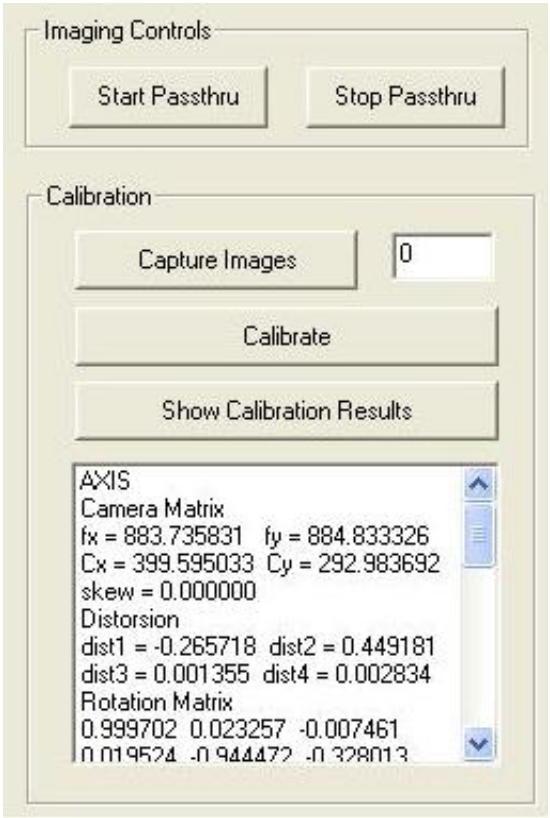


Figure C.4: Imaging and Calibration Control Buttons
(The Blue Box)

The ”Imaging Control” category involves “Start Passthru” and “Stop Passthru” buttons. These buttons are designed to start and stop the continuous display respectively. For calibration control buttons see Appendix D.

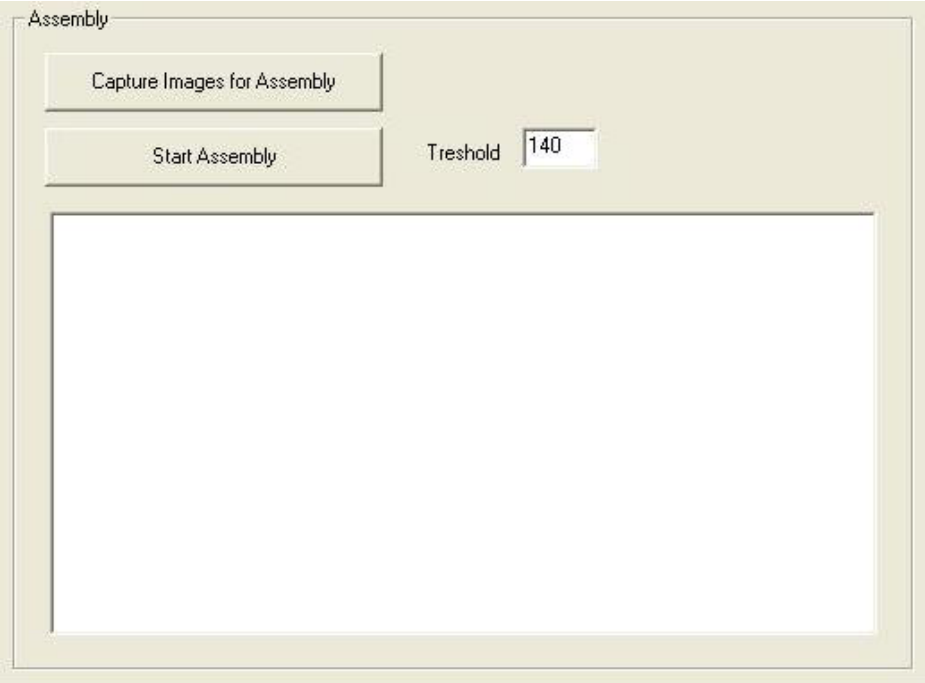


Figure C.5: Assembly Control Buttons
(The Green Box)

The Assembly category involves “Capture Image for Assembly” and ”Start Assembly”. “Capture Image for Assembly” button is designed to capture main assembly images. This function is separated from main assembly function to allow user to check the images. ”Start Assembly” button is designed to execute the main assembly operation. This function needs a treshold input, which is supplied from a text box near the button by the user. The code under the ”Start Assembly” button, involves the whole logical structure, which were explained in Chapters 3, 4 and 5. The list box under this button is designed to list the coordinates of the objects.

In order to run the system, the following steps should be taken:

1. Place the parts on the board which is placed on the table.
2. Press “ON” button in the robot control frame to turn on the robot.
3. Press “Start Passthru” button in the Imaging Control Frame to check the cameras.
4. Press “Main Calibration Position” button in the “Pre-defined Positions” frame.
5. Press “Capture Images for Assembly” button. The system will show the captured images sequentially. Use “Return” in the keyboard to close the images and to see the next image.
6. Be sure that the captured images are not erroneous.
7. Enter a value between 0 and 255 to treshold input text box. This value can be changed until the desired segmentation results are obtained.
8. Press “Start Assembly” button to start automatic assembly process. The system will show the preprocessing results, recognition results and extracted assembly strategy visually in order to allow user to interfere if needed.
9. At the end of the process the system will ask to start assembly. If the user presses “NO”, the robot will be back to its home position. Else the robot will start moving. After having done the assembly, the robot will be back to its home position.

APPENDIX D

CAMERA CALIBRATION PROCEDURE

To calibrate the system the following steps should be followed :

Step1 :

Put the gauge and set it to one of the farthest corners as shown in Figure D.1.

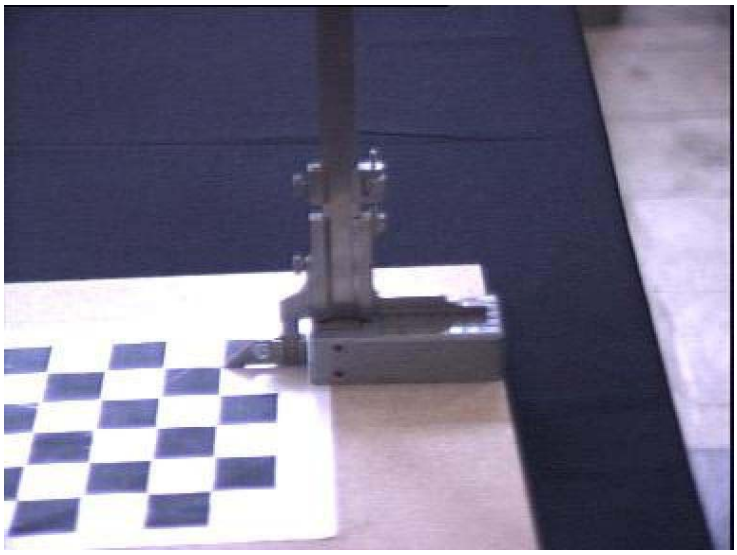


Figure D.1: Positioning The Gauge

Step 2 : Set the gauge to 280 mm as shown in Figure D.2.

Figure D.2: Setting The Height of The Gauge

Step 3 :

Press "Calibration Reference Point" button (see Figure D.3) to move the robot to the pre-defined position (1140 mm , 0 mm ,1074 mm) (see Figure D.4) . This coordinates will be used as 3D offset for extracting model coordinates of the corners.

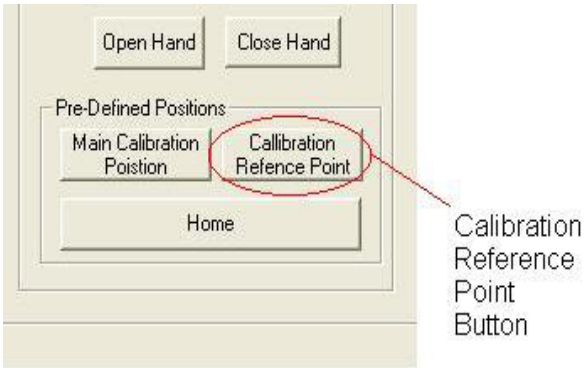


Figure D.3: Calibration Reference Point Button

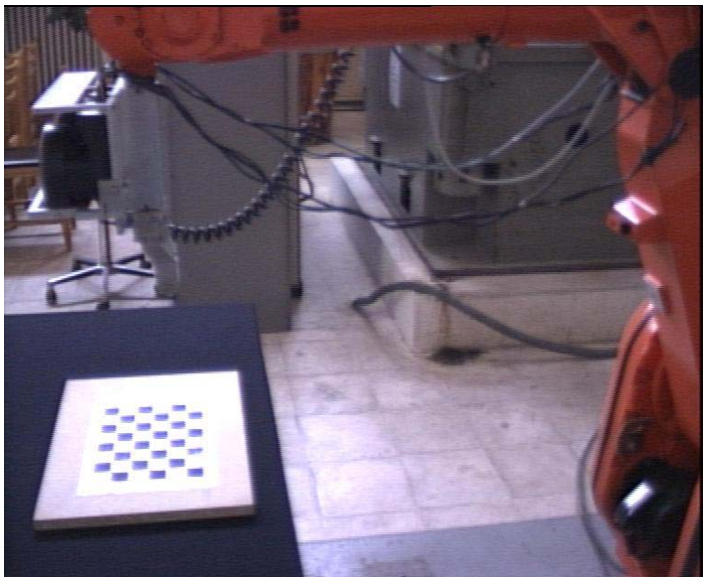


Figure D.4: Calibration Reference Point

Step 4 :

Move the pattern with the gauge very carefully to align the gauge to the mark on the end effector of the robot (see Figure D.5).

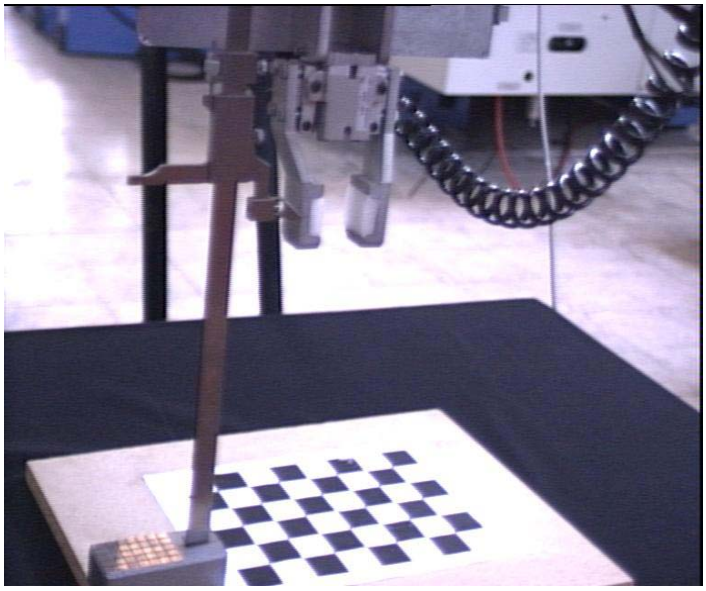


Figure D.5: Setting The Height of The Gripper

Step 5 :

Now remove the gauge and press “Main Calibration Position” button (see Figure D.6) to move the robot to the main position (see Figure D.7).

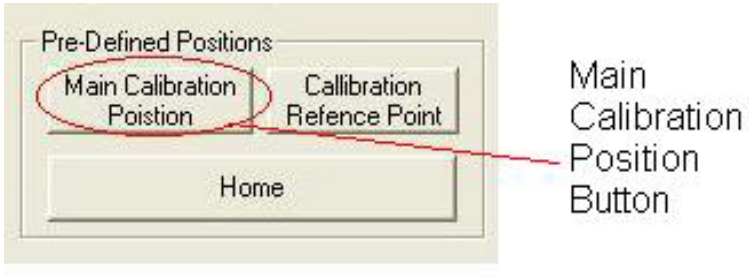


Figure D.6: Calibration Button

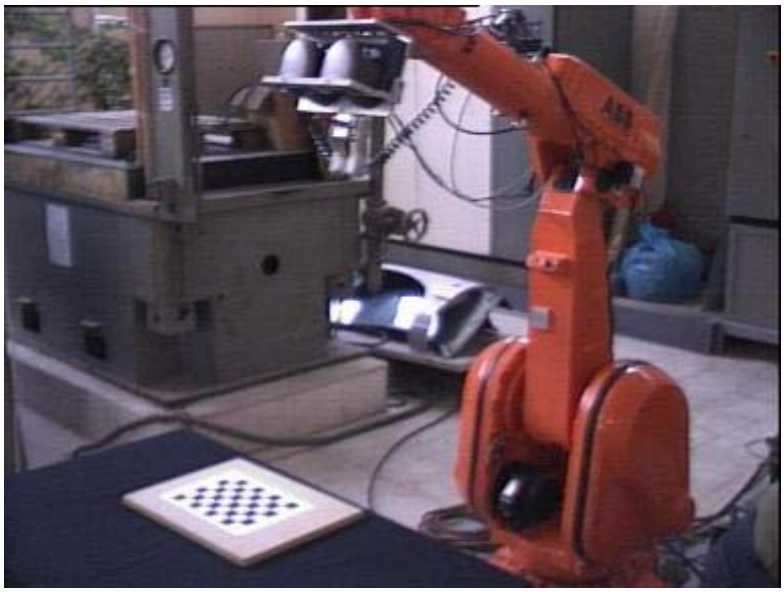


Figure D.7:Main calibration position of the robot

Step 6 :

Press “Capture Images” to capture images from both cameras (see Figure D.8).

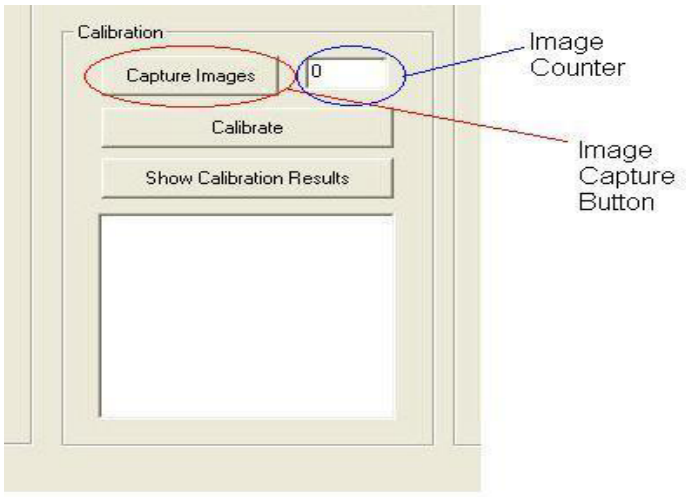


Figure D.8: Image Capture Button

The system will wait for a time period which can be defined by the user, and will start capturing images. It will also wait about this time period between two captures to allow user arranging the board orientation for the next capture (see Figure D.9). User can see the number of the images captured from the Image counter box. After having captured all images, the program warns the user.

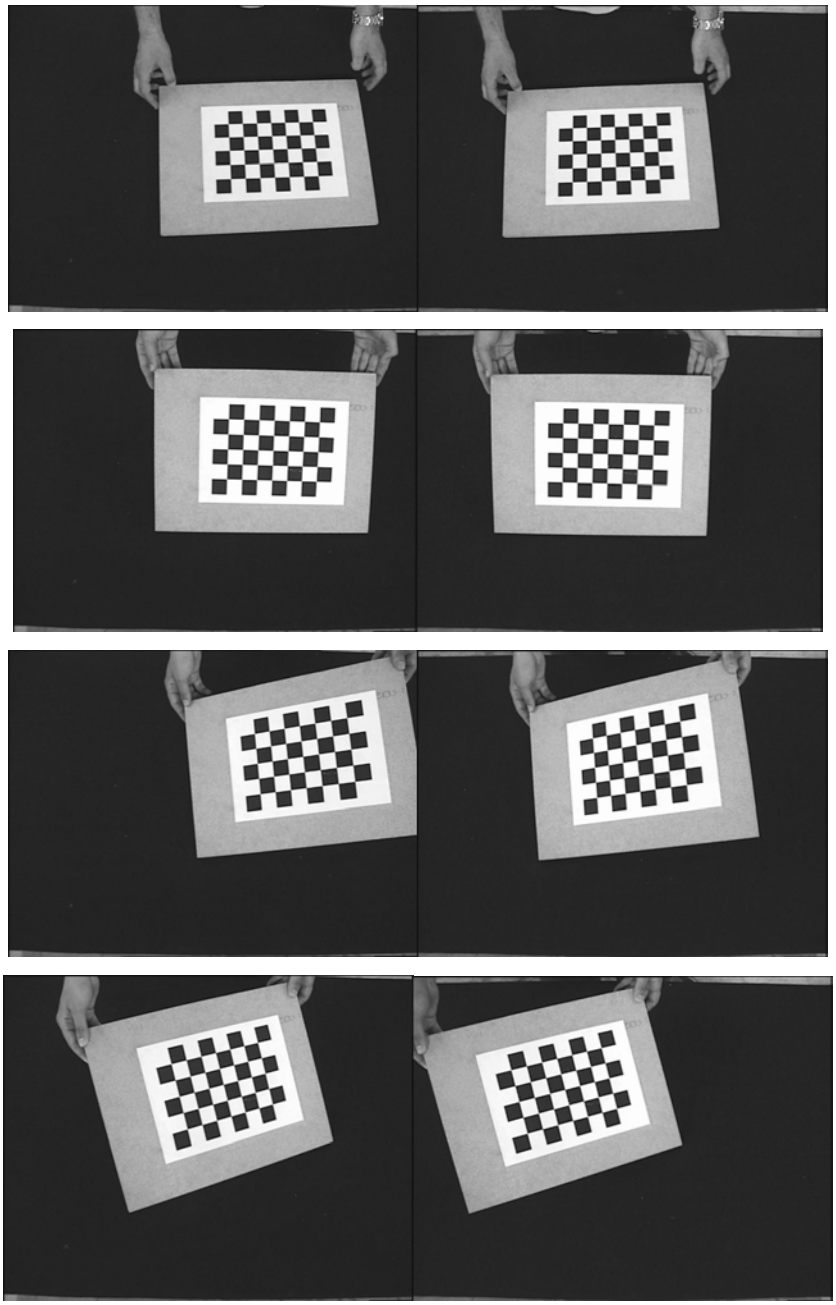


Figure D.9: Calibration Image Samples for
Sony(left) and Axis(Right) Cameras

Step 7 :

Press “Calibration” button to start calibration process (see Figure D.10).

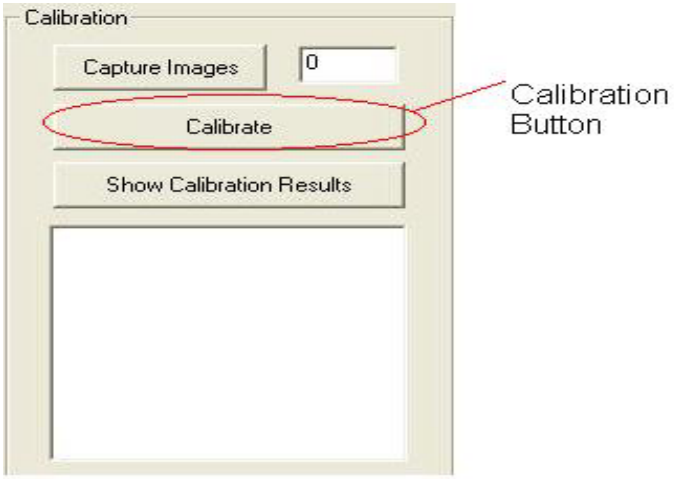


Figure D.10: Calibration Button

The program stops responding during this operation. After extracting calibration data for each camera, program informs user. Main calibration routine results in 26 secs for each camera. Time depends on the number of the images and the number of the points. When the calibration process end program saves the results, update itself and show results in the result box (see Figure D.11).

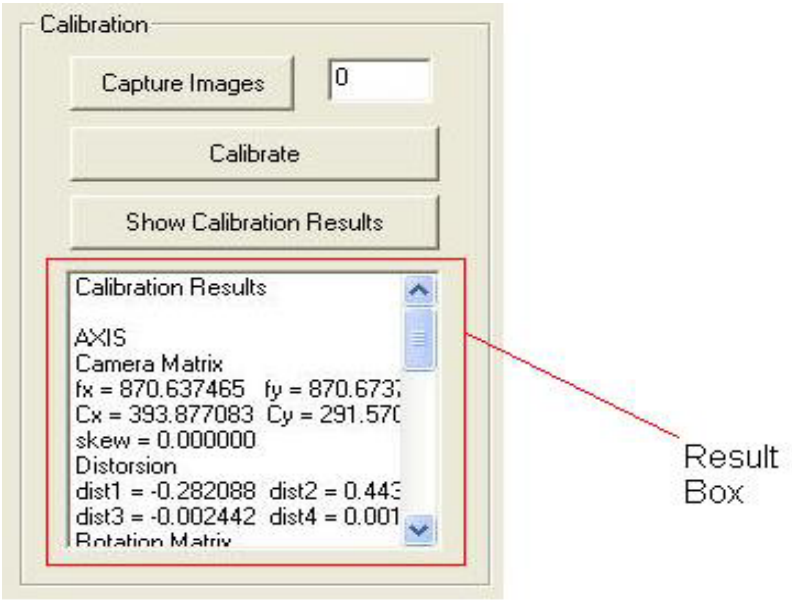


Figure D.11: Calibration Results Box

Now both cameras are calibrated and the system is ready for operation.

APPENDIX E

SETUP DRAWINGS

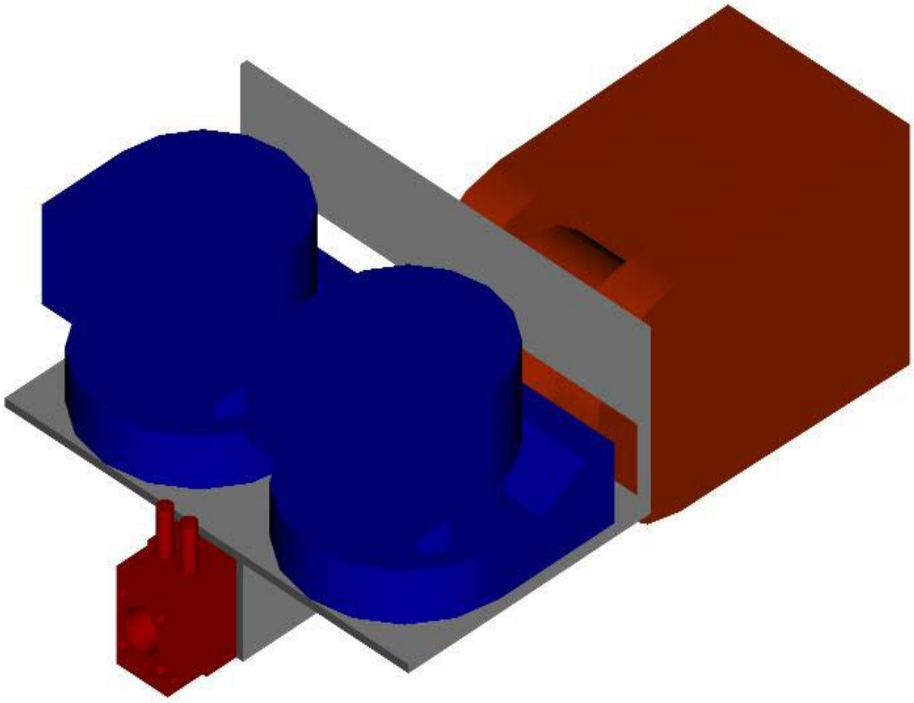


Figure E.1: Main Setup (Cameras Mounted on the End Effctor)

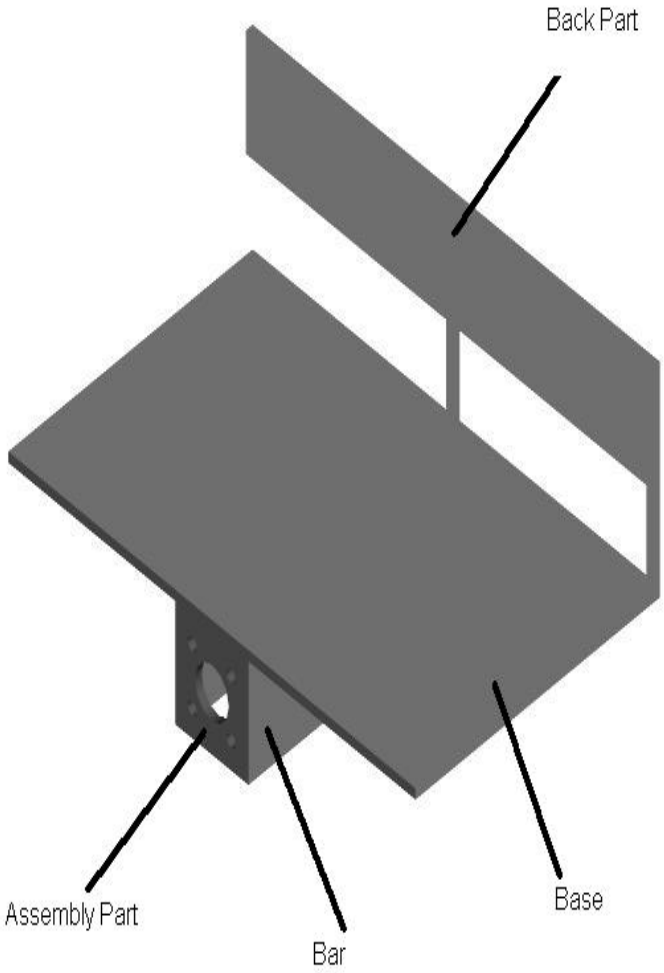


Figure E.2 : Camera Holder Assembly

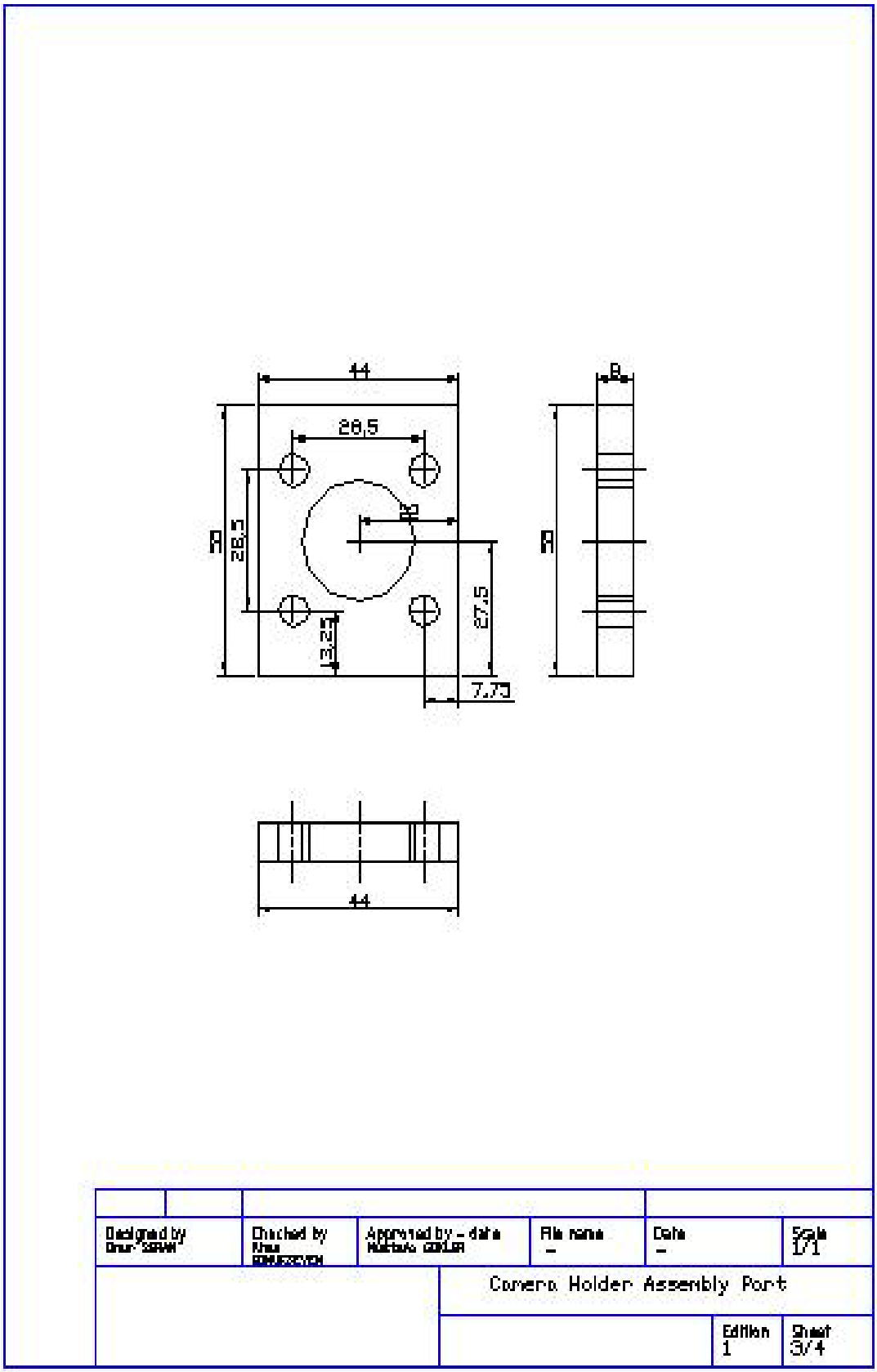


Figure E.3 : Camera Holder Assembly Part Drawing

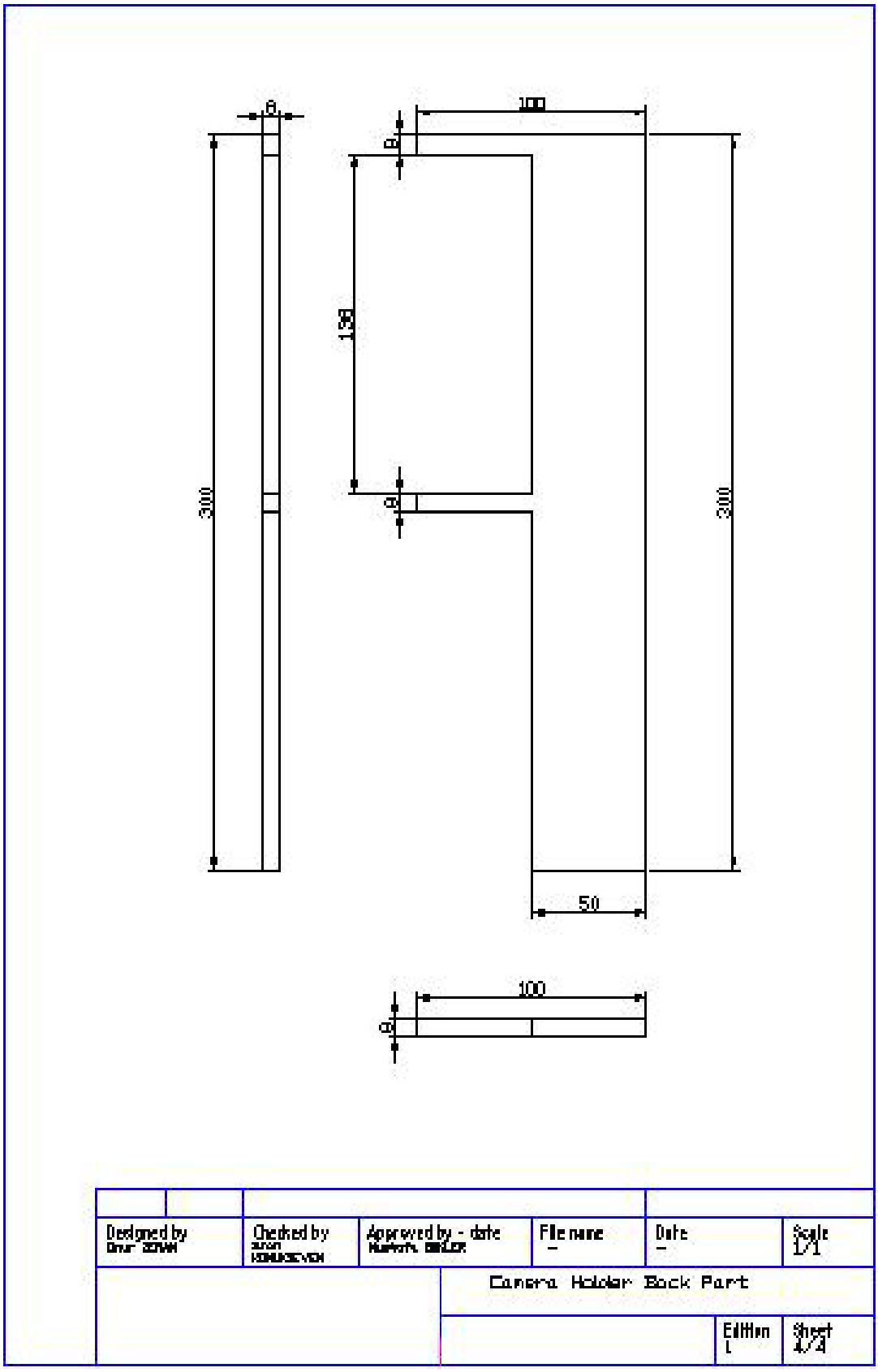


Figure E.4 : Camera Holder Back Part

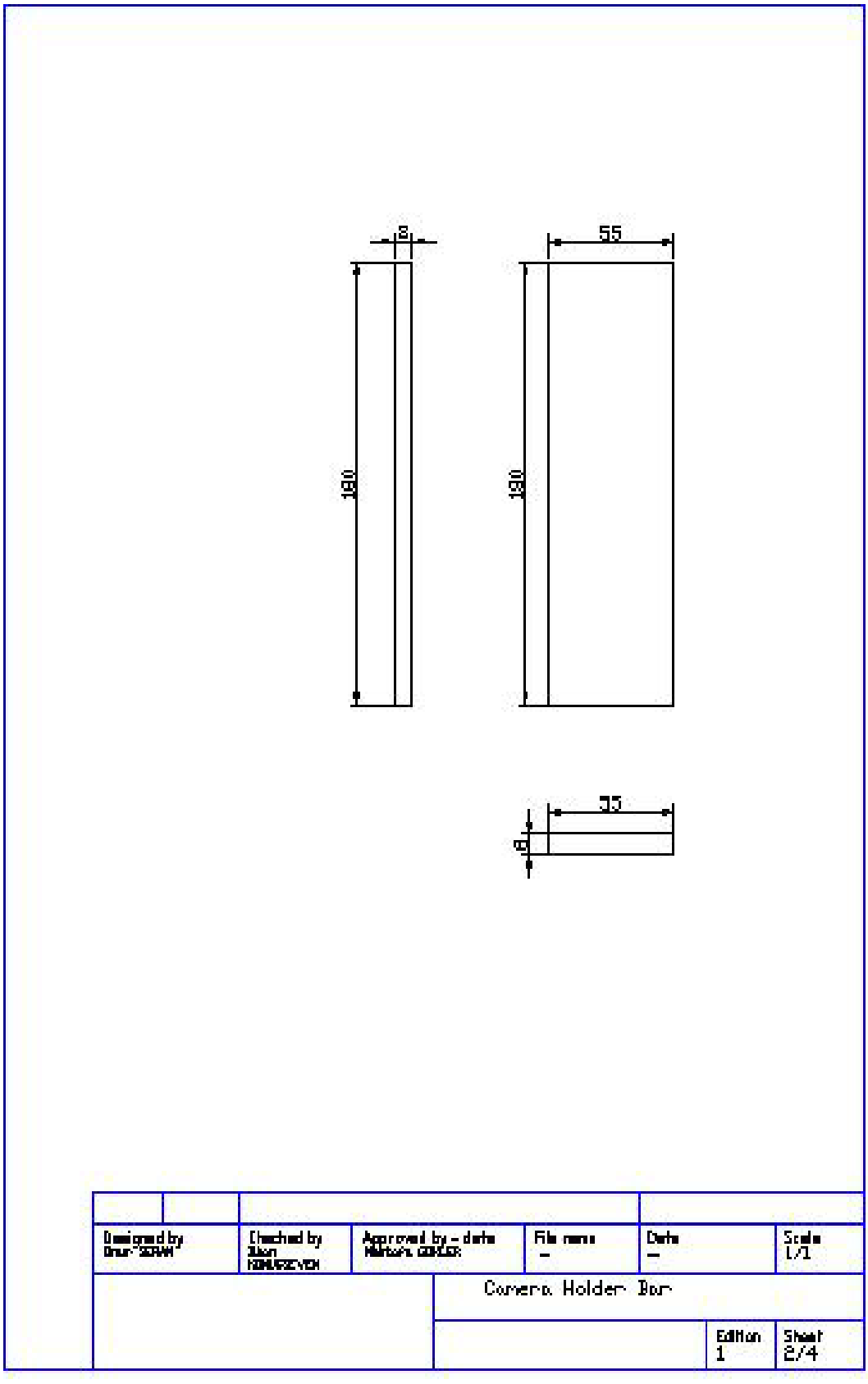


Figure E.5 : Camera Holder Bar Drawing

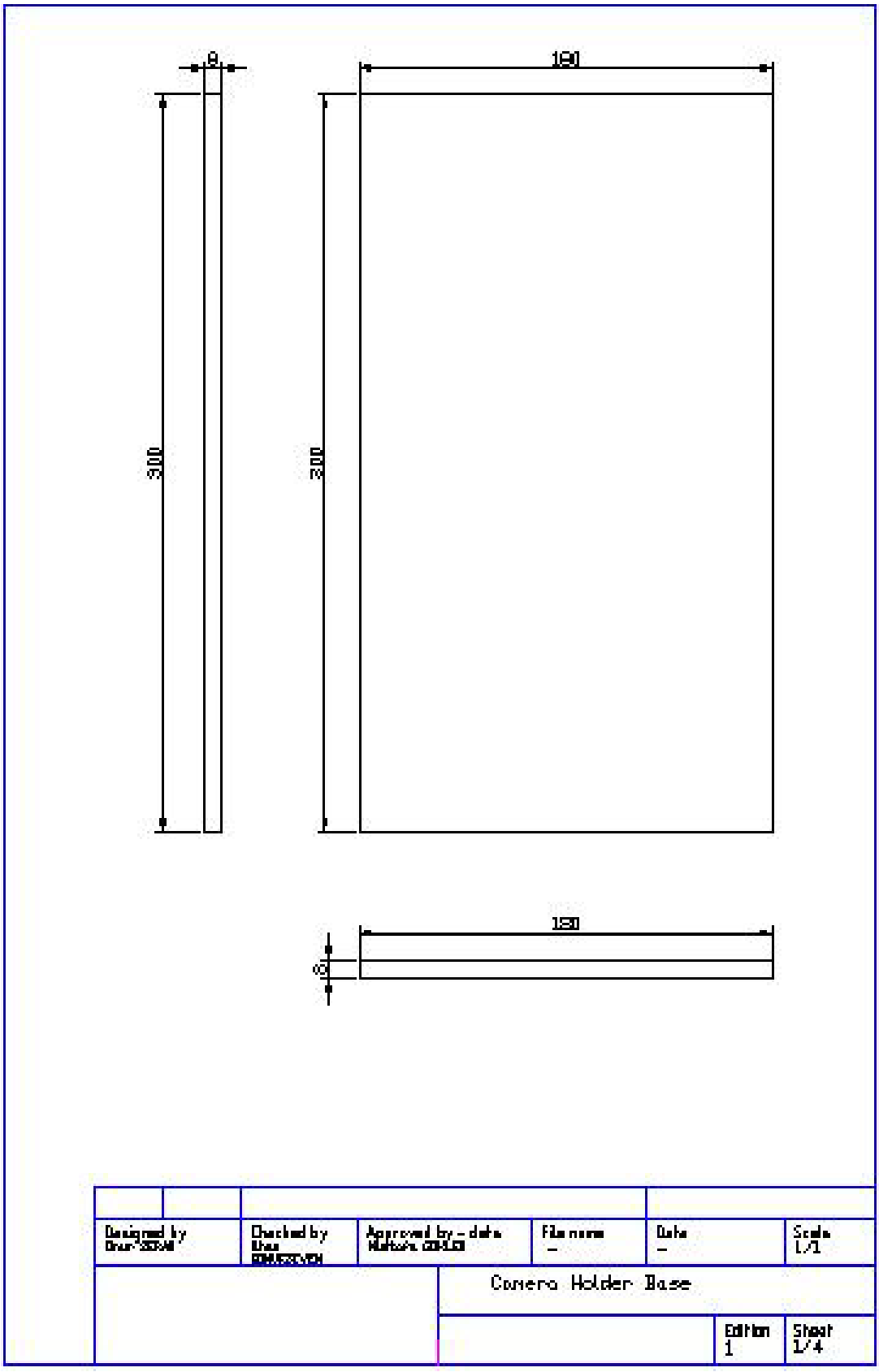


Figure E.6 : Camera Holder Base Drawing

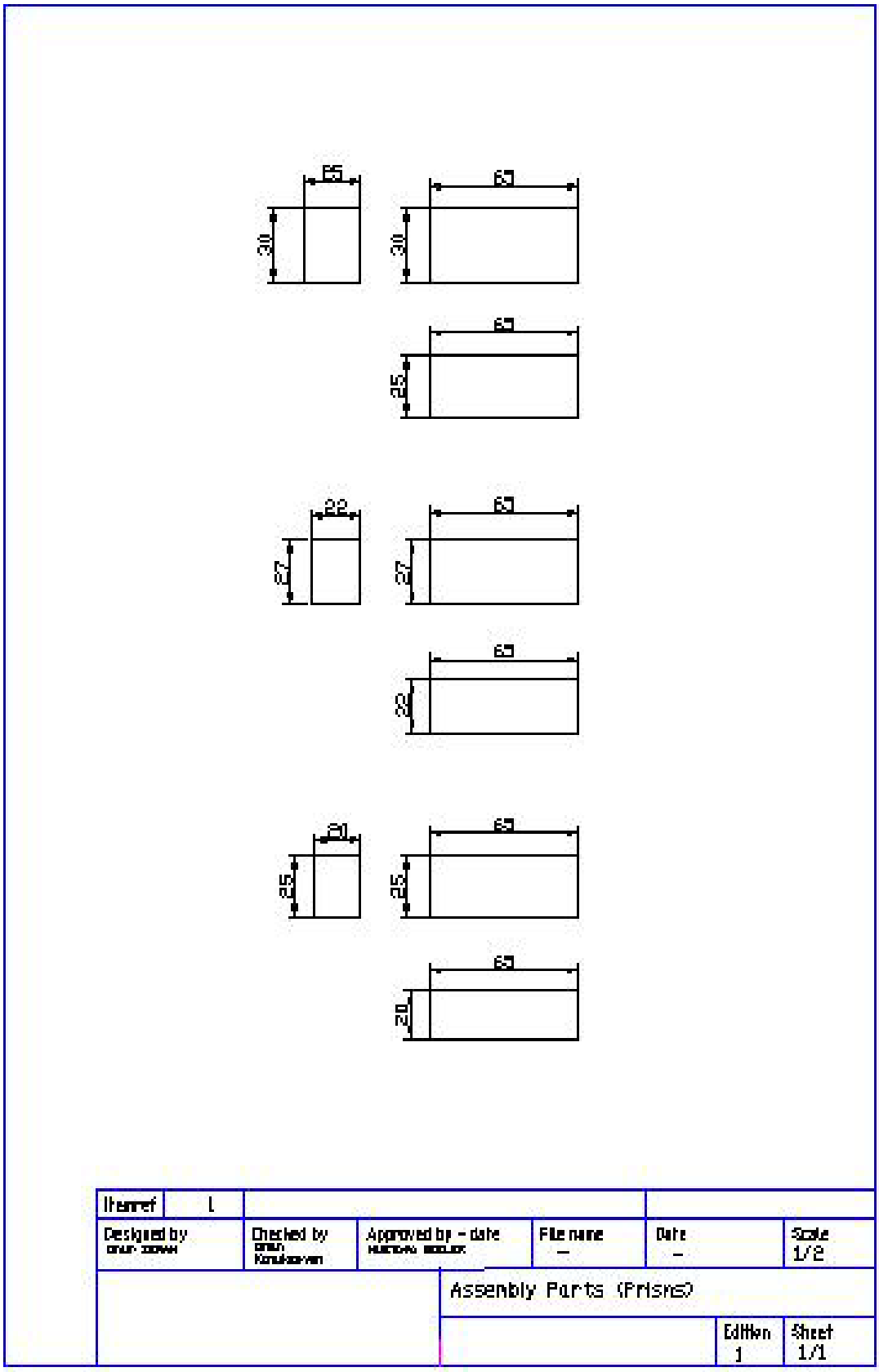


Figure E.7 : Assembly Parts(Prisms) Drawing

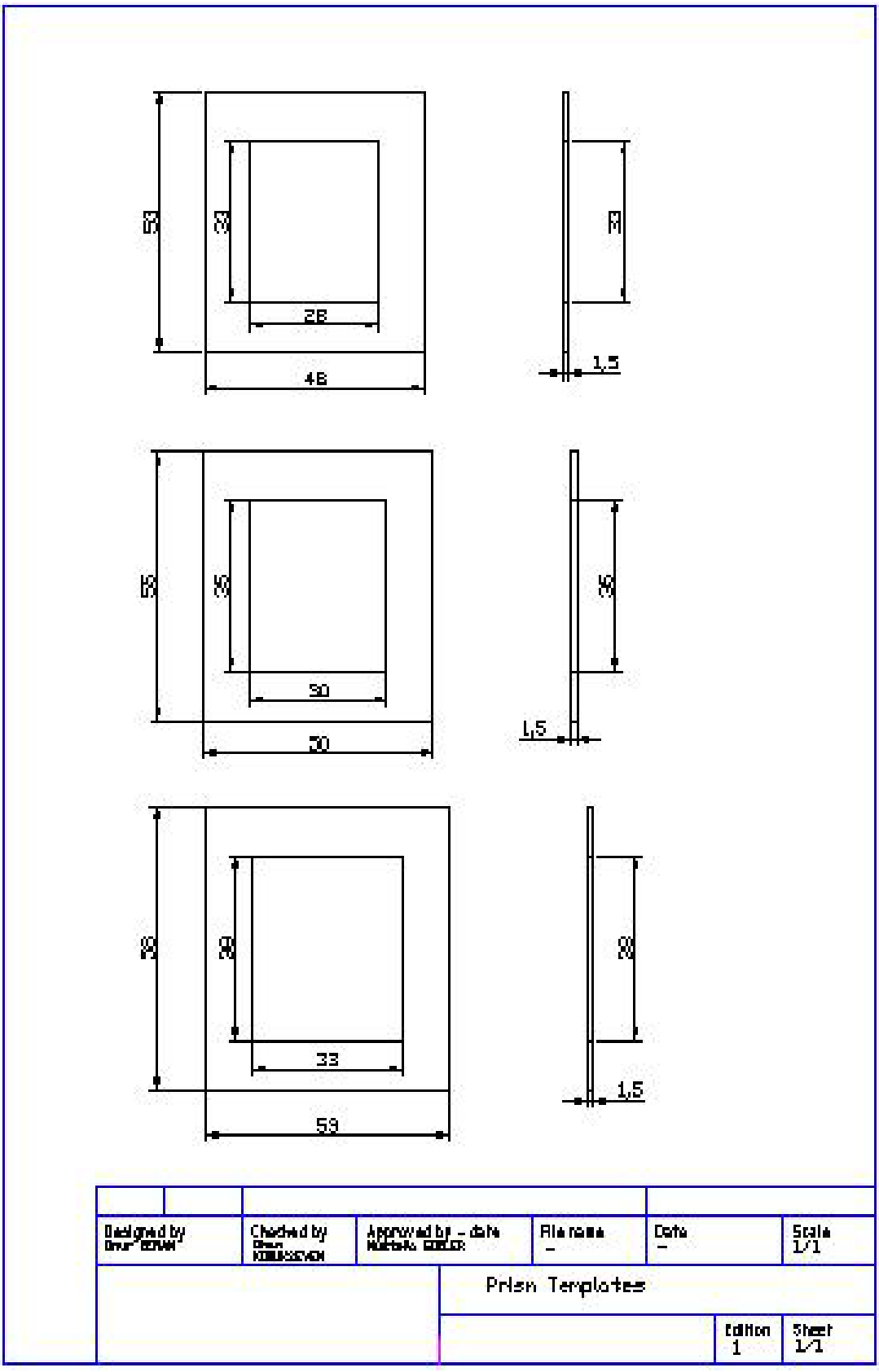


Figure E.8 : Prism Templates Drawing

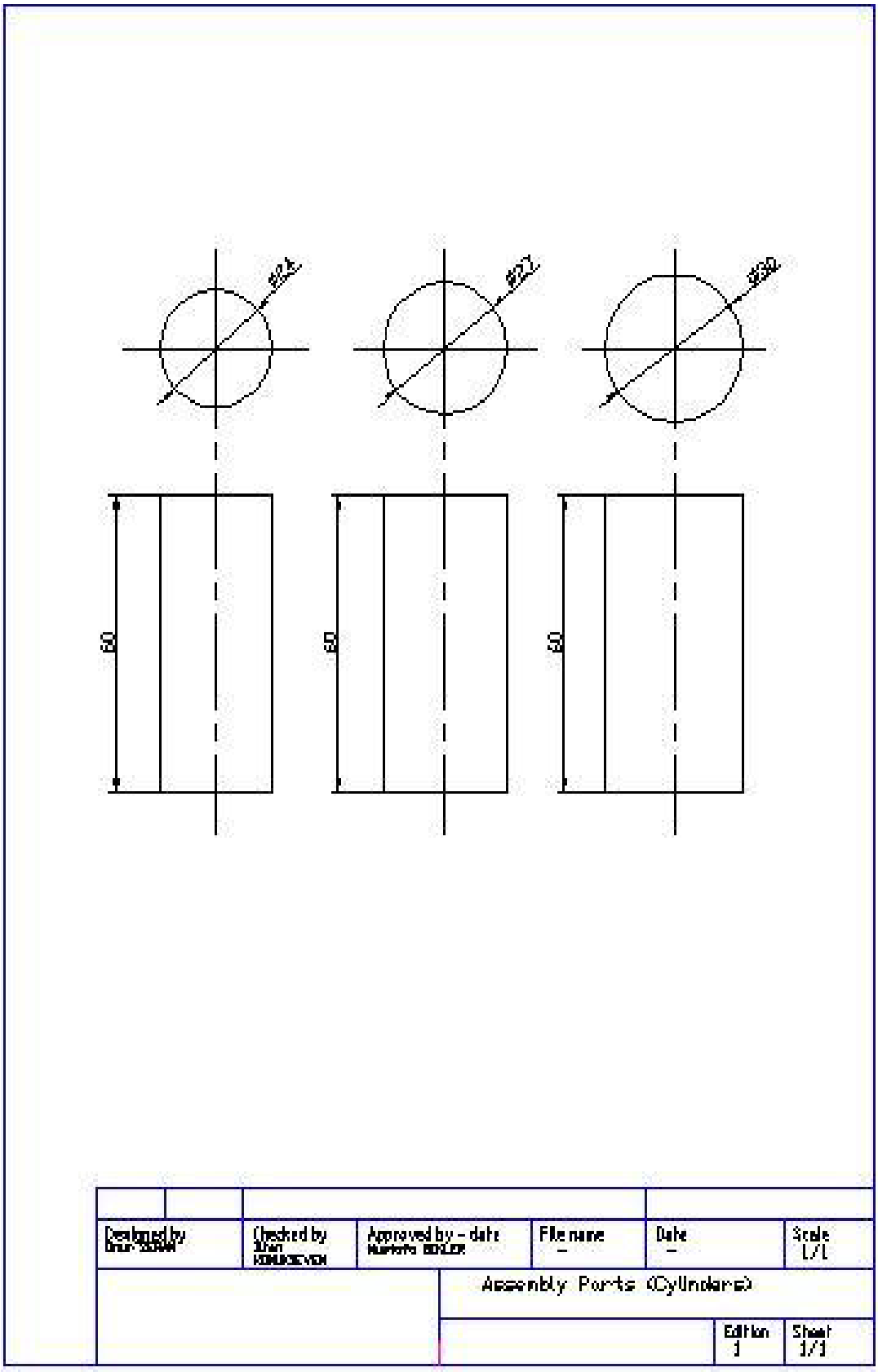


Figure E.9 : Assembly Parts(Cylinders) Drawing

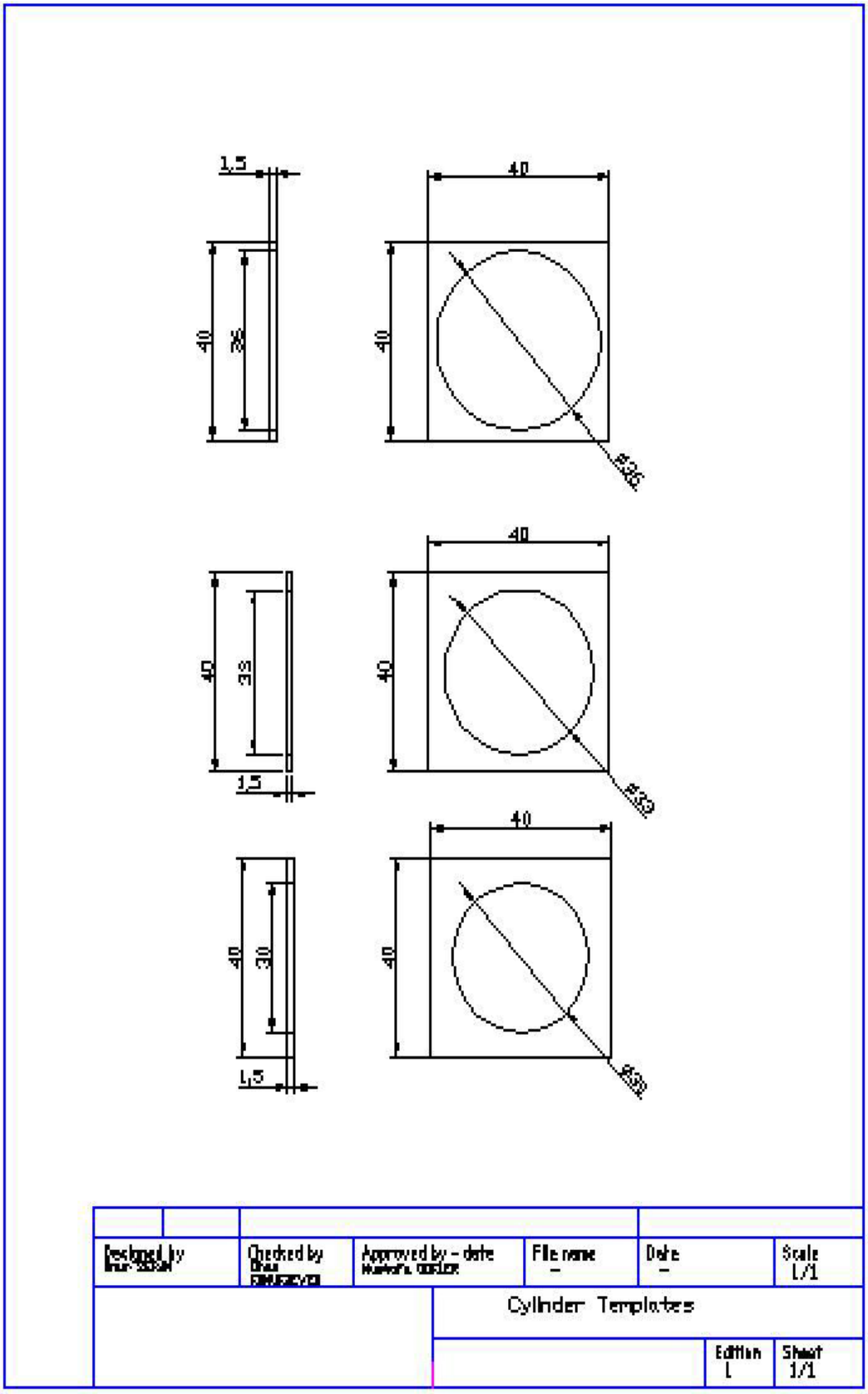


Figure E.10: Cylinder Templates Drawing