2012

SCENE PERCEPTION AND MOTION PLANNING THROUGH ROBOTIC VISION

Durga Rajan
University of Windsor

SCENE PERCEPTION AND MOTION PLANNING THROUGH ROBOTIC VISION

by

DURGA RAJAN

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada
2012

SCENE PERCEPTION AND MOTION PLANNING THROUGH ROBOTIC VISION

by

DURGA RAJAN

APPROVED BY:

Dr. Nader Zamani
Department of Mechanical, Automotive and Materials Engineering

---

Dr. Huapeng Wu
Department of Electrical and Computer Engineering

Dr. Xiang Chen, Advisor
Department of Electrical and Computer Engineering

Dr. Rashid Rashidzadeh, Chair of Defense
Department of Electrical and Computer Engineering

May 15, 2012

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

The use of robotic vision is extensive in any field. This thesis aims to apply the theoretical concepts of the coverage strength model in order to achieve a task using robotic vision. The task mentioned is motion planning for the robotic arm such that it does not collide with an obstacle while performing its operation. This research can be applied to robots in an industrial work cell or an unmanned system. Hand in eye configuration is used to control the robotic motion. Different levels of calibration are implemented, as a camera network is involved. An algorithm is used to implement the motion planning. This algorithm also involves the pose estimation of the objects used in the work cell. The work cell is modeled with a camera network and a robot. The calibration helps to visualize the position of the various entities of the work cell. The algorithm is applied to the physical experimental setup and results are recorded.

Dedication

To my friends and family.

Acknowledgements

I would like to thank my advisor, Dr. Xiang Chen, for his guidance, encouragement and support throughout my masters studies. I would also like to extend my gratitude to Dr. Huapeng Wu and Dr. Nader Zamani for their valuable suggestions and comments.

I would like to take this opportunity to thank Dr. Maher Sid-Ahmed, Dr. Boubakeur Boufama and Dr. Ana Djuric for their guidance and help. I am grateful to my fellow students, particularly Aaron Mavrinac and Jose Alarcon Herrera, who have helped me by providing useful comments at all stages of my masters degree.

I am grateful to the University of Windsor for the support provided in completing my masters degree. I would also like to thank the department technicians, Mr. Don Tersigni and Mr. Frank Cicchello, and the technical support center staff.

I thank my friends and family for their help and support.

Contents

List of Figures

Chapter 1

Introduction

Robotic manipulators have always been the major components of automated industries. The design of an industrial robotic work cell is a complex issue and it is highly dependent on the application and the environment. A robot equipped with a camera works more efficiently and prevents the damages that could occur in an industrial setup that has no vision component for the robot. A camera network, also known as the vision sensor network, is always considered an essential element in a robotic work cell design. The use of visual information to control a robot has been used extensively in many sectors for various applications. The vision sensor network in a robotic work cell helps to monitor the work cell continuously. It leads to perceiving the entire scene in a robotic work cell. A dynamic work cell can be effectively controlled only with the help of a vision sensor network. This thesis also deals with a work cell consisting of a calibrated camera network. It is also interesting to note that a dynamic entity might enter this robotic work cell at any given time. These dynamic entities can either denote the target of interest or an obstacle to the target. The target of interest will be grasped by the robot's end effector or gripper. The network of cameras helps to locate these dynamic entities and differentiate whether they are the target or obstacle. The robotic manipulator has to adapt to the position data provided by the vision sensor. The coverage strength model [6, 7] is used to achieve the detection and discrimination between the target and obstacle. The validation of the coverage strength model is found in [8] and an application of the coverage strength is found in [9]. Section 2.4 explains the coverage strength model in detail.

1.1 Computer Vision and Robotic Vision

1.1.1 Computer Vision

Computer Vision is the combination of image processing, pattern recognition, and artificial intelligence technologies, which focus on the computer analysis of one or more images taken with a single or multiple sensors. The analysis recognizes and locates position and orientation, and provides a sufficiently detailed symbolic description or recognition of those imaged objects deemed to be of interest in the 3D environment [10].

Computer vision involves acquiring, processing and analyzing images in order to make certain decisions based on the output of the analysis. Computer Vision can be defined by discussing its problems. The primary issue of computer vision is computing the 3D properties of the world scene from the images acquired. The 3D properties are the geometric and dynamic properties of the world scene. Computer vision is a huge field and it is still growing with new inventions and development. Areas related to image processing, robotics, artificial intelligence and pattern recognition merge effectively with computer vision. Many terms and tools associated with computer vision overlap with other fields or disciplines. For instance, image processing is quite different from computer vision as it deals mainly with image to image transformations. The overlapping occurs due to the fact that images needs to be to pre processed using image processing techniques before they are given as inputs to the computer vision algorithms [11]. Computer vision has widespread applications and offers an array of research topics. This thesis work addresses research areas related to “active and purposive vision” and “pose estimation” with application in “surveillance and hand-eye robotics systems”.

1.1.2 Robotic Vision

The term Robot vision is used for systems that take the principles of animated attention, purposive perception, visual demonstration, compatible perception, biased learning, and feedback analysis [10]. Robot vision can be assumed as an animated vision through the use of attention control. Almost all 3D vision applications are to be treated by analyzing images at different viewing angles and distances. This idea of animated vision operates on mechanisms of selective attention called attention control. Environmental Information that is more relevant for the vision task needs to be extracted for processing. These data may be qualitative or quantitative in nature.

Data acquired proves useful only when it instantaneously perceives the scene in reality. There should always be a close relationship between the physical data acquisition and the appearance of the actual situation. After acquisition, images are processed using their geometrical parameters. Compatibility holds between the geometric shape (global) and gray value structure (local) of the image. The output signals of the imaging processes need to be converted into 2D or 3D features. This in turn is fed to the downstream robotic control (arms, wheels).

1.2 Autonomous camera equipped robot system

A robot is a mechanical system which can be programmed to do a specific task. It consists of an actuator, which is mobile, and a controller fed with software. The software within the controller helps to solve the inverse kinematics. A camera equipped robot system can be used for two purposes: a vision supported robot system or as a robot supported vision system. The camera equipped robot could have the hand in or the hand off configuration. An autonomous robot system must incorporate four important characteristics, they are: appropriate interpretation of the situation, corporeality, competence and emergence. An autonomous camera equipped robot system does autonomous task solving, which helps in visual perception and action. The characteristics of an autonomous robot system can be obtained by animated attention, purposive perception, visual demonstration, compatible perception, biased learning, and feedback analysis [10]. All these characteristics are inter -related and evolve from one another.

1.3 Visual Sensor Network (VSN)

Visual sensor network plays a major role in computer vision and its related fields. Visual sensor networks aided by human monitoring, were famous in the field of surveillance and security for a quite a long time. The feedback obtained from a visual sensor network can be used to control various robotic manipulators. A visually controlled robot can easily emulate the skills of a human being. Computer Vision algorithms have helped to eliminate the need for human monitoring. This thesis work also aims toward building a camera network that helps in the automation of an industrial based robotic application. It is important to understand the various terms and definitions involved with the explanation of a camera

network. A vision sensor network can be described as a distributed smart camera network.

There are two types of VSN, they are: Homogeneous and Heterogeneous VSN. Depending upon the requirement we can select the type of VSN necessary. This research work involves homogeneous VSN system. In homogeneous VSN the camera sensor nodes are of same type and of similar capabilities, and are connected to one or more base stations. In heterogeneous VSN the camera sensors are of different types and capabilities [12]. A VSN requires higher bandwidth and energy to transmit data. VSNs are also used for environmental monitoring, surveillance applications and industrial control. There are options available to process the image data locally at the camera level, which will reduce the transferring of data through the network. Hence, this function leads to the development of a VSN that consists of totally independent and intelligent system of distributed cameras. This system provides only highly relevant input to the processing system (base station). The deployment of a visual sensor network in a robot work cell increases the effectiveness of the robot. This type of robot can function like a human being for certain tasks in different industries. This type of VSN has strict boundaries on the maximum allowable delay of data from the camera to the processing unit. Yet another challenging task in VSN is object tracking, since it is computationally intensive and it involves real time processing of images. This can be improved by increasing the number of views of the system. Another important feature of VSN is location and orientation of camera, which is obtained by the camera calibration process. This process is widely used in this project to locate the object in a precise manner.

1.4 Multi Sensor Robot System

Robots need to operate in an environment that is filled with uncertainties. The uncertainties could be with the perception of the environment, manipulator motion or task planning. Robots can easily perceive their environment if the environment is already known, but the task becomes complicated when the environment is uncertain. If there is going to be any unexpected motion in the work cell, the robot then faces the issue of moving without hindrance. This thesis deals with one of the uncertainties that could occur in a robotic work cell: this uncertainty is modeled as a dynamic obstacle. The usage of the predetermined positions of objects in a robotic work cell makes the task very simple for a robot. However, single sensors will not help to resolve a many problems that could arise in an

unstructured environment. Hence, it is necessary to use multiple sensors in order to make the robot movement robust, even in an unstructured environment. Instead of depending on a data from one single source, combining the data from multiple sources helps to reduce uncertainty in the final decision taken. The main objective of a multi sensor network is to combine all the information from the different sources into a consistent environment description.

Multi-sensor systems have gone through many developments in recent years. Single sensor systems can only provide partial information about the system. Hence, multi sensor systems play a major role. Robot systems must use more than one vision sensor in a dynamic manner in order to make the multi sensor robot system intelligent and autonomous.

1.5 Literature Survey

This thesis focuses on the collision avoidance of industrial robots, when there are changes in the environment of the robot's work envelope. An approach is being adopted in this project in order to prevent the robot colliding with obstacles. Obstacles are modeled such that they can be tracked continuously with the help of a camera network. This section discusses about some of the previous research works done, related to this thesis work.

1.5.1 Robotic Vision in Industries

Robots work in an environment that is customized according to the needs and dimensions of the robot. In industries many subsystems closely work together to guarantee a collision free environment. Sensor system is the most critical subsystem. By making use of sensory equipments like cameras, the robot senses the environment and locates the objects (static or dynamic) that are blocking the path of the robot. As discussed by Brecht et al. [13], a human-robot coexistence system incorporated with a camera network was already proposed in [14] in the early nineties. They use reference images for detecting human operators and other obstacles. Their approach was not optimal with regard to computation time and memory requirements. In 1999 Steinhaus et al. [15] developed a MEPHISTO system, which is a combination of both local and global sensors. This includes a laser sensor at the top of robot and two surveillance cameras at the corner of the cages. They also use the concept of reference images and a distributed environmental model in order to establish local path planning. Cervera et al.[16] state that in order to avoid collision in human-robot

systems a 360 field of view needs to be incorporated in the robotic vision system . In [17], a robotic vision system is built in order to avoid dynamic objects using a 3D camera. This robotic vision system requires a robot, a sensor and a logical unit to perform its task. On the hardware side, a controlling program running on a PC evaluates the workspace data provided by the 3D camera. Accordingly, commands will be transferred to the robot thereby controlling its movements. In [18], Winkler et al. investigated the approach of dynamic collision avoidance. This involves artificial potential fields, visual servoing and impedance control. The software needs to be common for both the static and moving objects.

1.5.2 Camera View Selection

In [19], Tessens et al. have discussed the view selection in a distributed smart camera network. The mode of operation and precision of the output depends on the deployment of the camera network. The major advantage of the network is the elimination of occlusion problems. A limited number of additional views are used to obtain the desired observation. The usage of smart cameras with in built image processing and a communication system will help to reduce the bandwidth usage required for networking. In [20], Monari et al. have discussed the several algorithms used for camera selection in sensor networks. It has been stated that the main drawback in all methods of camera selection discussed is the need for a learning period. Learning period is defined as the correspondences between objects disappearing in one camera and reappearing in another. This research work uses a model called coverage strength model for view selection. The coverage strength model has been used more widely for sensor planning in literature than for view selection, as discussed in [9]. View selection using the coverage strength model establishes the smoothness of the view sequence, in addition to the selection of the best view [9].

1.5.3 Purposive Sensing in Robotic Manipulators

Chen et al. [21] did an extensive survey about recent developments in active robotic vision systems. The main objective of purposive sensing in the field of robotics is to produce good images that can be efficiently understood by the robots. Highly redundant visual information increases the process efficiency; however, redundancy could also help for reconstruction. They have also discussed purposive sensing in robotic manipulators. The

use of robotic manipulators has demonstrated increased manufacturing productivity. This increase depends critically on the simplicity with which the robot manipulator can be re-configured or re-programmed to perform various tasks. Actively placing the camera to guide the manipulator motion has become a key component of automatic robotic manipulator systems. The vision-guided approach achieves high precision in manipulation along with improved productivity. For the assembly or disassembly tasks, a long-term aim in robot programming is the automation of the complete process chain, from planning to execution. Positional uncertainties are also an important error that needs to be considered in robotic vision. In the case of a hand in eye system, various factors like resolution, depth of view and field of view are taken into consideration. Kececi et al.[22] used a mobile camera along with a 6 DOF robot to monitor a disassembly process. The optimal view pose is found from a set of candidate pose estimates. An optimal view pose, which avoids collisions and occlusions is determined. Hence, the optimal view pose helps for sensor planning in the work done in [22]. As per the survey done in [21], most of the purposive sensing methods designed for robotic manipulators deal with sensor planning and optimal sensor placement.

1.5.4 Obstacle Avoidance in an industrial robotic work cell

In [18], Winkler et al. discuss collision avoidance in industrial robots. They use the concept of artificial potential or force fields. The field is generated from the charges on the obstacles. The robot path control is achieved using impedance control. Image processing using a USB camera helps to find the position of the dynamic obstacles. The robot work cell is monitored using one USB camera and the obstacle is mounted on the end effector of another robot. The obstacle position is found using step edge detection and Hough transform; after the obstacle detection, the path planning is done using artificial potential fields. In [23], Gecks et al. present an industrial work cell that is supervised by a group of stationary cameras in order to prevent collisions due to human beings or obstacles. Difference image method is used to decide the robot's path. Their paper helps to achieve safe human robot coordination with no collisions. The difference image method is implemented using a set of static reference images of the robot's workspace. The future robot's configuration is checked for any collision using path planning methods and, based on the detected obstacles, the system plans alternate paths around the obstacles.

1.6 Thesis

1.6.1 Problem Statement

In this thesis, the concept of visual feedback is used for robot's motion planning. Robots play a major role in every part of life. They might offer assistance in the home, function as an appliance, or work in an industrial environment. Dynamic obstacles can be found in any kind of work cell of a robot. Obstacle detection plays an important role in a dynamic environment. In an industrial robot's pick and place application there might be dynamic obstacles in addition to the object to be picked. The approach taken for this thesis can be compared to a pick and place application where the target objects are in continuous movement on a conveyer belt. The overall objective is to detect occlusion of the dynamic target object and to prevent collision of the robotic arm with the dynamic obstacle. The robot's movement has to be changed in order to avoid the dynamic obstacles. Dynamic or static obstacles can disrupt the operation of any robot. The robot has to find a way to deal with the occlusion of the target object caused by dynamic obstacles. Collision avoidance is an important aspect in an industrial robotic cell. Hence, it is necessary for the robot to perceive an obstacle in order to prevent collision. The robot must reach the target when there are no dynamic obstacles that would cause collision between the obstacle and robotic arm. Once occlusion is detected the robot must change the direction of its motion such that it does not collide with the obstacle. The objective is achieved using a camera network equipped with coverage strength model.

1.6.2 Thesis Organization

The objective and the important terms related to this research work have been explained in the previous sections. The literature relevant to the thesis topic has been discussed in Section 1.5. The rest of the thesis is structured as stated in the following paragraph.

Chapter 2 explains the theoretical concepts related to computer vision that will be used in the remaining chapters for the implementation of the thesis objective. The primary concepts explained are the camera and its calibration, graph theory and the coverage strength model. Chapter 3 deals with the main objective of this thesis. It provides the solution along with the methodology taken using a block diagram and algorithm. Chapter 4 explains the experimentation. It describes the apparatus used and procedure for the experiments and analyzes the results. Finally, Chapter 5 provides conclusions along with recommendations.

Chapter 2

Theoretical Foundations

2.1 Geometry in Computer Vision

Robots seek the help of computer vision to detect and perceive the 3D environment. Geometry plays a major role in understanding the 3D structure of the robot's environment. The three main geometry types involved in computer vision are Euclidean, Affine and Projective Geometry.

2.1.1 Transformations

The transformations involved during the formation of an image are based on the Euclidean, Affine and Projective geometry properties. Euclidean transformation, also known as the rigid transformation, is invariant to angles and distances. It involves a translation and rotation. The Euclidean transformation of vector x can be represented as given below:

$E[x] = Rx + T$, where R represents rotation matrix and T represent the translation vector from origin.

Also, R is orthogonal and $det(R) = 1$. In Affine geometry the ratios and parallel property are invariants. Affine geometry is the super set of Euclidean geometry. Projective geometry is the most suitable for optics since optics require a perspective transformation. When the human eye looks at a scene, objects in the distance appear smaller than objects close by, which is known as perspective. Projective geometry is the super set of affine geometry. Here the invariant is cross-ratio. Point P of an n-dimensional projective space has $(n+1)$ homogeneous coordinates. A projective transformation from P^n to P^m can be given as $P \rightarrow WP$, where W is $(m+1) \times (n+1)$ matrix and it has $[(m+1) \times (n+1)] - 1$

parameters. W denotes the transformation matrix from n-dimensional to m-dimensional space. W is a square and non-singular matrix, iff the projective transformation is a homography. When $m = n$ the projective transformation is termed homography. Hence, in a three dimensional projective space each point has 4 homogeneous coordinates and W has 16 entries. The projective basis of P^n has a set of $(n+2)$ points. In a 3D projective space any five points in space can form the projective basis if no four points among them are coplanar. In Affine geometry, the transformation matrix W has only 12 parameters, whereas in Euclidean geometry W has 6 parameters, which constitute the rotation and translational components [1].

2.1.2 Pose Estimation

Pose estimation is vital for any vision related task. The external calibration of a camera is also a pose estimation technique as explained in Section 2.2.2. Pose can be defined as a transformation required to map an object from its own coordinate system into an agreement with the sensory data [24]. The main objective of pose estimation is to find the pose of an object in the image with reference to a coordinate system. The term pose denotes six components: the three rotational and three translational components. Pose is the combination of the position (translation) and orientation (rotation) data. Pose estimation can deal with a single, stereo or even sequence of images. The pose estimation method differs according to the object whose pose is found out. Certain methods work for a particular class of objects. The pose consists of a rotational and translational transformation. The rotational transformation can be expressed using different notations according to the order of the rotation and various other factors that will be discussed in the following section. The methods for pose estimation can be classified into three categories, they are listed below [25]:

1) Geometrical method: If the geometry of the object is known, then the projected image of the object in the camera is a function of the object's pose, as the camera is calibrated. This method requires extracting some control parameters from the image, such as the corners or features in order to establish the correspondence between the 3D scene points of the object and its 2D image points. This research work also uses this method in order to find the pose of the target and the obstacle. Planar targets are used and their geometrical details determine the pose.

2) Genetic Algorithm method: In this case calibration of the camera is not required. This method is used when the pose is not required to be computed in real time.

3) Learning based method: This uses a set of images of the object in arbitrary poses that are fed as inputs to the system. This indicates the learning phase of the system. Once this phase is over, the system will find the pose of the object when an image of the object is provided.

2.1.3 Pose representations and composition

2.1.3.1 Pose representations

The important representation types used for the rotation component are discussed in this section.

1) Matrix representation: A rotation matrix R in Euclidean n-space is an nxn real orthogonal matrix, whose transpose is its inverse, and whose determinant value ($det(R)$) is equal to 1. The rotation matrix is generated from the Euler angles (ϕ, θ, ψ) by multiplying the matrices obtained by the rotation through the X, Y and Z axis respectively.

2) Axis-Angle representation: Axis-Angle representation is also known as the exponential coordinates of a rotation. It expresses a rotation using two values: a unit vector representing the direction of the axis and an angle representing the magnitude of rotation about the axis.

3) Quaternion representation: Rotation is expressed in terms of a four dimensional normalized vector. It is the extension of complex numbers.

4) Euler Angles: Euler Angle represents the rotation around the three major axes: the x-axis, the y-axis, and the z-axis. We can represent any pose by the "roll" around the x-axis along the plane, the"pitch" around the y-axis, which extends along the wings of the plane, and the "yaw" or "heading" around the z-axis as a vector (roll, pitch, and yaw). The order of rotation can vary.

In this thesis the Euler angle representation and the matrix representation in homogeneous form are mostly used to represent the rotational component of the pose.

The 3D rotation around an axis can be represented in multiple ways. In this research pose estimation is implemented using Halcon machine vision libraries [1]. The pose representation types available in Halcon are given by the following parameters: 'Order of Rotation, 'View of Transform' and the 'Order of Transform' [26]. By using different values for these parameters various pose representation types are available. Each combination

[1] HALCON. MVTec Software GmbH. [Online]. Available: http://www.mvtec.com/halcon

of these parameter values represents a pose type and is indicated by a code ranging between 0 and 13. Section 4.1.2.1 explains Halcon pose types in detail.

2.1.3.2 Pose Composition

The relative pose of an object F with respect to an object A can be represented as P_{FA}. This also represents an Euclidean transformation from the coordinate system of F to that of A. Pose composition is denoted by the symbol $\diamond$ in this thesis. In case of a homogeneous matrix representation of a pose, determining the composition of two poses simply means multiplying the two matrices corresponding to the two poses. A simple pose composition can be represented as follows:

$$P_{FP} = P_{FA} \diamond P_{AP} \tag{2.1}$$

Equation 2.1 denotes that the composition of 'the transformation from the coordinate system of F to that A' and 'the transformation from the coordinate system of A to that of P' leads to the computation of 'the pose transformation from the coordinate system of F to that of P'. This implies that the pose of an object with respect to another object can be found using a chain of intermediate poses in between them as determined by pose composition.

2.2 Camera Model and Calibration

This section describes the camera model and also camera calibration theory. The camera model is based on the perspective model. The camera calibration equations are derived from the pin hole camera model. Camera calibration is a method used to find the internal and external parameters of the camera.

2.2.1 Pin Hole Camera Model

Pin hole camera model is the basis of all camera models. It is also known as the perspective camera model and is most commonly used to approximate a real camera model. This model helps to understand the process of image formation and also leads to the formation of the calibration matrix. An image can be defined as a 2D transformation of a 3D structure with the help of a vision sensor. There are three coordinate systems involved for the 2D image formation using the pin hole camera model. They are: the image coordinate system, scene coordinate system and camera coordinate system. An image point (pixel) is formed as the

result of a three step process, which is listed below:
Step 1) A 3D Euclidean transformation from the scene frame to the camera reference frame;
Step 2) A 3D to 2D transformation from the camera reference frame to the image coordinate system using perspective projective transformation;
Step 3) A metric conversion for the image to be defined in terms of pixels, using 2D Affine transformation.

The above three steps can be explained in detail. The final output of the above mentioned three steps is a calibration matrix. The pin hole camera model is represented in Figure 2.1. In Figure 2.1 the origin, O, represents the center of projection, Ox and Oy are parallel to the image plane and Oz represents the optical axis. The camera is attached to the reference frame whose center is O [1].

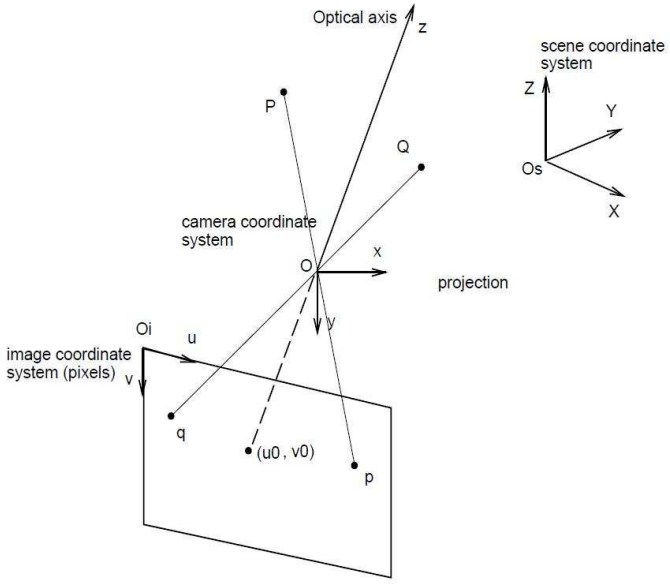

Figure 2.1: Perspective camera model (Image courtesy of [1])

Step 1) 3D Euclidean Transformation:

Assuming that P and $P^{'}$ represent the same point in the scene and the camera reference frame, the relationship between them can be expressed in homogeneous coordinates as

follows:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} r11 & r12 & r13 & t_x \\ r21 & r22 & r23 & t_y \\ r31 & r32 & r33 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2}$$

In Equation 2.2, $P = (X,Y,Z)$ and $P' = (X',Y',Z')$; Also, $r11, r12$..... represent the rotational components and t_x, t_y... represent the translational components required for the Euclidean transformation from the scene frame to the camera frame.

Equation 2.2 can be represented as $P' = EP$, where E is a 4×4 matrix representing the displacement matrix.

Step 2) 3D to 2D transformation:

Using the triangle similarity properties in Figure 2.2 we get the following set of equations:

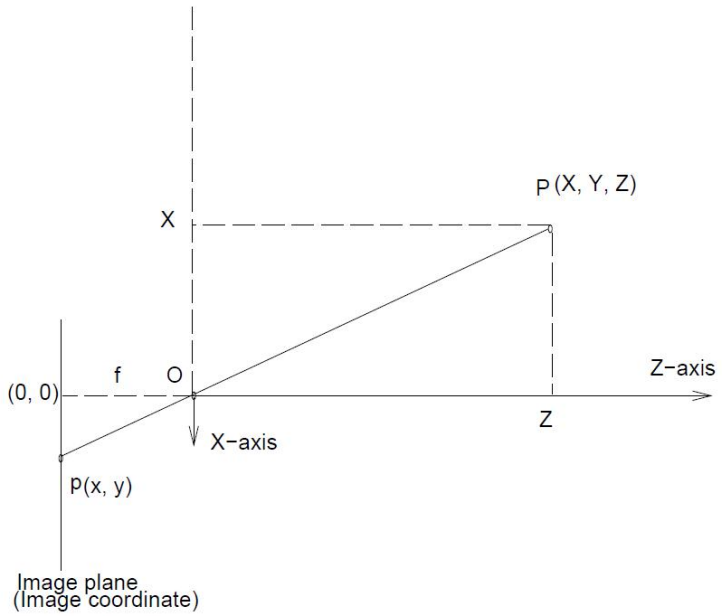

Figure 2.2: Perspective projection of a point (Image courtesy of [1])

$$x = (Xf)/Z \tag{2.3}$$

$$y = (Yf)/Z \tag{2.4}$$

In the Equations 2.3 and 2.4, x and y represent the image plane coordinates and f represents the focal length. Also, these equations represent the projective transformation from 3D to 2D space. Assuming that f=1 this transformation can be represented as shown in Equation 2.5

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.5}$$

Equation 2.5 can be represented as $p = \lambda IP$, where I represents the perspective projection matrix and λ represent the scale factor.

Combining step 1 and step 2 the perspective projection of point P can be represented as $p = \lambda IEP$

Step 3) 2D-2D Transformation:

This transformation takes care of the conversion of scene units to pixels. In other words, it is a unit conversion from metric to pixel coordinates. If $(x, y, 1)$ represents a point p in the image plane using scenic units, then $(u, v, 1)$ represents the same point, p, in the image plane using pixels. The transformation from scenic units to pixel is achieved by using the scaling factors α_u and α_v. The equations are listed below:

$$u = \alpha_u x \tag{2.6}$$

$$\alpha_u = f k_u \tag{2.7}$$

$$v = \alpha_v y \tag{2.8}$$

$$\alpha_v = f k_v \tag{2.9}$$

In the Equations 2.7 and 2.9, k_u and k_v indicate the pixels per mm (millimeter) along the x and the y axis, respectively. The 2D to 2D transformation is completed when the origin of the image is shifted to the left corner of the image. The origin is denoted as u_0 and v_0. The 2D-2D transformation is affine in nature and is represented using the below given matrix

A,

$$A = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

Hence $(u,v,1)^T = A(x,y,z)^T$

We obtained three matrices from the three steps explained above. Combining the matrices E, I and A, we obtain the final matrix that represents the complete scene to image transformation. This matrix is represented as M. Matrix M represents a projective transformation from the 3D to 2D space. It is a 3×4 matrix. Therefore $M = AIE$. Matrix M can be written as shown in Equation 2.11 by ignoring the details of the individual matrix components.

$$M = \begin{bmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \end{bmatrix} \tag{2.11}$$

The prospective projection of a 3D point $P = (X,Y,Z,1)$ to the pixel coordinates $p = (u,v,1)$ is given by Equation 2.12, where λ is a non zero scale factor [1].

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.12}$$

The elements of the matrix M denote the extrinsic and intrinsic parameters of the camera, which are found out using Equation 2.12. This procedure is known as camera calibration and it is explained in detail in Section 2.2.2.

2.2.2 Camera Calibration Theory

Camera calibration is the method by which the extrinsic (external) and the intrinsic (internal) parameters of the camera are found out. The external parameters of the camera determine the pose and orientation of the camera relative to the world coordinate system.

The internal parameters of the camera determine certain properties of the camera, such as the focal length, image center, distortion coefficient, etc. As explained in the previous section the 3D point P must undergo a three step transformation in order to be transformed into a 2D point in the image reference frame. Camera calibration involves a process that estimates the projection matrix M. The elements of the matrix, M, give the internal and the external parameters of the camera [1]. 'Step 1) 3D Euclidean Transformation' explained in Section 2.2.1 leads to the formation of the matrix E, which consists of 6 extrinsic parameters of the camera. Also, 'step 3) 2D-2D Transformation' explained in Section 2.2.1 finds the four internal parameters of the camera according to the pin hole camera model. The estimation of matrix M starts with the process of adding two constraints to it, due to the fact that M is defined up to a scale factor. The two constraints are $m34 = 1$ and $m31^2 + m32^2 + m33^2 = 1$. Note that the first constraint does not hold true if $t_z = 0$ and the second constraint is non-linear, which implies that the last row of the rotation matrix has a norm equal to 1.

Equation 2.12 leads to the formation of two projection equations. The estimation of matrix M is dependent on solving these two projection equations, which give the relationship between the scenic points and corresponding pixels. The projection equations are listed below:

$$u = \frac{m11X + m12Y + m13Z + m14}{m31X + m32Y + m33Z + m34} \tag{2.13}$$

$$v = \frac{m21X + m22Y + m23Z + m24}{m31X + m32Y + m33Z + m34} \tag{2.14}$$

Estimation of the Equations 2.13 and 2.14 reveals that there are three sets of parameters in which two are known quantities and one is unknown. They are: (u,v) denotes the 2D pixel coordinates that are known, (X,Y,Z) denotes the known 3D scenic coordinates and the 12 entries of matrix M denotes the unknown quantities to be estimated. The 3D scenic points are known, as we use a calibration pattern similar to the one shown in Figure 2.3, whose geometric properties are known. The camera is set to the correct focus settings and the image of the calibration pattern is taken, from which the 2D points are extracted and matched with their corresponding 3D scenic points. These 2D and 3D points are used as inputs to find the elements of matrix M using Singular Value Decomposisition (SVD) method [1]. In this research work the camera calibration is done using the calibration

assistant tool, Halcon. The experimental procedure related to camera calibration will be explained in Section 4.1.1.

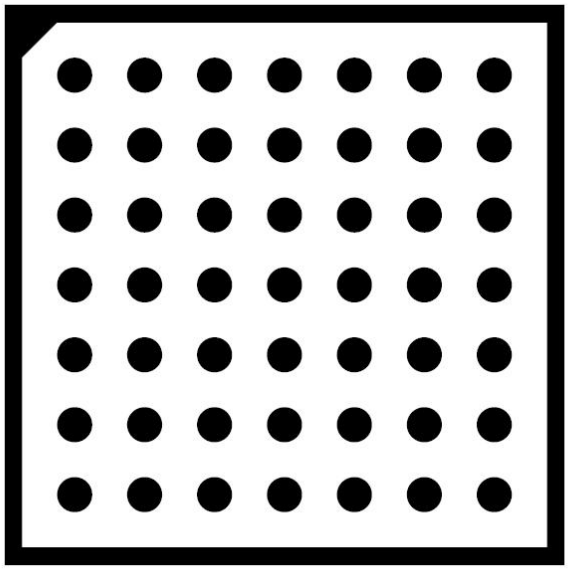

Figure 2.3: Calibration Pattern

Note that the calibration theory to find the external parameters, which has been explained in this section, holds true if the application involves only one camera. If there is a network of cameras involved, that is similar to that of this research work the procedure is different as explained in Section 2.3.2. In the case of multiple cameras, the geometric relationship between the cameras needs to be established. The internal calibration procedure remains the same regardless of the number of cameras used. Also, this camera calibration theory is derived from the pin hole camera model, which uses perspective projection.

2.2.3 Types of Camera Calibration

This research work involves three different types of camera calibration. The experimental step by step procedure for each of these calibration techniques is explained in Section 4.1. The different calibration techniques used are as listed below:

1) Internal calibration: This involves the computation of the internal parameters of each camera separately. This calibration technique is independent for each camera and is not

dependent on the network. Internal parameters represent the properties of the camera. The theory behind internal calibration was explained in the Section 2.2.2.

2) Hand-Eye calibration: The term Hand-Eye evolved from the field of robotic vision and it denotes the calibration of the camera with the robot. The robot can denote any system that helps to move objects like pan-tilt heads or multi-axis manipulators. There are two different kinds of hand-eye calibration based on the placement of the camera, which are moving camera hand-eye calibration (hand in eye calibration) and stationary camera hand-eye calibration (hand off eye calibration). The hand-eye calibration helps to find the relationship between the robot and the camera and the robot and the calibration target, respectively. In moving camera configuration the camera is mounted on the robot's end effector. In this configuration, the pose of the calibration target with respect to the base coordinate system of the robot, and the pose of the tool tip of the robot with respect to the camera coordinate system are determined [2]. The chain of homogeneous transformations that happen during this process is depicted in Figure 2.4.

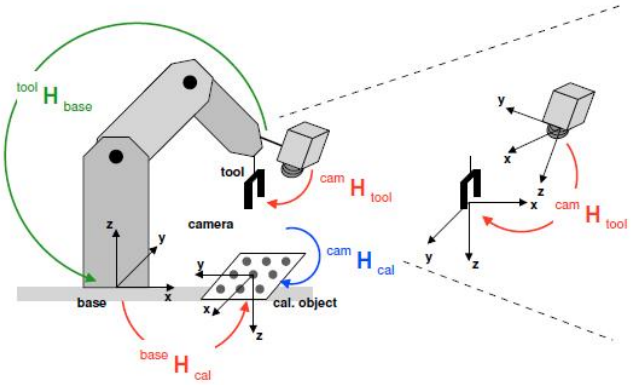

Figure 2.4: Chain of transformations for moving camera hand-eye calibration (Image courtesy of [2])

In the stationary hand-eye calibration the camera is fixed in a position and does not move. This configuration helps to find the pose of the robot's base with respect to the camera coordinate system and the pose of calibration target with respect to the robot's tool coordinate system. This research work deals with the moving camera hand-eye calibration and this steps involves only the camera that is mounted on the robot.

3) External calibration of the camera network: This technique uses a calibration graph. Calibration graph and its theory are explained in Section 2.3. This process is important in any camera network application as the pose of each camera with respect to a common

reference frame is vital information in order to perform any operation. The pose of each camera with respect to a common coordinate system is determined.

2.3 Graph Theory

The concepts of graph theory help to solve the issues in computer vision [27]. Some of the important terms of graph theory are discussed in this section. These terms will be used in the next section, which explains the calibration of a camera network. A graph is defined as a set of vertices connected using edges. The vertices are the nodes and the edges that denote the link between these nodes. In computer vision, the pixels or the interest points can represent the vertices and the spatial relationships between them are denoted by the edges. If the vertices are represented by cameras, then the graph can be termed calibration graph, as explained in Section 2.3.2. These edges can be directed or undirected based on the application. Digraphs are the directed graphs in which the edges are given directions. In a digraph each edge is an ordered set of vertices. If there is an edge between two vertices, vi and vj, they are termed to be adjacent, which would be denoted as $vi \backsim vj$. When the vertices of a path in the graph are distinct, except for the end vertices, the path is called a cycle. Graph with no cycle in it is called acyclic. A tree is a connected acyclic undirected graph. A graph is termed bipartite if it can be partitioned into two parts, A and B, such that every edge of the graph connects the vertices in A to the vertices in B. Thus, matching in bipartite graph defines a one to one relation between the vertices in A and vertices in B. An undirected graph is connected if there is a path between any two vertices of the graph. A digraph is strongly connected if every pair of vertices has a directed path between them. A connected graph has one connected component. In weighted graphs, a value is assigned to each edge. The shortest path between vertices is considered in terms of its length, where path length is given by the sum of all weights along the path.

2.3.1 Shortest Path Algorithm

The shortest path algorithm is used to find the shortest path distance between two vertices in a graph. This algorithm is used to resolve many optimization problems. The problem of finding the shortest path can be described in two ways. One method is to find the shortest path available from one single source vertex to all other vertices of the graph and the other is to find the shortest path available between every pair of vertices of the graph [27]. The

algorithms make use of the fact that the a sub path within the shortest path is also the shortest. Dijkstra's algorithm helps to find the shortest paths available from the source and assumes the weight of all the edges to be nonnegative. To compute the shortest path using negative edge weight is nondeterministic polynomial time (NP) hard. The shortest path among all pairs of vertices can be found using the Dijkstra's [28] algorithm recursively. The camera network calibration explained in Section 2.3.2 uses Dijkstra's algorithm.

2.3.2 Camera Network Calibration using graph theory

The concepts of graph theory are used to perform the multi camera calibration. A graph is said to be connected if a path exists from any point to any other point in the graph. The connected calibration graph is designed using the cameras and the target object at different orientations, as the vertices and edges denote the error of the associated poses. The calibration plate is used as the target object. The calibration target is placed at different orientations and the images are taken using the relevant cameras (the cameras in which the target appears in its Field of View). A subset of the camera network is formed for one particular orientation of the target taken. A graph is constructed with the camera nodes and their associated target poses. One of the camera vertices is chosen to take the image of the calibration target whose absolute position is measured. This image is termed as the reference image or the reference target. Using Dijkstra's algorithm [28], the shortest path from each camera vertex to the reference target is found. Once the shortest path is found, the pose composition is done to find the Pose of the source camera with respect to the reference target. The same procedure is repeated for all camera vertices, to find the pose of each camera with respect to the reference target, along with a maximum aggregated estimation error.

Figure 2.5 explains this calibration method with an example. One of the subsets of the cameras is represented by the cameras D, F and E with respect to the orientation Z of the calibration target. If P_{AW} denotes the pose of camera A with respect to the target orientation W, then according to the Figure 2.5 we have the following poses:

P_{AW},P_{BW},P_{CW},P_{BX},P_{DX},P_{CY},P_{EY},P_{DZ},P_{FZ},P_{EZ} and P_{ER}

The calibration graph corresponding to the organization of the cameras and the target's orientations in Figure 2.5 is represented in the Figure 2.6. The 'e' in Figure 2.6 denotes the edges, which represent the error of the associated poses. Here the reference target is R. Consider that we need to find the pose of camera C with respect to the reference target R.

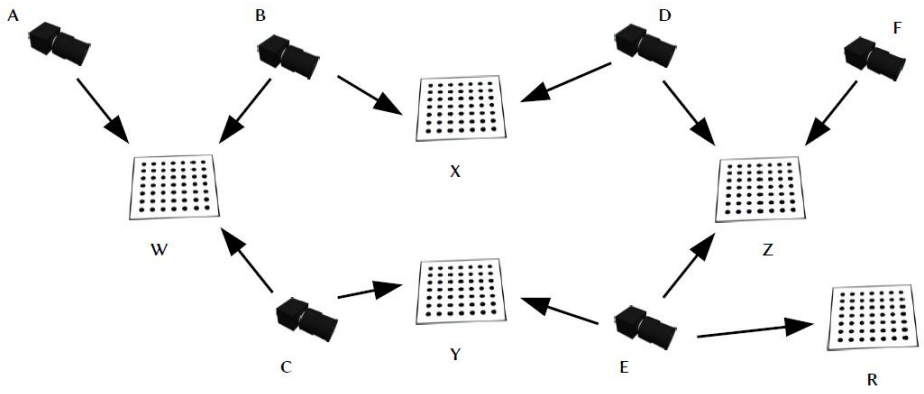

Figure 2.5: Multiple Camera Calibration

The shortest path will then be C, Y, E, R.

So $P_{CR} = P_{CY} \diamond {P_{EY}}^{-1} \diamond P_{ER}$, where the symbol $\diamond$ denotes pose composition.

This process is repeated for all camera vertices to find the shortest path that will finally lead to the solution P_{C_jR}, where C_j represents the camera.

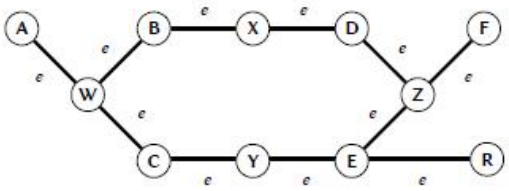

Figure 2.6: Calibration Graph

2.4 Coverage Strength Model

The coverage strength model is a modeling approach applied to the sensor system, which is either monocular or multiple [6, 7, 9]. The various components of the coverage model are discussed in this Section.

2.4.1 Visual Stimulus Space

The sensor coverage model requires a *stimulus space* to be defined in order to describe individual observable data [9]. A visual stimulus is localized to a point in three-dimensional space, and also has a direction (normal to the surface on which the point lies; the view angle). This assertion is reflective of the fact that most, if not all, vision tasks can be reduced to operations on sets of point features [29, 30] and, furthermore, are or can trivially be made invariant to rotations about the principal axis, i.e. in the image plane. We therefore define a *directional space* as the stimulus space for a system of vision sensors.

Definition 1 *The directional space* $\mathbb{D}^3 = \mathbb{R}^3 \times [0,\pi] \times [0,2\pi)$ *consists of three-dimensional Euclidean space plus direction, with elements of the form* $(p_x, p_y, p_z, p_\rho, p_\eta)$.

We term $\mathbf{p} \in \mathbb{D}^3$ a *directional point*. For convenience, we denote its spatial component $\mathbf{p}_s = (p_x, p_y, p_z)$ and its directional component $\mathbf{p}_d = (p_\rho, p_\eta)$.

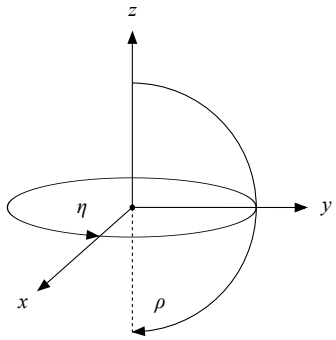

Figure 2.7: Axes and Angles of $\mathbb{D}^3$

A rigid 3D pose $P \in SE(3)$, represented by a rotation matrix $\mathbf{R}$ and translation vector $\mathbf{T}$, may be applied to $\mathbf{p} \in \mathbb{D}^3$. The spatial component is transformed as usual, i.e. $P(\mathbf{p}_s) = \mathbf{R}\mathbf{p}_s + \mathbf{T}$. The direction component is transformed as follows. If $\mathbf{d}$ is the unit vector in the direction of $\mathbf{p}_d$, then

$$P(\mathbf{p}_d) = \begin{bmatrix} \arccos([\mathbf{R}\mathbf{d}]_z) \\ \arctan 2([\mathbf{R}\mathbf{d}]_y, [\mathbf{R}\mathbf{d}]_x) \end{bmatrix} \tag{2.15}$$

2.4.2 Coverage Function

The coverage function assigns a bounded scalar measure of coverage strength to every point in $\mathbb{D}^3$.

Definition 2 *A coverage function is a mapping $C : \mathbb{D}^3 \rightarrow [0,1]$, for which $C(\mathbf{p})$, for any $\mathbf{p} \in \mathbb{D}^3$, is the strength of coverage at $\mathbf{p}$.*

Definition 3 *The set $\langle C \rangle = \{\mathbf{p} \in \mathbb{D}^3 | C(\mathbf{p}) > 0\}$ is the coverage hull of a coverage function C.*

In order for the coverage function to offer a useful gauge of sensor system performance, it requires the context of a task, which minimally includes a definition of a coverage objective over $\mathbb{D}^3$.

Definition 4 *A relevance function is a mapping $R : \mathbb{D}^3 \rightarrow [0,1]$, for which $R(\mathbf{p})$, for any $\mathbf{p} \in \mathbb{D}^3$, is the coverage priority at $\mathbf{p}$.*

Given coverage and/or relevance functions X_i and X_j, we define their union and intersection, respectively, as

$$X_i \cup X_j(\mathbf{p}) = \max(X_i(\mathbf{p}), X_j(\mathbf{p})) \quad (2.16)$$

$$X_i \cap X_j(\mathbf{p}) = \min(X_i(\mathbf{p}), X_j(\mathbf{p})) \quad (2.17)$$

for all $\mathbf{p} \in \mathbb{S}$. This, together with Definition 3, implies that $\langle X_i \cup X_j \rangle = \langle X_i \rangle \cup \langle X_j \rangle$ and $\langle X_i \cap X_j \rangle = \langle X_i \rangle \cap \langle X_j \rangle$.

2.4.3 Monocular Camera Coverage Model

The main components of the coverage strength model as described by Mavrinac et al. [9] are: Visibility , Resolution, Focus, Direction and Occlusion. They are explained in this Section. In defining the components of the single-camera model, we use a bounding function $B_{[0,1]}(x) = \min(\max(x,0),1)$ to limit x to the range $[0,1]$, which simplifies the formulation. We also let $\mathbf{p}' = P_E^{-1}(\mathbf{p})$, where $P_E : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is the external pose of the camera, and the transformation is applied to $\mathbf{p}$ as shown in Section 2.4.1 (thus, $\mathbf{p}'$ is $\mathbf{p}$ in the camera's coordinate system).

Given a task parameter γ indicating a margin in the image (in pixels) for full coverage, the horizontal and vertical cross-sections of the visibility component, C_V, are given by

$$C^{Vh}(\mathbf{p}) = B_{[0,1]}\left(\frac{1}{\gamma_h} \min\left(\frac{p'_x}{p'_z} + \tan\alpha_l, \tan\alpha_r - \frac{p'_x}{p'_z}\right)\right) \quad (2.18)$$

$$C^{Vv}(\mathbf{p}) = B_{[0,1]}\left(\frac{1}{\gamma_v}\min\left(\frac{p'_y}{p'_z} + \tan\alpha_t, \tan\alpha_b - \frac{p'_y}{p'_z}\right)\right) \tag{2.19}$$

for $\gamma > 0$, where α_l and α_r are the horizontal field of view angles, and α_t and α_b are the vertical field of view angles, as given by

$$\alpha_l = 2\arctan\frac{o_u s_u}{2f} \tag{2.20}$$

$$\alpha_r = 2\arctan\frac{(w-o_u)s_u}{2f} \tag{2.21}$$

$$\alpha_t = 2\arctan\frac{o_v s_v}{2f} \tag{2.22}$$

$$\alpha_b = 2\arctan\frac{(h-o_v)s_v}{2f} \tag{2.23}$$

where f is the focal length, s_u and s_v are the effective pixel dimensions in units of distance, o_u and o_v are the pixel coordinates of the principal point, and w and h are the image (sensor) width and height in pixels. The complete C_V is then given by

$$C^V(\mathbf{p}) = \begin{cases} \min(C^{Vh}(\mathbf{p}), C^{Vv}(\mathbf{p})) & \text{if } p'_z > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2.24}$$

Given task parameters R_{xi} (maximum ideal resolution), R_{xa} (maximum acceptable resolution), R_{ni} (minimum ideal resolution), and R_{na} (minimum acceptable resolution), the resolution component, C^R, is given by

$$C^R(\mathbf{p}) = B_{[01]}\left(\min\left(\tfrac{p'_z - z_R(R_{xa})}{z_R(R_{xi}) - z_R(R_{xa})}, \tfrac{z_R(R_{na}) - p'_z}{z_R(R_{na}) - z_R(R_{ni})}\right)\right) \tag{2.25}$$

for $R_{xi} > R_{xa}$ and $R_{ni} < R_{na}$, where

$$z_R(R) = R\min\left(\frac{w}{\tan\alpha_l + \tan\alpha_r}, \frac{h}{\tan\alpha_t + \tan\alpha_b}\right) \tag{2.26}$$

Given task parameters c_a and c_i, indicating the maximum acceptable and maximum ideal blur circle diameters, respectively, the focus component, C^F, is given by

$$C^F(\mathbf{p}) = B_{[0,1]}\left(\min\left(\frac{p'_z - z_n}{z_\triangleleft - z_n}, \frac{z_f - p'_z}{z_f - z_\triangleright}\right)\right) \tag{2.27}$$

for $c_a > c_i$, where $(z_\triangleleft, z_\triangleright)$ and (z_n, z_f) are the near and far limits of depth of field as given by (2.28), substituting blur circle diameters c_i and c_a, respectively, for c.

$$z = \frac{Afz_S}{Af \pm c(z_S - f)} \tag{2.28}$$

In the preceding equation, A is the effective aperture diameter and z_S is the subject distance. For many tasks, it is sensible for c_i to equal the physical pixel size, yielding the depth of field for perfect focus.

The direction (angle of view) component, C_D, is given by

$$C^D(\mathbf{p}) = B_{[0,1]}\left(\frac{\cos(\Lambda(\mathbf{p}')) - \cos\zeta_a}{\cos\zeta_i - \cos\zeta_a}\right) \tag{2.29}$$

where $\zeta_i, \zeta_a \in [0, \pi/2]$ are task parameters indicating the ideal and maximum view angles, respectively, and $\Lambda(\mathbf{p})$ is the angle of view relative to the ray from the camera's principal point, which can be calculated as

$$\Lambda(\mathbf{p}) = \cos^{-1}\left(\left[\sin p_\rho \cos p_\eta, \sin p_\rho \sin p_\eta, \cos p_\eta\right]^T \cdot \mathbf{p}_s\right). \tag{2.30}$$

Given a scene model S consisting of a set of triangles (which represent opaque faces of polyhedral objects in the scene), the point $\mathbf{p}_s$ is occluded iff the point of intersection between the line segment from $\mathbf{p}_s$ to the camera's principal point and any triangle in S[2] exists, is unique, and is not $\mathbf{p}_s$.

If $V : \mathbb{R}^3 \to \{0,1\}$ is a bivalent indicator function such that $V_i(\mathbf{p}_s) = 1$ iff $\mathbf{p}_s$ is not occluded from camera i's viewpoint, then the full coverage function is defined by

$$C_i(\mathbf{p}) = C_i^V(\mathbf{p})C_i^R(\mathbf{p})C_i^F(\mathbf{p})C_i^D(\mathbf{p})V_i(\mathbf{p}_s) \tag{2.31}$$

for all $\mathbf{p} \in \mathbb{D}^3$.

Precisely speaking, the coverage function notation $C_i(\mathbf{p})$ is shorthand for $C_i(\mathbf{p}, I_i, E_i, T, S)$,

[2] This intersection may be computed efficiently per Möller and Trumbore [31], for example. Furthermore, triangles in S which do not intersect with the (convex) viewing pyramid induced by C^V and the far limits of C^R and C^F, which can be determined efficiently using the method of separating axes [32], may be culled.

where

$$I_i = (f, s_u, s_v, o_u, o_v, w, h, A, z_S)$$
$$E_i = (x, y, z, \theta, \phi, \psi)$$
$$T = (\gamma, R_{xi}, R_{xa}, R_{ni}, R_{na}, c_i, c_a, \zeta_i, \zeta_a)$$

are the intrinsic, extrinsic, and task parameter vectors, and S is the scene model. Because not all tasks have requirements matching all the task parameters, the "default" permissive values for T are $\gamma = 0$, $R_{xi} = \infty$, $R_{xa} = \infty$, $R_{ni} = 0$, $R_{na} = 0$, $c_i = 1.0$, $c_a = \infty$, $\zeta_i = \frac{\pi}{2}$, and $\zeta_a = \frac{\pi}{2}$.

2.4.4 Discrete Model

Given any point $\mathbf{p} \in \mathbb{D}^3$ where $C(\mathbf{p}) = 1$, C^V, C^R, C^F, and C^D are monotonic nonincreasing functions over any half-space of $\mathbb{D}^3$ induced by a hyperplane through $\mathbf{p}$. Thus, in the absence of occlusions — that is, where $S = \emptyset$, or less restrictively, $V(\mathbf{p}_s) = 1$ for all $\mathbf{p}_s$ in the projection of $\langle C \rangle$ on $\mathbb{R}^3$ — $\langle C \rangle$ is a convex polytope. However, in general, V is not as well-behaved, and $\langle C \rangle$ is merely star-convex with respect to the camera's principal point. This complicates the computation of $\langle C_i \rangle \cap \langle C_j \rangle$, since, as shown by Tiwary [33], intersection of non-convex polytopes is NP-hard.

An arbitrarily close approximation can be achieved in the discrete domain. A coverage function C has a discrete counterpart denoted as $\dot{C}$, such that $\dot{C}(\mathbf{p}) = C(\mathbf{p})$ for all $\mathbf{p} \in \dot{\mathbb{D}}^3$, where $\dot{\mathbb{D}}^3$ is a discrete subset of $\mathbb{D}^3$. We denote the summation $\sum_{\mathbf{p} \in \dot{\mathbb{D}}^3} \dot{C}(\mathbf{p})$ as $|\dot{C}|$. Then, given $\dot{C}_i$ and $\dot{C}_j$ sampled over a common $\dot{\mathbb{D}}^3$, $\dot{C}_i \cap \dot{C}_j$ can be computed exhaustively.

A discrete relevance function $\dot{R}$ also allows computation of a bounded coverage performance metric for any coverage function C as

$$F(C, \dot{R}) = \frac{\sum_{\mathbf{p} \in \langle \dot{R} \rangle} C(\mathbf{p}) \dot{R}(\mathbf{p})}{\sum_{\mathbf{p} \in \langle \dot{R} \rangle} \dot{R}(\mathbf{p})} \tag{2.32}$$

where, by definition, $F(C, \dot{R}) \in [0, 1]$.

In our coverage model, $\dot{R}$ is the product of a task's *relevance model*, which is a set of points in $\mathbb{R}^3$ and/or directional points in $\mathbb{D}^3$, each with an associated relevance value in $[0, 1]$.

Defining a relevance point in $\mathbb{R}^3$ is a convenient shortcut for leaving out the view angle

criterion in C, which is frequently useful, rather than defining separate coverage functions. For $\mathbf{p}_s \in \mathbb{R}^3$, we redefine (2.31) as

$$C_i(\mathbf{p}_s) = C_i^V(\mathbf{p}_s)C_i^R(\mathbf{p}_s)C_i^F(\mathbf{p}_s)V_i(\mathbf{p}_s) \tag{2.33}$$

where the redefinitions of C^V, C^R, and C^F are trivial, as they are invariant to p_ρ and p_η.

Chapter 3

Scene Perception and Motion Planning Using Vision

3.1 General Solution

The general solution section explains the procedure used to obtain the robot's position in terms of its Joint angular positions using the pose given by the camera mounted on the robot's end effector. It is necessary to understand the pose convention used by the robot and the various terms involved with the syntax of the robot's pose. Hence, the Section 3.1.1 explains the syntax for pose and joint constants of the robot.

This section also explains the pose estimation and coverage value estimation based on pose data of the target and obstacle.

3.1.1 Pose Type of the Robot

The various pose type representations were discussed in Section 2.1.3. The robot expresses its pose in the coordinate system of its tool. Hence, its angular representation is in the form of Euler angles. The robot used for experimental purpose is Mitsubishi RV-1A. This is a 6 axis robot with six rotational joints. The most commonly used command to control the robot in this research work deals with two important constants. They are joint and position constants. The position constant denotes robot's pose. The robot's pose represent the pose of TCP (tool center point) with respect to the origin of the robot. The robot's pose always starts with the letter P when it is defined in MELFA IV BASIC. MELFA IV BASIC is a programming language used to control the robot and teach points for it to move. It helps to

move the end effector of the robot to the defined target positions. The MELFA IV BASIC file must end with an empty line. Similar to the pose constant the robot's movement can also be defined by using joint constant that has the angle of rotation with respect to each joint of the robot. The joint constant also starts with letter J. The syntax for both position and joint constants are explained in the below sections [3].

3.1.1.1 Position Constant

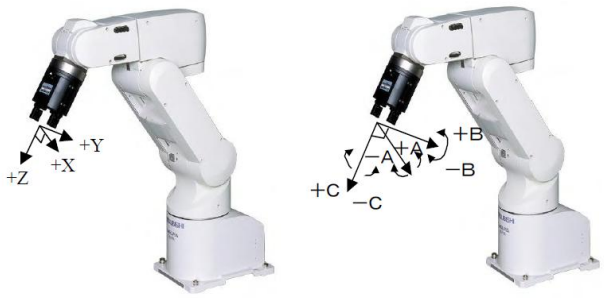

Figure 3.1: Tool Center Point pose of the robot (Image courtesy of [3])

The position constant cannot contain any variables. The syntax is :
P = (X,Y,Z,A,B,C,LI,L2)(FL1,FL2)
where,

- X,Y and Z indicate the coordinate data of the tool tip. Coordinate data is the position of the TCP with respect to the robot's base in mm (millimeter) units.
- A,B,C represent the posture data, which is the orientation of the robot's tool tip. A, B, and C are the robot's posture in the coordinate system of its hand's leading end (or flange center), each indicating a angle of rotation on the X axis, Y axis, and Z axis of the world coordinate system.
- L1, L2 are the additional axis data and are expressed either in mm (millimeter) or radians.
- FL1 represents the posture data. It is a binary number of length 7 whose last three digits provide the posture details.
- FL2 is Multi rotation data information. It has a default value of 0. It is a hexadecimal number that carries information for the eight axes.

If the structure Flags (FL1 and FL2) are omitted the default value of (7,0) is taken. The additional axis data is mostly given as 0.

The coordinate and posture data of the robot are indicated clearly in Figure 3.1

3.1.1.2 Joint Constant

The syntax for Joint constant is as follows:

J = (J1,J2,J3,J4,J5,J6,J7,J8)

J7 and J8 axis indicate the additional axis. The robot axis data are expressed in degree. Variables cannot be defined within this joint constant.

The robot's rotational joints are shown in Figure 3.2

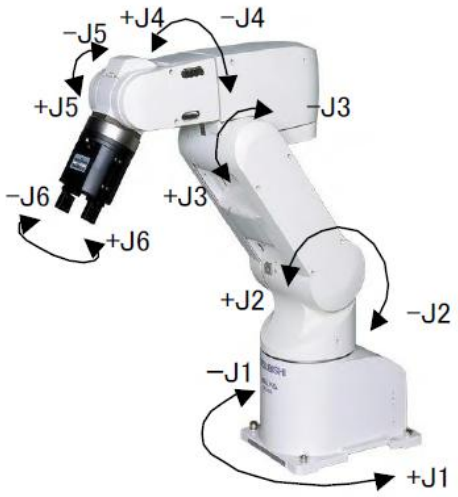

Figure 3.2: Joint Axis (Image courtesy of [3])

3.1.2 Robot's motion planning to reach the target

The robot's motion planning is achieved using the coverage strength value of the camera (cam A) mounted on the robot's end effector. Once C_A is greater than all other C_i values at a given time and if the occlusion is NULL, the robotic arm is instructed to move towards the target to perform a desired operation of an industrial robot. Here, C_A indicates the coverage strength value of camera A and C_i is the coverage strength value of all other cameras. The

robot is designed to move in a predefined set of movements at which camera *A* can cover the field of movement of the target within the workspace of the robot. Some of the important specifications of the Mitsubishi RV 1A robot are:

Pose Reliability - ± 0.02 mm (millimeter).

Arm Reachable radius - 418 mm (millimeter).

The arm reachable radius helps us to visualize the work envelope of the robot. The work envelope of the robot is discussed in Section 4.2.2.1 and is shown in Figure 4.7. A set of pose transformation and compositions are done in order to help the robotic arm to reach the target. The compositions are done using homogeneous transformation matrices. The series of pose compositions done are illustrated below:

$$(Pose_{tool\_camA}) \diamond (Pose_{target\_camA})^{-1} = (Pose_{tool\_target}) \tag{3.1}$$

$$(Pose_{tool\_base})^{-1} \diamond (Pose_{tool\_target}) = (Pose_{base\_target}) \tag{3.2}$$

$$(Pose_{base\_target})^{-1} = (Pose_{target\_base}) \tag{3.3}$$

The $Pose_{tool\_camA}$ is obtained from hand-eye moving camera calibration (please refer Section 4.1.2). The $Pose_{tool\_base}$ is obtained from the robot's controller. This denotes the robot pose.

Finally, $Pose_{target\_base}$ denotes the robot's pose used for reaching the target. The pose conversion to the robot's pose configuration should be done before transmitting the pose value to the robot. This pose value is converted to the joint configuration using the robot controller.

3.1.3 Estimation of the pose of the target and the obstacle

The pose of the target and the obstacle is estimated using Halcon machine vision library functions. The target has a standard calibration pattern, hence finding the pose of the target is based on extraction of the calibration marks. In this case the descriptor file of calibration pattern used is included. This file has details in relation to geometric properties of the calibration pattern such as length and breadth of the pattern, the diameter of circles in the pattern, distance between the circles in the pattern, etc. The calibration pattern resembles the pattern shown in Figure 2.3.

The pose of the obstacle is estimated by creating a model of it. The obstacle chosen is of rectangular in shape. The pose estimation of obstacle involves the creation of a descriptor

model for it. An image of the obstacle is taken by placing it along the Z-direction of the camera. In other words, the obstacle is placed perpendicular to the camera. This image is taken after the internal calibration of the camera. This image is fed as a template to create descriptor model [26]. The descriptor model could be created using any of these detector types: Lepetit, Harris or Harris _ binomial. This model will be used for planar calibrated matching. Once the model is created, the obstacle pose is found during the experiment by matching the descriptor model with that of image data. The internal parameters of the camera must be provided in order to succeed in the pose estimation of both the target as well as obstacle. A major advantage of this technique is that the pose estimation succeeds even when the object of interest is partially occluded by another object. Also, the model of obstacle must be created by each of the camera in the network before the start of the experiments. Model creation of the obstacle is one of the preparatory experiments to be done.

Sample output for pose estimation:

3D POSE PARAMETERS: rotation and translation

Rotation angles [degree] : r 141.029208993504 55.1707694587515 112.75893817857

Translation vector (x y z [m]): t -0.396276077712865 0.143798039521594 0.206699754969463

3.1.4 Coverage Strength Data Computation

The theory behind the coverage strength model is described in detail in Section 2.4. The coverage strength model creates a model of the experimental setup with the help of the calibration data. The data required to create the model is fed into a '.yaml'[1] file. For example, the YAML file consists of the following data with regards to this research work:

1) The various task parameters and their optimal values are listed. The values of the task parameters vary according to the application. These default values of these task parameters are listed at the end of the Section 2.4.3.

2) The cameras used to build the network are listed along with their internal parameters and the external parameters with reference to a common frame as explained in the calibration procedure of Section 4.1.

3) The scene data constitutes the next section of the YAML file. It has the pose details of all the elements that constitute the scene that includes the target and obstacle. The detection of occlusion is defined in Section 2.4.3.

[1] YAML is a human friendly data serialization standard for all programming languages.

4) The relevance data is provided in the form of geometric points that constitutes the boundaries of the target. Refer the Definition 4 in Section 2.4.2.
The coverage strength value is calculated for each of the camera in the network based on the data provided in the '.yaml' file. The pose of the target and obstacle must be updated to the coverage strength model for it to provide the accurate coverage value of each camera. The coverage strength value of the target is dependent on the obstacle being out of target's way. The '.yaml' file that was used for the experiments is shown in the Figure 4.9 and 4.10 in Section 4.4.1.

3.2 Block Diagram of the System

This section explains the important functional blocks and framework of the system.

3.2.1 Functional Block Diagram

The functional block diagram as shown in Figure3.3, describes the important functional blocks of the system, which helps to compute the coverage strength of each camera. The coverage strength computation plays a vital role in this system as it helps to decide on the camera to switch on and also to decide about the movement of the robot. The three main inputs provided are:

1) Calibration parameters

2) Task parameters

3) Pose estimates of the target and obstacle.

The calibration parameters (internal and external) and the task parameters are provided as a priori information to the coverage strength model. This priori information is recorded in the '.yaml' even before the start of the experiments. More information about the task parameters can be found in the Section 2.4.3 and the calibration parameters are explained in detail in Section 4.1. The pose estimates of the target and obstacle are updated to the model online during the execution of the experiments and there is no priori information about it. The procedure used to obtain the pose of the target and obstacle is explained in Section 3.1.3.

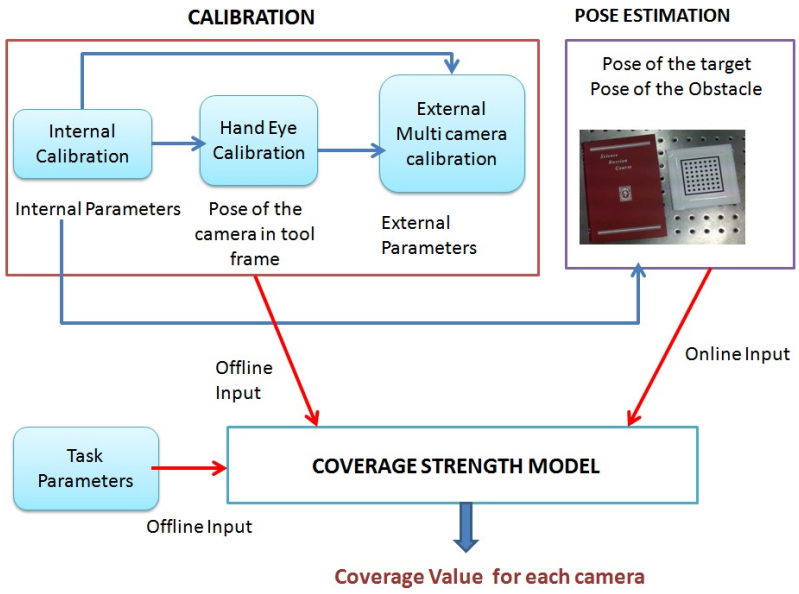

Figure 3.3: Functional Block diagram of the system

3.2.2 Framework of the System

This section explains the framework of the system using a simple block diagram. As shown in Figure 3.4 the framework shows the various entities of the system along with the networking of data that exists between them. The personal computer (PC) consists of the vision software (Halcon) and the coverage strength model. The robot consists of one camera on its end effector. The cameras are connected to the PC using the firewire ports and the controller of the robot is connected to the PC using RS232. The serial link communication between the robot and the vision system reduces the bandwidth of the system to a certain extent. The robot is connected to its controller. The program written in MELFA IV BASIC to control the serial communication and the robot's movement is copied to the controller from the PC using the 'MELFA PC Support Software'. The camera selected estimates the pose of the target and the obstacle using the vision software in the PC. The vision software updates the coverage strength model with the pose of the target and the obstacle and it accepts the "Best camera view" data from the coverage strength model. Based on the selection of the camera and the pose of the obstacle or target, the vision algorithm computes the pose of the robot. The computed robot pose is sent to the controller through serial communication. The controller converts the pose to joint values. The robot is commanded

to move to the joint values given by the controller. The robot also provides its current pose data to the vision software through serial communication.

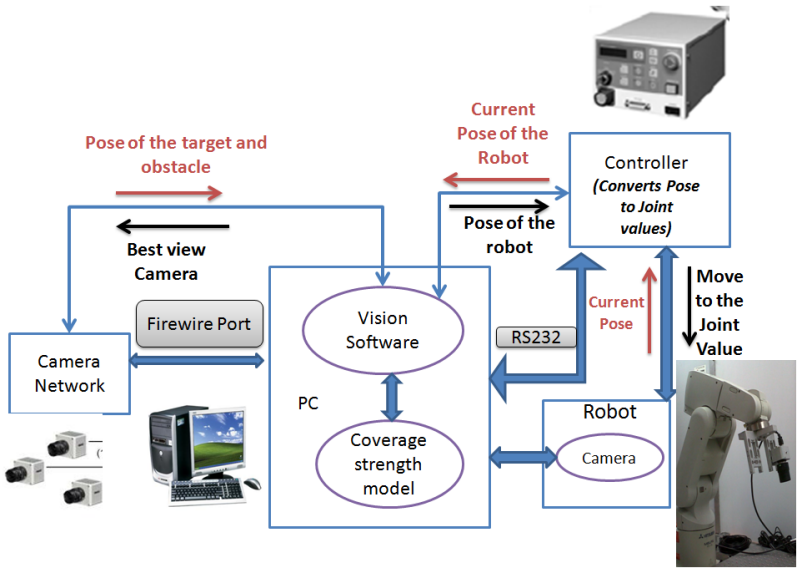

Figure 3.4: Framework of the system

3.3 Algorithm of the System

The algorithm for scene perception and robot motion planning is given below:

Step 1: Find the internal and external parameters of the camera network. This will be a preparatory phase for the experiments to be performed.

Step 2: Model the target using a relevance model and model the obstacle as an occlusion in the '.yaml' model file.

Step 3: Initialize the camera that has the best view of the target as camera A.

Step 4: Track the target pose P_{target} and the pose of the occluding object $P_{obstacle}$ using the camera selected with the best view .

Step 5: Update the coverage strength model with P_{target} and $P_{occlusion}$ values.

Step 6: Calculate C_i for all the sensors at any given time and the camera with the highest C_i value is chosen as the camera with best view.

Step 7: Move the robot to a predefined set of movements such that Camera A covers the work range of the target. It is always made sure that the predefined set of movements of the robot never collides with obstacle.

Step 8: Move the robotic arm towards the target *iff* C_A is chosen as the camera with the best view and the occlusion factor is null *else* move the robotic arm to another pose taht avoids collision with the obstacle.

Here C_i indicates the coverage strength value of the each camera. i indicates the camera. *A* indicates camera mounted on the robot. The image acquisition of the target and obstacle is always done by the camera with the highest coverage strength value. This image acquisition is necessary to update the pose of the target and occluding object to coverage strength model. The condition, "occlusion factor is null" can be checked in two ways. They are:
1) The non-existence of the obstacle pose given by the Halcon vision algorithm indicates that the "occlusion factor is null".
2) The coverage value of a camera given by the coverage strength model also depends on the fact that the obstacle is out of the way of the target. Hence, if the coverage value is equal to zero for a camera even if the target is in its field of view, it is a clear indication that "occlusion factor is not null".

Step 7 can be implemented by checking if there is an obstacle pose update even when camera *A* is not selected. If there is an obstacle pose, then the robotic arm is moved to a pose that does not cause collision. The check for the occlusion factor is robust enough as the pose of the obstacle can be found even if it is partially occluded by the target.

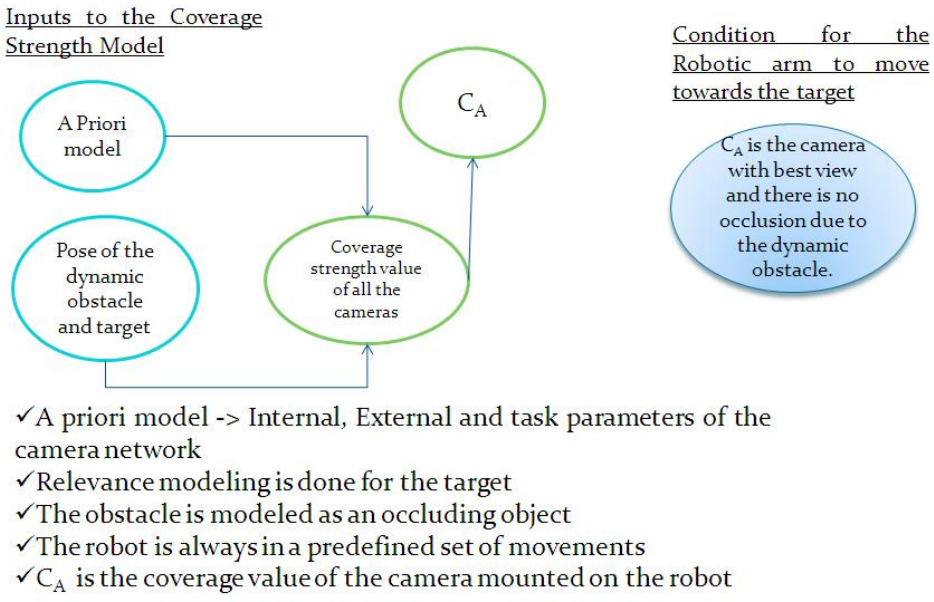

Figure 3.5: Simplified representation of the algorithm

The image acquisition of the target and the obstacle is always done by the camera with the highest coverage strength value. This image acquisition is necessary to update the pose of the target and the occluding object to the coverage strength model.

A simple representation of the algorithm is shown in Figure 3.5.

Chapter 4

Experiments and Analysis of Results

4.1 Preparatory Experiments

This section explains the calibration of the camera network with the robot. Camera calibration is a method by which the external and the internal parameters of the camera are found out. It involves the process to estimate the projection matrix of the camera. The internal parameters of the camera includes the focal length, radial distortion coefficient, image format and the optical center. The extrinsic parameters express the camera pose (Translational and rotational component) relative to a reference frame. The theoretical background for camera calibration is given in Section 2.2.2. The internal calibration of the camera is done using the calibration assistant of Halcon. This application involves a camera network of three cameras in which one of the camera is mounted on the end effector of the robot. Let us name the cameras as A, B and C. Camera A denotes the camera mounted on the robot's end effector. The step by step procedure to calibrate the network of cameras with the robot is given below:
Step 1) Calibrate all the cameras internally.
Step 2) The pose of the robot's tool centre point in the coordinate frame of the camera A is found out using the moving camera hand-eye calibration technique. This pose is denoted as $Pose_{tool\_camA}$.
Step 3) The external calibration of the camera network is implemented by applying Dijkstra's [28] algorithm on a connected calibration graph.

Step 3 finally provides the pose of each camera in the network with respect to one reference target frame whose absolute pose is known. Step1, Step 2 and Step 3 are explained

in detail in the Section 4.1.1, 4.1.2 and 4.1.3 respectively. The descriptor model for the obstacle is also created for all the cameras, after the completion of step 1. The procedure to create the obstacle descriptor model is explained in Section 3.1.3. Note that all the above three steps are implemented using Halcon machine vision libraries except that Dijkstra's algorithm is implemented separately in Python [1].

4.1.1 Internal Calibration of the camera network

The internal calibration is performed using the calibration assistant of Halcon machine vision libraries. According to the theoretical pin hole camera model, the 2D-2D Affine Transformation explained in the Section 2.2.1 finds the four internal parameters of the camera. The calibration assistant of Halcon helps to find the following internal parameters of a CCD camera [2]:
1) Focus $\longrightarrow$ Focal length of the lens.
2) Kappa $\longrightarrow$ Radial distortion coefficient.
3) Sx $\longrightarrow$ Scale Factor, Width of a cell on the CCD-chip.
4) Sy $\longrightarrow$ Scale Factor, Height of a cell on the CCD-chip.
5) Cx $\longrightarrow$ Column or X coordinate of the image center point (center of the radial distortion).
6) Cy $\longrightarrow$ Row or Y coordinate of the image center point (center of the radial distortion).
7) ImageWidth $\longrightarrow$ Width of the video images.
8) ImageHeight $\longrightarrow$ Height of the video images.

Apart from the above parameters it also provides the pose of the world coordinate system (target) with respect to the camera coordinate system. It is a 3D pose with 3 translational and 3 rotational parameters. This pose can be used to find the extrinsic parameters of a single camera.

The procedure to perform the internal calibration using calibration assistant of Halcon is as follows: Adjust the focus settings of the camera such that the target is in complete focus. Adjust the subject distance according to the need of the application. Also, adjust the aperture opening if required. Once the adjustments are made we need to make sure that the camera's settings are not disturbed. The settings must be the same during the entire experimental phase. If any of the camera setting changes during the course of the exper-

[1] Python is a programming language. Refer the link http://www.python.org/ for more information about python.
[2] Refer Section 4.2.2 for the specifications about the camera used

iment then the internal calibration procedure has to be repeated from the beginning. The calibration assistant of Halcon is opened. In the settings tab the descriptor file corresponding to the calibration target used is selected. The cell width and cell height of the sensor and focal length of the lens are also provided (This is in accordance with the data provided by the manufacturer). Details about the descriptor file is available in Section 3.1.3. In the calibration tab the Image acquisition is done. The calibration target is moved along the entire field of view of the camera and the images are captured for different orientations of the calibration target. 20 images are captured and then the button 'calibrate' is clicked in order to get the results. The results are stored as a '.cal' file as it is required for future experiments involving the pose estimation of the target, multiple camera calibration, etc. The Figure 4.1 shows the results of the internal calibration using the calibration assistant of Halcon.

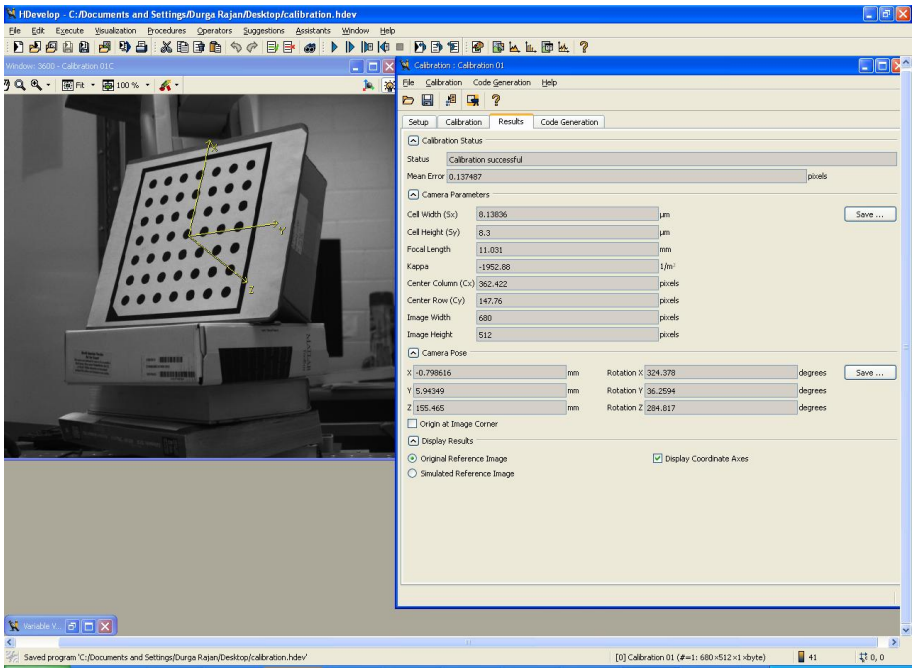

Figure 4.1: Calibration Assistant of Halcon machine vision library

4.1.2 Hand in Eye Calibration

The term hand-eye indicates the 3D pose of the robot (hand) relative to the camera (eye). The term robot here indicates any object that can move with a degree of freedom. There

are two different kinds of hand-eye calibration based on whether the camera is stationary or in movement. In this research work, camera A is mounted on the robot's end effector. Hence, the moving camera configuration is applied for the hand-eye calibration. The procedure explained in the remaining of this section is intended to find out two important poses. They are: pose of the calibration object in the robot's base coordinate system and pose of the robot tool in the camera $A's$ coordinates. The moving camera hand-eye calibration procedure involves grabbing the image of a stationary calibration plate at different robotic poses. The calibration plate is designed using Halcon. As described in the procedure of Section 4.1, the internal parameters of the camera A has to be given as an input to perform the hand-eye calibration. Apart from that, the robot poses used should also be given as an input along with the corresponding images taken. It is important to make sure that the pose format given by the robot is translated according to the format used by the hand-eye calibration procedure, this will be explained below in the Section 4.1.2.1. The input and the output parameters of this procedure are listed below:
Input parameters $\longrightarrow$ Images taken along with the corresponding robot poses and the internal parameters of the camera A.
Output parameters $\longrightarrow$ The pose of the calibration object in the robot's base coordinate system , the pose of the robot tool in the camera $A's$ coordinate and the error in the pose values obtained.

The pose of the robot tool in the camera $A's$ coordinate frame will remain constant all through the experiment. Hence, this value obtained from hand-eye calibration is an important input for the next Section 4.1.3 that deals with multi camera calibration. An image of the calibration target taken using the camera mounted on the camera's end effector during this hand-eye calibration process is shown in Figure 4.2

4.1.2.1 Pose Format Conversion for the robot

The pose values generated by Halcon consist of 7 parameters. The 7 parameters are: 3 translational parameters, 3 rotational parameters and the 7th parameter indicates the type of the pose. The type of the pose is decided using the following parameters: OrderOfTransform, OrderOfRotation, and ViewOfTransform. The possible combinations of these parameters values give 0 to 13 pose types. Halcon recommends to use only the representation types with OrderOfTransform = 'Rp+T' and ViewOfTransform = 'point' (codes 0, 2, and 4). [26]. Please refer the Figure 4.3 for the different pose types avaialable in Halcon.

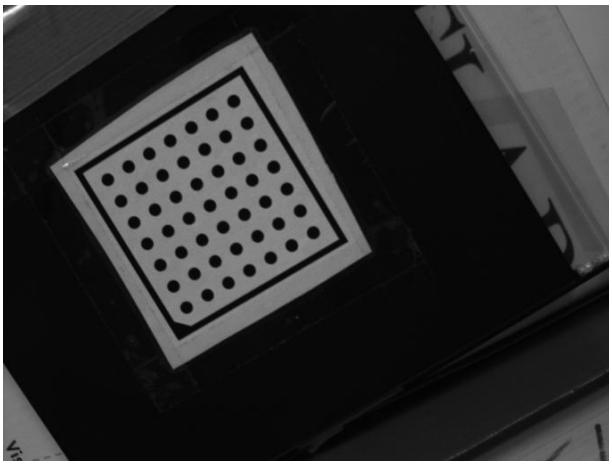

Figure 4.2: Image of the calibration target taken using the camera mounted on the robot during hand-eye calibration

OrderOfTransform	OrderOfRotation	ViewOfTransform	Code
'Rp+T'	'gba'	'point'	0
'Rp+T'	'abg'	'point'	2
'Rp+T'	'rodriguez'	'point'	4
'Rp+T'	'gba'	'coordinate_system'	1
'Rp+T'	'abg'	'coordinate_system'	3
'Rp+T'	'rodriguez'	'coordinate_system'	5
'R(p-T)'	'gba'	'point'	8
'R(p-T)'	'abg'	'point'	10
'R(p-T)'	'rodriguez'	'point'	12
'R(p-T)'	'gba'	'coordinate_system'	9
'R(p-T)'	'abg'	'coordinate_system'	11
'R(p-T)'	'rodriguez'	'coordinate_system'	13

Figure 4.3: Pose Types available in Halcon

In the values given for OrderOfTransform, R denotes the rotational matrix, T denotes the translational vector and p denotes the point to be transformed from one coordinate system to another. Similarly for OrderOfRotation: ‘g’ denotes rotation about the x-axis, ‘b’ denotes the rotation about y-axis and ‘a’ denotes the rotation about z axis.

During the moving camera hand-eye calibration and also during the movement of the robot’s end effector towards the target object, proper pose conversion between the robot’s pose type and the pose type used by Halcon must be performed. If the pose types do not match, it might lead to error in the calibration and in pose composition performed between the robot’s pose and the pose value estimated by Halcon. For example consider the following pose composition from Equation 3.2:

$(Pose_{tool\_base})^{-1} \diamond (Pose_{target\_tool})^{-1} = (Pose_{base\_target})$

Here $(Pose_{tool\_base})$ is obtained from the robot and $(Pose_{target\_tool})$ is obtained from Halcon. Both these poses must be of the same type for accurate results.

A series of experiments and calculations were done to find out the pose type of the robot as per the pose type convention formulated by Halcon. The pose type of the robot was found to be '1', as per Halcon's pose format. Pose type of 1 denotes the following: Order of transform is Rp+T , Order of rotation is 'gba' and view of transform is 'coordinate system.' Halcon uses the pose type '0' ('Rp+T' ,'gba','point') for all its results involving the pose. Hence, when a pose is obtained from the robot's controller the pose type should be converted to '0' before doing manipulations with the poses generated using Halcon vision libraries. Similarly when a pose is sent from Halcon to the robot's controller it must be converted to the pose type '1'. The translational part of the pose is expressed in mm (millimeter) for the robot whereas it is expressed in m (meter) for Halcon. The unit conversion for the translational part should also be done to avoid any error.

4.1.3 Multi Camera Calibration

The example explained in Section 2.3.2 gives the procedure to perform this calibration. The images of the target are taken using the Halcon's image acquisition interface and the pose of the camera with respect to the calibration target is found using the internal calibration method as explained in Section 4.1.1. Once the poses of the cameras are found, the absolute pose of the reference target is found by manual measurement from a fixed point in the world frame. If the reference target is attached to the tool tip of the robot, the pose of the robot gives the absolute pose of the target as this pose indicates the pose of the tool center point with respect to the robot's base coordinate system. Figure 4.4 shows the calibration target mounted to the robot's tool tip.

4.2 Apparatus

Apparatus includes both the hardware and the software components that were required to implement this research work.

The hardware components are:

- One 6 Axis Mitsubishi RV 1A robot with an end effector.
- Three Prosilica EC1350C CCD cameras.

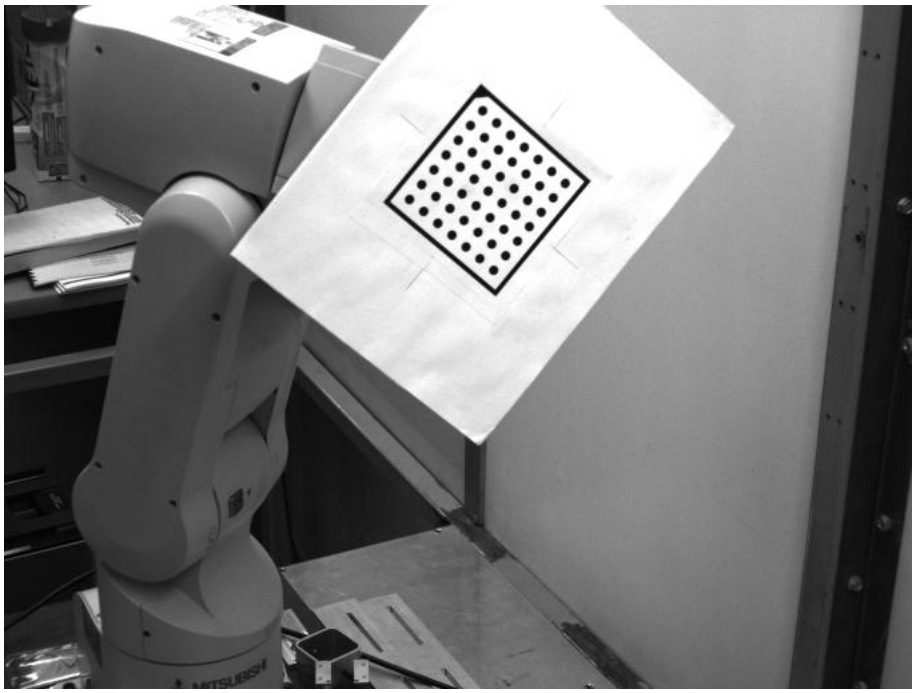

Figure 4.4: Target mounted on the robot's Tool tip

- Three Computar Lenses.
- One Robot Controller MELFA CR1-571.
- One Mitsubishi Teaching Pendant R28TB.
- One Personal Computer - Processor with the following specifications : Intel(R)Core (TM)2 Quad CPU Q6600 @2.40 GHz 2.39 GHz, 5.00 GB, 64 bit OS, Windows 7 Professional edition Service Pack 1.

The Vision Platform holds all the hardware components.

Software Components used are:

- Halcon Machine Vision Libraries.
- Adolphus Software [3].
- Python Programming language with YAML support.
- MELFA PC Support Software.
- MELFA IV BASIC programming language.

[3] Adolphus is free software licensed under the GNU General Public License. Full Python source code, documentation, and sample models are available at http://github.com/ezod/adolphus.

4.2.1 Experimental Setup

Experimental setup consists of a six axis Mitsubishi robot RV 1A with a network of three cameras A, B and C. Camera A is mounted on the end effector of the robot. The cameras used are Prosilica EC1350C. The cameras are Firewire cameras and are built using CCD sensors. The camera network and the robot is calibrated using three different methods as explained in Section 4.1. The camera calibration is performed using Halcon machine vision libraries.

The experimental setup in the vision platform is shown in Figure 4.5

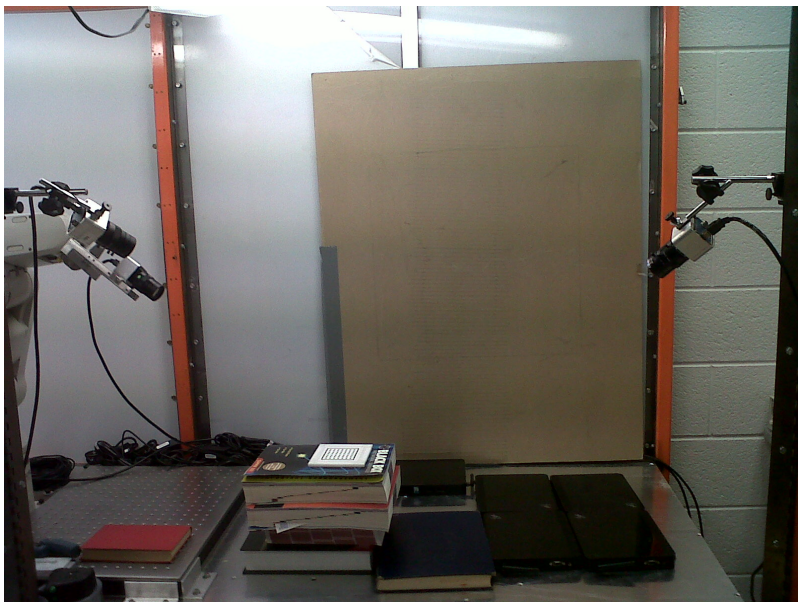

Figure 4.5: Experimental setup.

The calibration target used for the all the calibration methods is created using Halcon machine vision libraries. During the experimental procedure, the cameras must not be moved except camera A. It is also important that camera A is fixed to the end effector in other words, the pose the camera A with respect to the tool tip of the robot must remain constant during the entire experimental procedure. The communication between the robot and the PC-Processor is through RS-232 serial communication. The cameras are connected to the PC-Processor using firewire port.

4.2.2 Technical specifications of the robot and the camera

4.2.2.1 Technical specifications of the robot

The Mitsubishi RV 1A robot is compact and belongs to a class of powerful robots. These robots are best suited for component placement applications. High precision AC servo motors makes the driver system maintenance free. Additional axis could be added apart from the six axis. It is capable of multi tasking. Some of its specifications are [4] :

- Maximum composite speed - 2200 millimeter/second .
- Payload - 15 kg (kilogram).
- Position Repeatability - $\pm$ 0.02 .
- Weight - 19 kg (kilogram).
- Reach without hand - 418 mm (millimeter) .

The operating range of the joints is given below in degree:

- J1 $\rightarrow$ 300 (-150 to 150).
- J2 $\rightarrow$ 180 (-60 to 120).
- J3 $\rightarrow$ 95 (60 to 155).
- J4 $\rightarrow$ 320 (-160 to 160).
- J5 $\rightarrow$ 180 (-90 to 90).
- J6 $\rightarrow$ 400 (-200 to 200).

The operation range at section X-X is given in Figure 4.6

The work envelope of a robot shows all possible positions of the robot's tool tip. For a revolute robot this envelope is in the form of a sphere. Some points within the sphere is still not reachable by the robot due to the joint limits. This envelope helps to view the working limits of the robot graphically. Figure 4.7 shows the three joint working envelope of Mitsubishi RV 1A generated using the Workspace 5 software [34]. The three joint envelope shows the entire envelope of the robot created using 3 joints. The figure shows two spheres indicating the internal and external envelopes respectively.

Some of the major applications of Mitusbishi RV 1A are: assembly of electric machinery parts, adhesive application and sealing, keyboard inspection, inspection/testing of electronic parts, inspection/analysis of medical products, packaging of foods, etc.

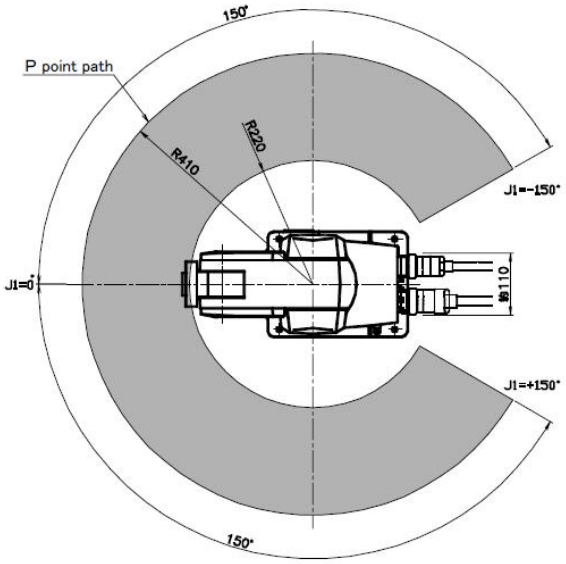

Figure 4.6: The operation range at section X-X (Image courtesy of [4])

4.2.2.2 Technical specifications of the camera and lens

The Prosilica EC1350C is a small, high performance, firewire camera operating at the speed of 18 frames per second. It is made of CCD sensor. The specifications of the CCD sensor is listed in Figure 4.8

The focal length of the Computar lens ranges between 12-36 mm(millimeter) with a maximum aperture ratio of 1:2.8 .

4.3 Experimental Procedure

The experimental procedure is based on the algorithm given in Section 3.3. The calibrated camera network is fixed. Camera *A* is fixed to the robot's end effector. Precaution must be taken to maintain the cameras in a fixed position throughout the experimental procedure. If any of the cameras is moved, the extrinsic parameters of all the cameras changes and this results in repeat of the entire calibration procedure. It is also important to maintain the settings of the lens in order to maintain the constant internal parameters of the camera.

The camera selection program is executed along with model file, (consists of coverage strength task parameter, the camera parameters, pose values of the scene components and

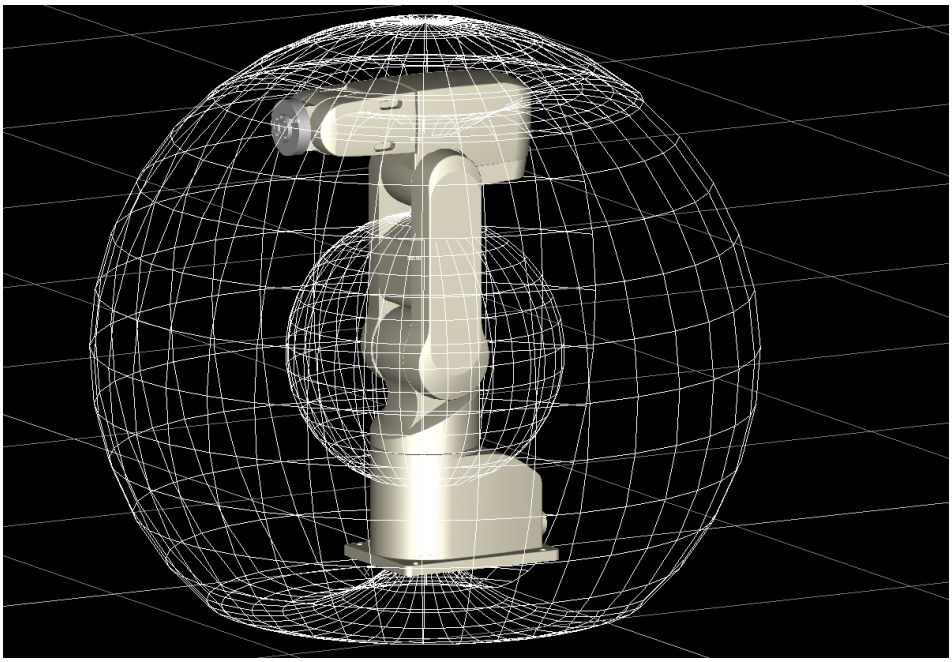

Figure 4.7: The work envelope for RV 1A

the relevance points of the target) which is in '.yaml' format. The target and the obstacle is in continuous steady movement similar to a pick and place industrial robot application. They could collide or have separate paths. The objective of this experiment is to move the robotic arm to a safe position. Once the camera selection program is ready to be executed, the Halcon motion planner program is executed. The robot is programmed using MELFA BASIC IV to move to a safe pose and the safe pose data is provided to it by the Halcon motion planner program. One of the camera's of the network, for example camera C is initialized to acquire images. Once the target enters the field of View of camera C, the target pose is updated to the camera selection program that uses the coverage strength model to calculate the coverage values of each camera. Consider the coverage values of camera A, B and C to be C_A= 0.5 , C_B = 0.75 and C_C = 0.2 respectively. The camera selection program selects the camera B as it has the highest coverage strength value. Camera B begins to acquire the images and the process repeats. When camera A is selected it is checked if there is any occlusion pose updation to the coverage strength model. If there is no such updation, then the robot's end effector is programmed to reach the target through a series of homogeneous pose transformations, else the the robotic arm is moved to a safe pose that does not collide with the obstacle. The robot is also continuously moved to a series

Sensor Type	Sony ICX205AL CCD (ICX205AK for color)
Sensor Shutter Type	Progressive Interline
Sensor Resolution	1360 x 1024 pixels
Pixel Size	4.65µm x 4.65µm
Optical Format	1/2 inch
Lens Mount	C-mount with adjustable back focus
Color Sensor Filter Pattern†	Bayer
Full Resolution Frame Rate	18.5 fps
Frame Rate (640 x 480 ROI)	30.9 fps
Power Requirements	Less than 2.5W††
	8V to 40V as per IEEE1394A specification†††
Digitization	12 Bits
Trigger latency	2µs
Trigger Jitter	±0.5µs
Operating Temperature	0 to 50 Celsius
Operating Humidity	20 to 80% non-condensing
Size	33mm (height) x 46mm (width) x 56mm (length)
Weight	84g
Interface Standard	IEEE 1394A (FIREWIRE™)

Figure 4.8: Specifications of the CCD sensor (Image courtesy of [5])

of positions that can cover its workspace and the field of view of camera *A*. It is made sure that the predefined set of movements of the robot never collides with the obstacle even when camera *A* is not selected .

4.4 Results and Analysis

The experiments are conducted on the vision platform. Some of the results obtained are discussed in this section.

4.4.1 Test 1

After the preparatory experiments were completed, the calibration parameters of the camera network were loaded into the '.yaml' file. The '.yaml' file also has the other parameters needed as described in Section 3.1.4. The '.yaml' file used is shown in the Figure 4.9 and 4.10.

The camera selection software was executed. The target and the obstacle were moved in various orientations possible and the cameras were selected according to the movement

of the target. Note that, camera $A \Rightarrow 99B$, camera $B \Rightarrow ABB$, camera $C \Rightarrow 9B3$. The output received from the coverage strength is given below:

Received Pose((404.1522, 234.7889, 219.3171), Rotation((-0.11, (-0.6451, -0 .1452, 0.7428)))) from camera ABB.
'ABB': 0.4745458398761846, '9B3': 0.0655008852129392, '99B': 0.1957 7700265603426
Best view is camera ABB.

Received Pose((403.5857, 234.2049, 216.2835), Rotation((-0.11, (-0.6453, -0 .1438, 0.7428)))) from camera ABB.
'ABB': 0.47499634947335256, '9B3': 0.06683608542113018, '99B': 0.19 665577770957626
Best view is camera ABB.

Received Pose((408.3109, 224.5176, 216.2989), Rotation((-0.10, (-0.6486, -0 .1397, 0.7418)))) from camera ABB.
'ABB': 0.9302372997910228, '9B3': 0.07549342515516985, '99B': 0.213 6430256301476
Best view is camera ABB.

Received pose Pose((415.1886, 208.9215, 216.6219), Rotation((-0.09, (-0.6548, -0 .1342, 0.7387)))) from camera ABB.
'ABB': 0.9641611997342434, '9B3': 0.0886157542359513 3, '99B': 0.24213363222339196
Best view is camera ABB.

Received Pose((420.7128, 193.5022, 216.9094), Rotation((-0.08, (-0.6635, -0 .1300, 0.7329)))) from camera ABB.
'ABB': 0.9973268334692935, '9B3': 0.100160045663883 12, '99B': 0.2736265161953803
Best view is camera ABB.

Received Pose((424.4441, 180.8432, 217.2108), Rotation((-0.08, (-0.6723, -0 .1281, 0.7259)))) from camera ABB.
'ABB': 1.0, '9B3': 0.10843988528039945, '99B': 0.30 20602748992918
Best view is camera ABB.

Received Pose((425.1358, 175.7675, 218.2912), Rotation((-0.07, (-0.6779, -0 .1292, 0.7215)))) from camera ABB.
'ABB': 1.0, '9B3': 0.1142528472037391, '99B': 0.322 13614564995313
Best view is camera ABB.

Received Pose((418.5265, 0.4981, 301.6184), Rotation((-0.64, (0.5805, -0.09 38, -0.5002)))) from camera 9B3.
'ABB': 0.0, '9B3': 0.37886242724015146, '99B': 0.20940239390718648

Best view is camera 9B3.

Received Pose((414.8180, 5.0542, 306.3871), Rotation((-0.61, (0.6051, -0.11 70, -0.4952))))
from camera 9B3.
'ABB': 0.0, '9B3': 0.3768535858143348, '99B': 0.2522978801634315
Best view is camera 9B3.

Received Pose((293.1630, 94.9148, 556.9753), Rotation((0.35, (-0.7581, -0.1 277, 0.5377))))
from camera 9B3.
'ABB': 0.0, '9B3': 0.40873353474520846, '99B': 0.75
Best view is camera 99B.

The output consists of three parts, they are:

1. **Received Pose**: It is the target pose with respect to the camera with the best view. Example: *Received Pose((404.1522, 234.7889, 219.3171), Rotation((-0.11, (-0.6451, -0 .1452, 0.7428)))) from camera ABB*.
2. **Coverage strength value:** Coverage strength value of all the cameras in the network. Example: *'ABB': 0.0, '9B3': 0.37886242724015146, '99B': 0.20940239390718648*.
3. **Best View:** Camera selected with the best view as per the coverage strength values obtained. Example: *Best view is camera 99B*.

If the above output is observed clearly , the shift in view from one camera to another based on the coverage strength model can be identified. Figure 4.11 shows the method in which the preliminary tests were conducted for camera selection.

Figure 4.12 shows a screen shot of the camera selection result obtained.

The performance metrics of the coverage strength model for view selection application is already discussed in section IV of [9]

4.4.2 Test 2

After doing the initial set of experiments to implement the camera selection, the experiments dealing with the motion planning of the robotic arm was conducted. The obstacle avoidance phenomenon is also considered here. The obstacle pose is continuously updated to the model. Figure 4.13 shows the pose estimation of the obstacle. The coordinate axis in red indicates the obstacle. As shown in Figure 4.14 coordinate axis in green indicates the target pose.

Hence the camera selected can have the following possibilities:

a) It can detect the target's pose

b) It can detect the obstacle pose

c)It can detect both target and obstacle poses.

If the camera selected is not 'A' (camera mounted on the robot), then the possibility (a) need not be investigated further as the robotic arm will be moved towards the target only if the selected camera is 'A'. But in (b) and (c) the obstacle pose exists. In this case irrespective of the camera selected, it has to be made sure that the robotic arm is in a safe position with no collision with the obstacle.

If the camera selected is 'A'(99B), then the following conditions are checked based on the a,b and c possibilities listed above.

i) if(camera = '99B'and obstacle pose *equals* NULL)

ii) if(camera equals '99B'and obstacle pose *not equals* NULL)

Condition (i) states that the camera selected is '99B' and there is no obstacle, hence the robotic arm can be moved towards the target. The pose transformations required to compute the pose of the robot's tool tip with respect to the target is calculated over here.

Condition (ii) states that the camera selected is '99B' and there is an obstacle, hence the robot is moved to a pose that does not allow the arm to reach the target. As the application imagined is pick and place, both the target and obstacle move slowly and steadily downwards towards the table. Figure 4.13 and 4.14 clearly shows that the target and the obstacle are always on the table of the platform similar to a pick and place application in which the targets move steadily on a conveyer belt.

An example program written using MELFA IV BASIC, that was used to move the robotic arm according to the input given by the vision algorithm in Halcon is shown in Figure 4.15

As observed in the program in Figure 4.15 the "OPEN" refers to the establishment of the serial communication between the robot and the computer loaded with the vision algorithms.

Some of the results obtained are shown in the figures described in the following paragraphs.

Figure 4.16 and 4.17 were captured at the same time. Only difference is that Figure 4.16 is a real time depiction of the occluded target during the experiments done and Figure 4.17 shows the corresponding pose estimation output of the obstacle from the vision algorithm.

Figure 4.17 and 4.18 both indicate the existence of an obstacle and the selected cam is 'A' (99B). Hence they fall under condition (ii), which implies that the robotic arm is not

moved towards the target.

Figure 4.19 shows the movement of the robotic arm towards the target as the camera selected is A and, there is no obstacle encountered in the target's path.

Figure 4.20 shows that the robotic arm moves away from the target and is in a safe distance, when there is occlusion of the target due to the obstacle.

The following graphs show the results obtained for a specified number of iterations. Each iteration indicates a frame or an image. A set of images were taken and the results were plotted as explained in the following paragraphs.

Figure 4.21,4.22 and 4.23 represent the motion of the robot along the x,y and z direction for ten iterations. This motion of the robot is computed based on the initial and final pose of the robot. The initial pose represents the pose of the robot before the detection of the target pose by camera *A* and the final pose represents the movement of the robot's end effector towards the target after the target pose has been detected by camera *A*. Its important to note that the robot moves towards the final pose only when the required conditions are satisfied (the selected camera is *A* and the occlusion factor is NULL). Figure 4.24 shows the target positions obtained along the x,y and z direction based on which the end effector is moved towards it. It also gives the root mean square error associated with the target's pose in pixels. So the final pose of the robot traced as shown in Figure 4.21,4.22 and 4.23, is calculated based on the target poses shown in Figure 4.24. The final pose is computed based on the set of pose composition equations listed in Section 3.1.2. The success rate for the end effector of the robot to reach the target efficiently with proper orientation is 8 out of the ten iterations.

Figure 4.25 shows the position of the obstacle obtained along the x, y and z direction with respect to the base of the robot and also the position of the robot, for twenty four iterations. So the end effector is moved to a pose that does not collide with the obstacle. It is clearly observable from the graph that the robot motion is in a safe position for the set of obstacle poses taken. Hence this provides a success rate of 100 percent.

C:\repositories\visionplatform\yaml\ICIRA.yaml April 27, 2012 4:33 AM

```
model:
    name:          Experiment 1

    res_min_acceptable:     0.9
    blur_max_acceptable:    10.0
    angle_max_ideal:        0.2
    angle_max_acceptable:   1.3

    cameras:
        - name:         ABB
          sprites:      [cameras/prosilicaec1350.yaml, lenses/computarm3z1228cmp.yaml]
          A:            4.470000
          f:            11.850477
          s:            [0.009295, 0.009300]
          o:            [331.170725, 267.858839]
          dim:          [680, 512]
          zS:           550.000000
          mount:        CameraFrame
          pose:
              T:            [-443.73071872240814, -424.67563356965564, 651.374849939946]
              R:            [-0.5704482898973338, [0.5008502387853137, -0.6006045055366682,
              -0.251021940643641]]
              Rformat:      quaternion
        - name:         9B3
          sprites:      [cameras/prosilicaec1350.yaml, lenses/computarm3z1228cmp.yaml]
          A:            4.470000
          f:            11.603908
          s:            [0.009272, 0.009300]
          o:            [364.956143, 243.341520]
          dim:          [680, 512]
          zS:           550.000000
          mount:        CameraFrame
          pose:
              T:            [74.82018971901279, 366.5383167106949, 94.81127055283334]
              R:            [-0.34444348076839504, [-0.04934750867585558, 0.3334867790548703,
              -0.8761906645874233]]
              Rformat:      quaternion
        - name:         99B
          sprites:      [cameras/prosilicaec1350.yaml, lenses/computarm3z1228cmp.yaml]
          A:            4.470000
          f:            11.902109
          s:            [0.009284, 0.009300]
          o:            [327.643978, 225.995933]
          dim:          [680, 512]
          zS:           550.000000
          mount:        CameraFrame
          pose:
              T:            [39.2636212, -71.6650996, 59.3008429]
              R:            [0.7431628852068535, [-0.18112317525571, -0.02874016403462481,
              0.64348840277039961]]
              Rformat:      quaternion
```

-1-

Figure 4.9: Model File I

```
    scene:
        - name:         VisionPlatform
          sprites:      [scene/visionplatform.yaml]
          pose:
              T:            [-978, -455, -553]
              R:            [0, 0, 0]
              Rformat:      'euler-zyx-deg'

        - name:         CameraFrame
          mount:        CameraFrameFrame
          pose:
              T:            [0, 0, 0]
              R:            [180, [1, 0, 0]]
              Rformat:      'axis-angle-deg'
        - name:         CameraFrameFrame
          pose:
              T:            [469.5889, -17.9072, 350.0681]
              R:            [-0.68, [-0.1327, -0.7171, 0.0843]]
              Rformat:      'quaternion'
        - name:         CalibrationPlate
          sprites:      [scene/calibrationplate.yaml]
          mount:        CalibrationPlateFrame
          pose:
              T:            [0, 0, 0]
              R:            [0, 90, 0]
              Rformat:      'euler-zyx-deg'
        - name:         CalibrationPlateFrame
        - name:         obstacle
          sprites:      [scene/obstacle.yaml]
          mount:        Obstacleframe
          pose:
              T:            [0, 0, 0]
              R:            [0, 90, 0]
              Rformat:      'euler-zyx-deg'
        - name:         Obstacleframe

relevance:
    - name:         target
      mount:        CalibrationPlateFrame
      points:
        - point:        [70, 70, 0, 3.141593, 0]
        - point:        [70, -70, 0, 3.141593, 0]
        - point:        [-70, -70, 0, 3.141593, 0]
        - point:        [-70, 70, 0, 3.141593, 0]
```


Figure 4.10: Model File I contd..

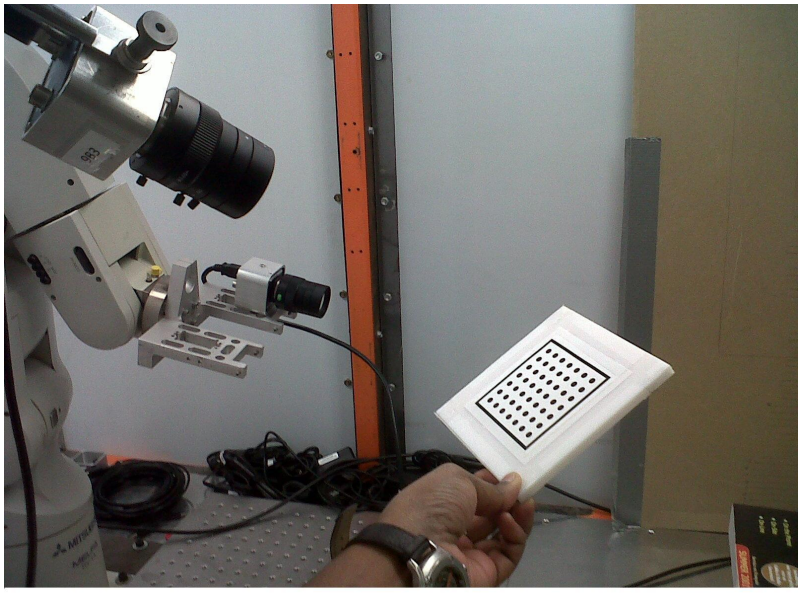

Figure 4.11: Preliminary test for the camera selection experiment

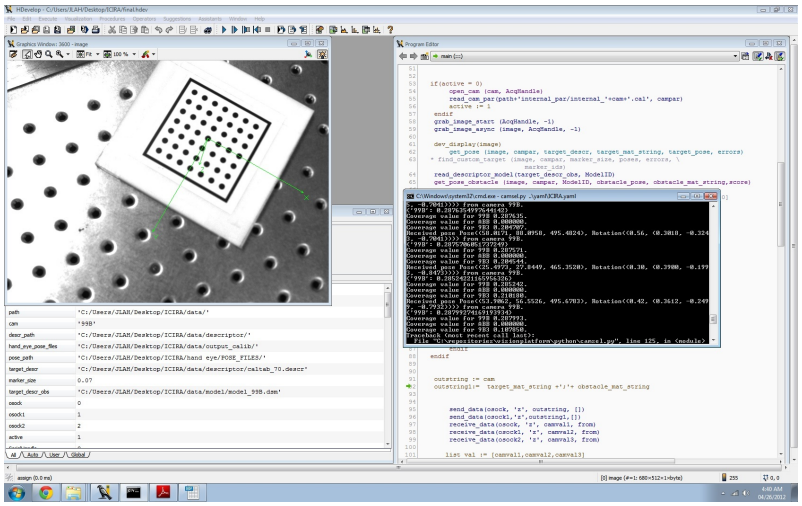

Figure 4.12: Camera Selection results

Figure 4.13: Obstacle Pose

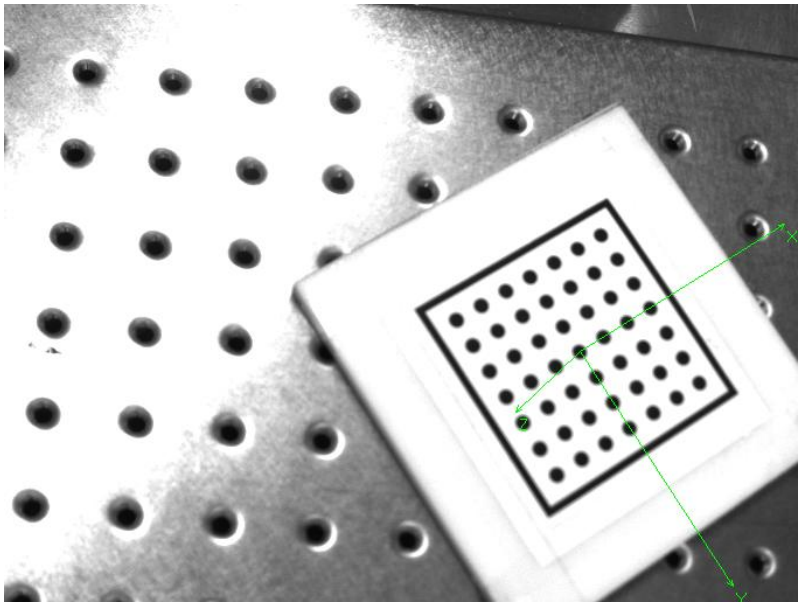

Figure 4.14: Target Pose

D:\RA_my work\ICIRA\WRI.prg April 27, 2012 10:47 AM

```
10 OPEN "COM1:" AS #1
15 OPEN "COM1:" AS #2
20 M2=1
25 INPUT #1,P1
30 PRINT #1,M2
35 J1=PTOJ(P1)
40 MOV J1
45 GOSUB 1060
50 M1=POSCQ(J2)
55 IF M1=1 THEN GOSUB 500 ELSE GOTO 60
60 END
500 MOV J2
505 RETURN
1060 INPUT #2,P2
1065 PRINT #2,M2
1070 J2=PTOJ(P2)
1075 RETURN
```

-1-

Figure 4.15: MELFA IV BASIC program to communicate with the robot

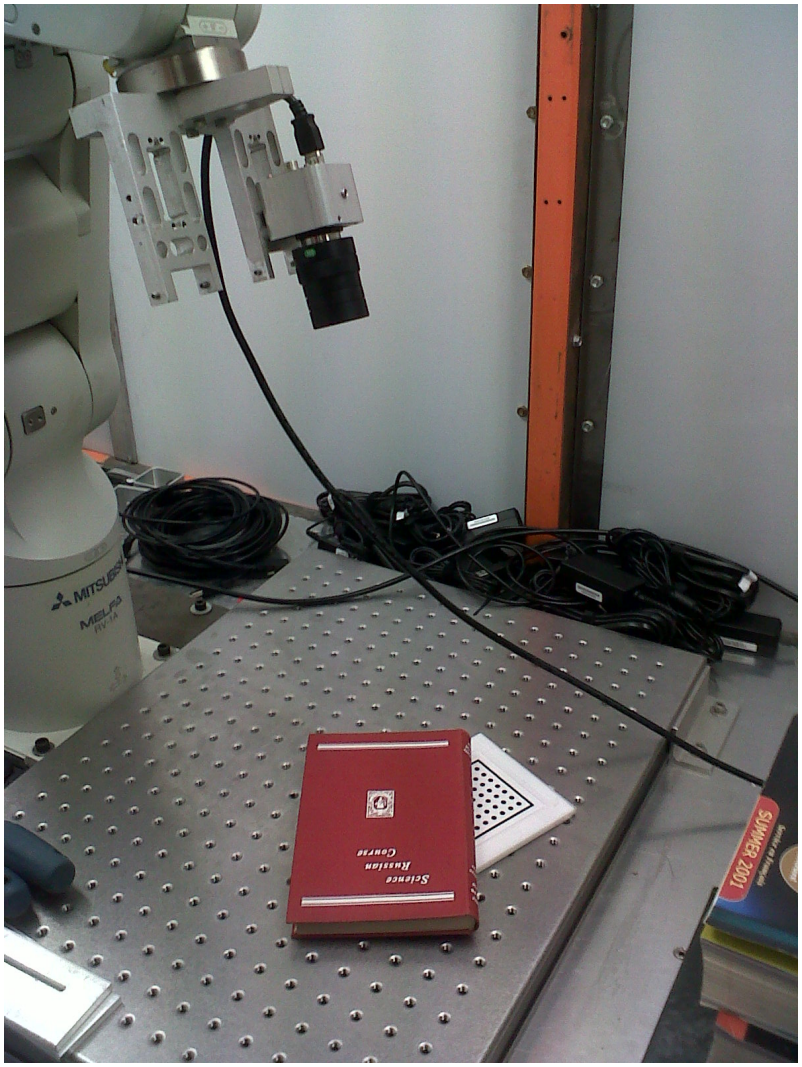

Figure 4.16: Target occluded by the obstacle in the vision platform

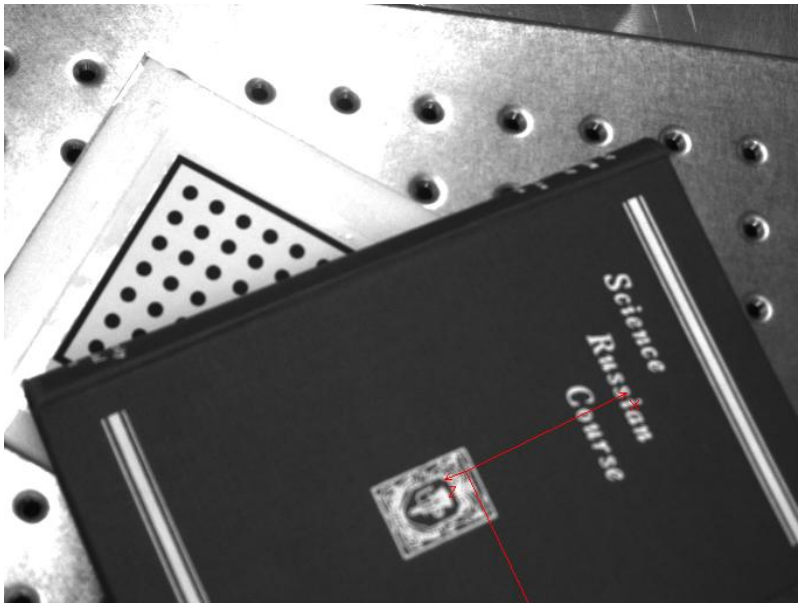

Figure 4.17: Obstacle pose updated to the model

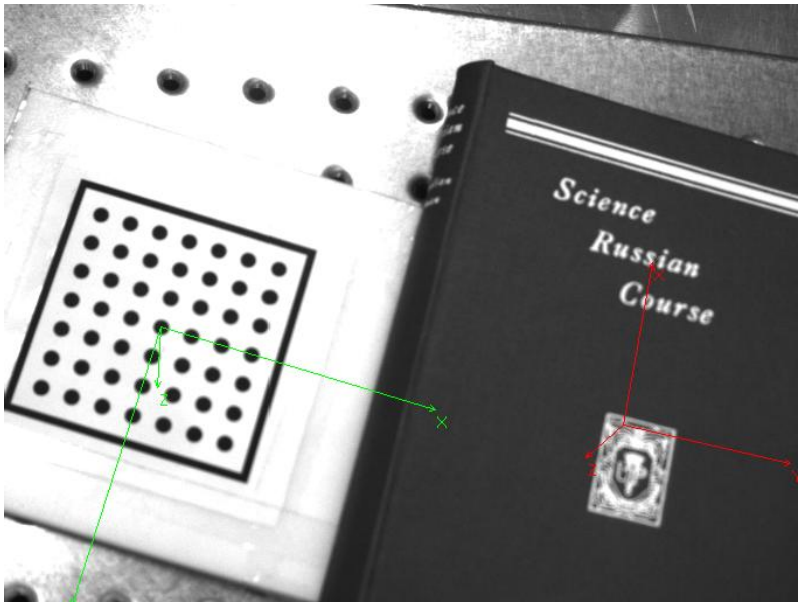

Figure 4.18: Simultaneous Obstacle pose and target pose

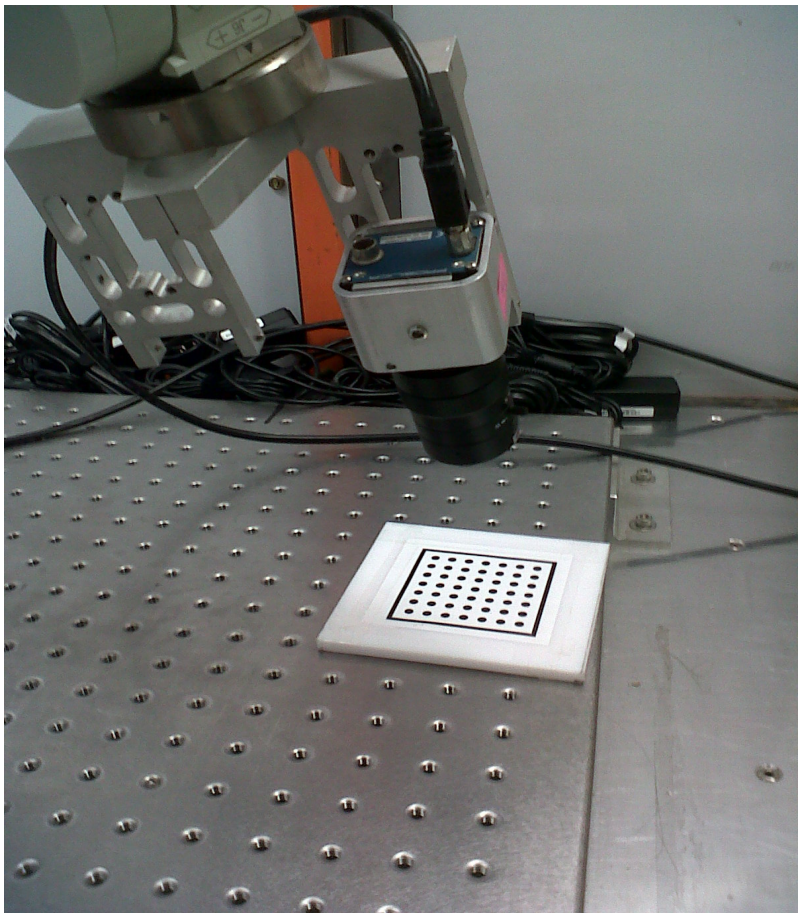

Figure 4.19: Robotic arm in movement towards the target

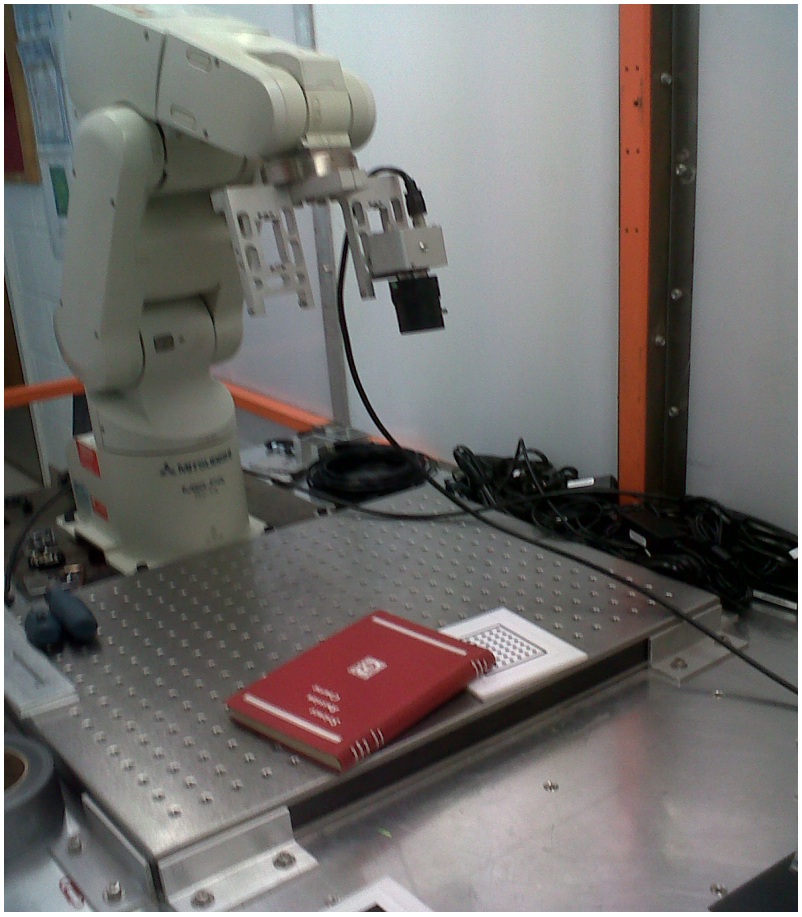

Figure 4.20: Robotic arm moving away from the target

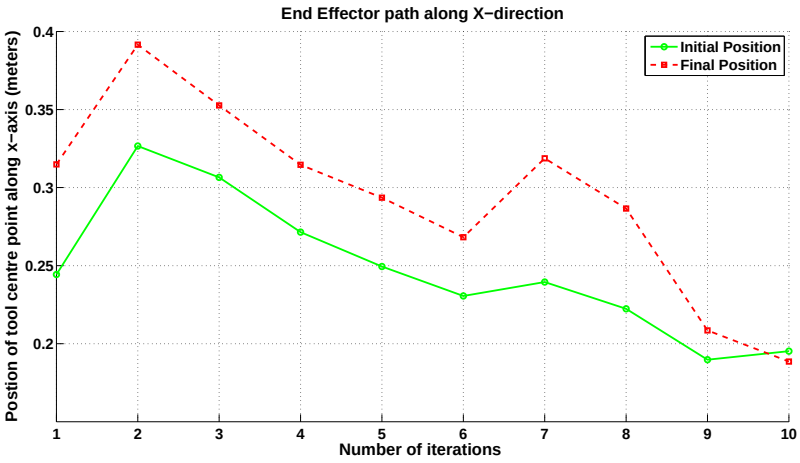

Figure 4.21: Robot's motion along the x-axis direction

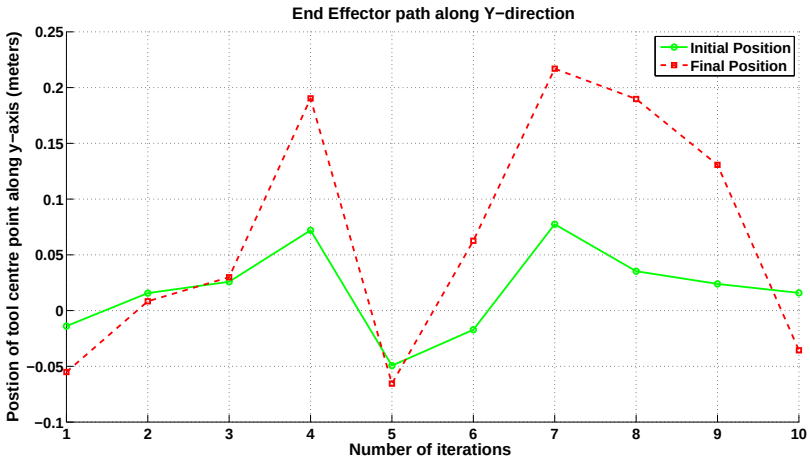

Figure 4.22: Robot's motion along the y-axis direction

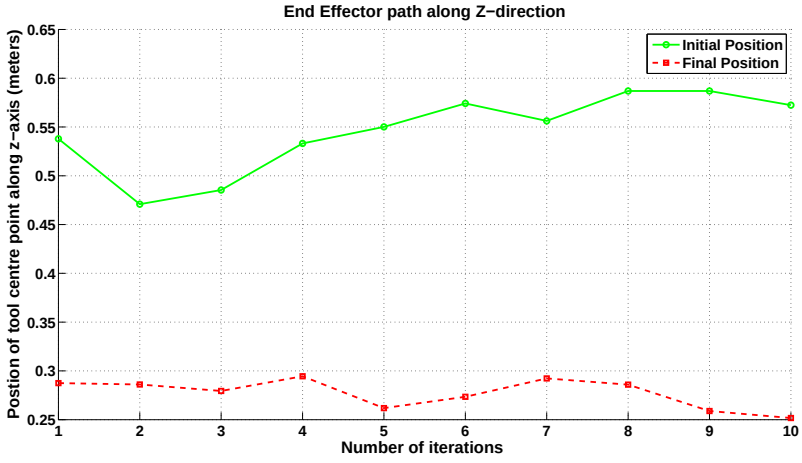

Figure 4.23: Robot's motion along the z-axis direction

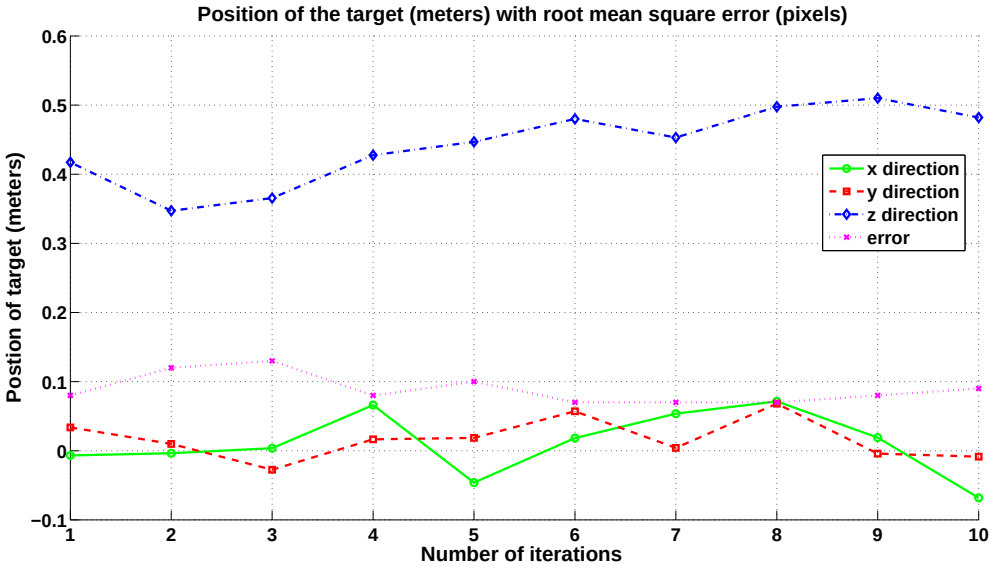

Figure 4.24: Target's position along with the root mean square error

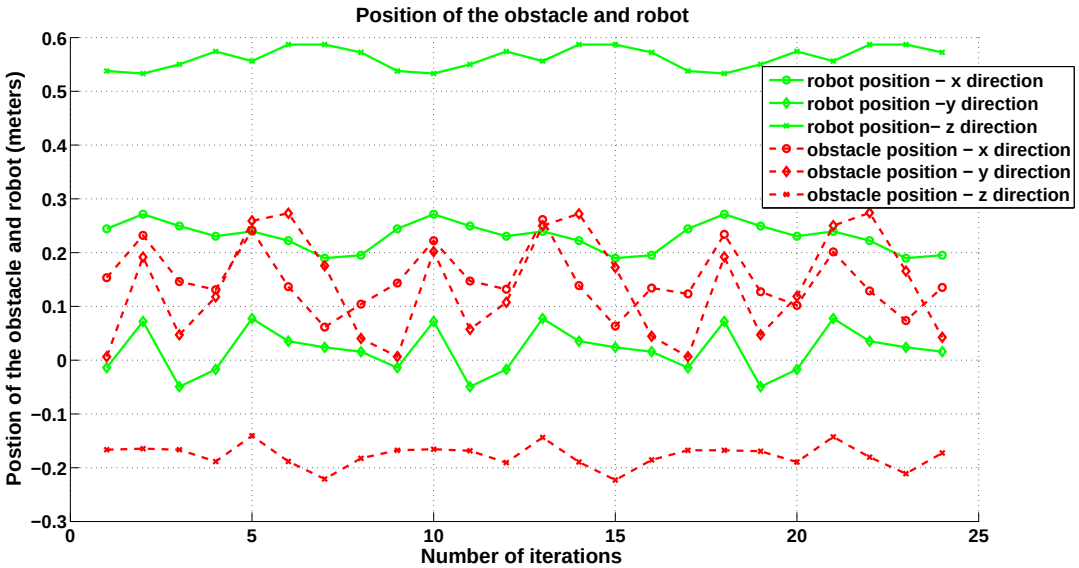

Figure 4.25: Position of the obstacle along with the robot's motion

Chapter 5

Conclusions and Recommendations

This chapter summarizes the main conclusions of the present work along with the suggestions for future research.

5.1 Conclusions

5.1.1 Overview

The use of a coverage model on a camera network is clearly illustrated for an industrial robotic work cell application. The coverage strength model helps to achieve accurate view selection results. The use of the occlusion model helps to avoid any obstruction in the robot's motion. A robot with the camera mounted on its end effector helps to avoid occlusion and ambiguity and in turn helps to build the accuracy of the system. But it is also important to note that the camera mounted on the end effector loses its focus on the target during its final stage of grasping. Building an autonomous camera equipped robot system is beneficial to many applications spread across various sectors. Robots that can perceive its environment with the help of the vision system is more powerful and useful. This research has aimed on building such a system.

The experiments indicate that the robot's motion can be controlled based on the obstacle pose. The 2D dynamic pose of more than two objects can be obtained from the same camera. Using this concept, the 2D target and obstacle are tracked by a single camera that belongs to a network of cameras. The camera network helps the camera equipped robot to perceive the position of the target and obstacle. The results indicate that obtaining the robust robot pose for target grasping or for a similar kind of application requires more detail

planning, as discussed in Section 5.2.

The research is significant, as it involves two dynamic (moving) entities (target and obstacle). It has been stated in Section 1.5.3 that, the use of a model like coverage strength has been used widely for sensor planning in literature, than for view selection [9]. Section 1.5.3 also states that, as per the survey done in [21], most of the purposive sensing methods designed for robotic manipulators deals with sensor planning and optimal sensor placement. Thus, scene perception through robotic vision using a model such as coverage strength model is novel in the field of 'purposive sensing for robotic manipulators.' There are very few applications in literature that involves a model like coverage strength model for view selection. Applying the concept of view selection to plan the motion of the robot has been an unexplored research area. When compared with the previous works [18, 23] related to collision avoidance of a robotic manipulator, the motion planning method implemented in this thesis is simple and provides robust results.

5.1.2 Summary of Contributions

In this thesis, motion planning for a robotic manipulator is implemented using a camera network. The primary contribution of this thesis work is the design and implementation of an algorithm for the motion planning of a robot. This algorithm succeeds in collision avoidance of the robotic arm with dynamic obstacles.

In conjunction with the development of this algorithm, several experiments have been conducted in order to achieve robust simultaneous pose estimation of 2D objects. The pose estimations techniques used can be applied to any planar objects. Another important contribution is the pose composition and conversion done in order to establish the motion planning of the robot.

The other important contributions are: calibration of a camera network along with hand-eye calibration of the robot, implementation of a successful and standard communication between different entities of the system and usage of coverage strength model in order to compute the camera with best view. The contributions listed in this Section were implemented practically and the results recorded are shown in Section 4.4.

5.2 Recommendations

Some of the future recommendations that could be inferred from the experiments done are discussed in this section. A 3D model of the target or obstacle could be used for pose estimation, as it would help to simulate the real world application scenario. Pose estimation of 3D objects gives much more information than planar pose estimation. The path planning for the robot could be improved by using a controller. A real time robotic application that does operations like grasping requires precise and accurate data. Hence, a detailed implementation of path planning for the robot could help the robot to reach the target more efficiently. The system proposed in this research work can be extended to a much larger camera network for an application that involves more than one robot.

Attaching a mechanical gripper to the end effector of the robot to do a particular task could be the most desired future enhancement of this research work. This would require a precise path planner for the robot that considers the time delay issues associated with the robotic manipulator and system. The pose estimation of the target with respect to the gripper becomes more specific and needs to be highly accurate for the successful execution of the required task.

Bibliography

[1] B. Boufama, "Image formation and camera model," in *Lecture notes for Visual Processing*. University of Windsor, 2011, ch. 3.

[2] *Solution Guide III-C 3D Vision*, MVTech Software GmBH, 2010.

[3] *MELFA Industrial Robots Instruction Manual (Detailed explanation of Functions and Operations)*, Mitsubishi Electric Industrial Automation, 2005.

[4] *MELFA Industrial Robots Specifications Manual RV 1A/RV 2A-J Series*, Mitsubishi Electric Industrial Automation, 2002.

[5] *Prosilica User Manual EC1350C and EC1350*, Prosilica Inc., 2005.

[6] A. Mavrinac, J. L. A. Herrera, and X. Chen, "A fuzzy model for coverage evaluation of cameras and multi-camera networks," in *Proc. 4th ACM/IEEE International Conference on Distributed Smart Cameras*, 2010, pp. 95–102.

[7] ——, "Evaluating the fuzzy coverage model for 3d multi-camera network applications," in *Proc. International Conference on Intelligent robotics and applications*, 2010, pp. 692–701.

[8] J. L. Alarcon Herrera, A. Mavrinac, and X. Chen, "Sensor Planning for Range Cameras via a Coverage Strength Model," in *Proc. IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2011, pp. 838–843.

[9] A. Mavrinac, D. Rajan, Y. Tan, and X. Chen, "Task-oriented optimal view selection in a calibrated multi-camera system," in *Proc. IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2012.

[10] J. Pauli, *Learning-Based Robot Vision: Principles and Applications*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2001.

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

[12] Y. Charfi, N. Wakamiya, and M. Murata, "Challenging issues in visual sensor networks," *IEEE Wireless Communications*, vol. 16, no. 2, pp. 44 –49, 2009.

[13] F. Brecht, W. Bart, B. Luc, L. G. Jos Ramn, and T. F. Carlos, *Industrial robot manipulator guarding using artificial vision*, ser. Robot Vision. In-Tech, 2010, pp. 429–454. [Online]. Available: http://sciyo.com/articles/show/title/industrial-robot-manipulator-guarding-using-artificial-vision

[14] A.-J. Baerveldt, "Cooperation between man and robot: interface and safety," in *Proc. IEEE International Workshop on Robot and Human Communication*, 1992, pp. 183 –187.

[15] P. Steinhaus, M. Ehrenmann, and R. Dillmann, "Mephisto: A modular and existensible path planning system using observation," in *Proc. 1st International Conference on Computer Vision Systems*, 1999, pp. 361–375.

[16] E. Cervera, N. Garcia-Aracil, E. Martinez, L. Nomdedeu, and A. Del Pobil, "Safety for a robot arm moving amidst humans by using panoramic vision," in *Proc. IEEE International Conference on Robotics and Automation*, 2008, pp. 2183 –2188.

[17] B. Ostermann, "Industrial jointed arm robot evading dynamic objects," Master's thesis, University of Applied Sciences Bonn-Rhein-Sieg, 53757 Sankt Augustin, January 2009.

[18] A. Winkler and J. Such, "Vision based collision avoidance of industrial robots," in *Proc. 18th International Federation of Automatic Control World Congress*, 2011.

[19] L. Tessens, M. Morbee, H. Lee, W. Philips, and H. Aghajan, "Principal view determination for camera selection in distributed smart camera networks," in *Proc. 2nd ACM/IEEE International Conference on Distributed Smart Cameras*, 2008, pp. 1 –10.

[20] E. Monari and K. Kroschel, "A knowledge-based camera selection approach for object tracking in large sensor networks," in *Proc. 3rd ACM/IEEE International Conference on Distributed Smart Cameras*, 2009, pp. 1 –8.

[21] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *International Journal of Robotics Research*, vol. 30, no. 11, pp. 1343–1377, 2011.

[22] F. Kececi, M. Tonko, H.-H. Nagel, and V. Gengenbach, "Improving visually servoed disassembly operations by automatic camera placement," in *Proc. IEEE International Conference on Robotics and Automation*, 1998, pp. 2947 –2952 vol.4.

[23] T. Gecks and D. Henrich, "Human-robot cooperation: safe pick-and-place operations," in *IEEE International Workshop on Robot and Human Interactive Communication*, 2005, pp. 549 – 554.

[24] G. Sommer, B. Rosenhahn, and Y. Zhang, "Pose estimation using geometric constraints," in *Proc. 10th International Workshop on Theoretical Foundations of Computer Vision: Multi-Image Analysis*, 2001, pp. 153–170.

[25] G. Stockman and L. G. Shapiro, *Computer Vision*, 1st ed. Prentice Hall PTR, 2001.

[26] *Halcon/HDevelop Reference manual*, MVTech Software GmBH, 2010.

[27] P. Felzenszwalb and R. Zabih, "Dynamic programming and graph algorithms in computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 4, pp. 721 –740, 2011.

[28] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[29] C. K. Cowan and P. D. Kovesi, "Automatic Sensor Placement from Vision Task Requirements," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 407–416, 1988.

[30] K. A. Tarabanis, R. Y. Tsai, and P. K. Allen, "The MVP Sensor Planning System for Robotic Vision Tasks," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 1, pp. 72–85, 1995.

[31] T. Möller and B. Trumbore, "Fast, Minimum Storage Ray/Triangle Intersection," *Journal of Graphics Tools*, vol. 2, no. 1, pp. 21–28, 1997.

[32] P. J. Schneider and D. H. Eberly, *Geometric Tools for Computer Graphics*. Morgan Kaufmann, 2002.

[33] H. R. Tiwary, "On the Hardness of Computing Intersection, Union and Minkowski Sum of Polytopes," *Discrete and Computational Geometry*, vol. 40, no. 3, pp. 469–479, 2008.

[34] "Workspace 5, robot simulation software," 1989-2012, 'Demo Version 5.2.4.1'. [Online]. Available: http://www.workspace5.com/

Appendix A

Glossary of Terms

Non-singular matrix
A square matrix for which inverse exists. Also, a square matrix is non-singular iff its determinant is nonzero.

Basis
A basis is a set of linearly independent vectors that can represent every vector in a given vector space . In more general terms, a basis is a linearly independent spanning set.

NP-hard
It denotes non-deterministic polynomial-time hard.

$SE(3)$
The special Euclidean group that represents rigid body motions.

Pose
A rigid Euclidean transformation describing an object's position and orientation.

Rotation matrix
A n x n real orthogonal matrix corresponding to a geometric rotation about a fixed origin in n-dimensional Euclidean space.

Translational vector

A n-element vector corresponding to a geometric translation in n-dimensional Euclidean space.

Homography
A homography is an invertible transformation from a projective space to itself that maps straight lines to straight lines

View selection
A method by which the camera which has the best and accurate coverage of an object is selected

2D space
Two dimensional space is a geometric model of the planar projection of the physical universe.

3D space
Three-dimensional space is a geometric 3-parameters model of the physical universe. These three dimensions are commonly called length, width, and depth.

$\mathbb{R}^3$
$\mathbb{R}^3$ represents the three dimensional Euclidean space.

SVD
Singular Value Decomposition is a method used to factorize a real or complex matrix.

Vita Auctoris

NAME	:	Durga Rajan
BIRTH YEAR	:	1985
BIRTH PLACE	:	INDIA
EDUCATION		
2010–2012	:	**Master of Applied Science**
		Electrical and Computer Engineering
		University of Windsor, Windsor, Ontario, Canada
2003–2007	:	**Bachelor of Engineering**
		Electronics and Communication Engineering
		Anna University, Chennai, India