ACCURACY ENHANCEMENT OF INDUSTRIAL ROBOTS BY DYNAMIC POSE CORRECTION

Sepehr Gharaaty

A thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science

Concordia University

Montréal, Québec, Canada

September 2016

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Sepehr Gharaaty**

Entitled: **Accuracy Enhancement of Industrial Robots by Dynamic Pose Correction**

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

Master of Applied Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Dr. Mehdi Hojjati, Chair

_____ Dr. Amir Aghdam, ECE, External Examiner

_____ Dr. Brandon Gordon, MIE, Internal Examiner

_____ Dr. Wen-Fang Xie, Supervisor

Approved _____

Chair of Department or Graduate Program Director

_____ 2016 _____ _____

Amir Asif, Ph.D., Interim Dean

Faculty of Engineering and Computer Science

Abstract

Accuracy Enhancement of Industrial Robots by Dynamic Pose Correction

Sepehr Gharaaty

The industrial robots are widely employed in various industries. Normally, the industrial robots are highly repeatable. However, their accuracy is relatively poor. If the robot's end-effector is required to move to a pre-calculated pose (as in off-line programming), its position error may reach a couple of millimetres. To meet the growing demand for high absolute accuracy in robotic applications such as deburring, polishing, drilling, and fastening, a lot of research work has been carried out. Robot calibration is normally used to enhance the accuracy by using external pose measurement sensors such as laser tracker. However, the calibration procedure is long and the cost is high. Also, the accuracy enhancement is limited and the best reported accuracy is around $\pm 0.1\, mm$. In addition, the changing operating environment and the wear and tear of the robot affect the accuracy.

This research aims at developing a novel and cost-effective dynamic pose correction (DPC) strategy to address the above-mentioned issues on the accuracy enhancement. This strategy uses the vision system, i.e. C-Track from Creaform Inc. to measure the pose and integrates with the robotic controller, also known as visual servoing. To realize this strategy, three main research activities have been conducted.

First, the pose of the robot is obtained from the binocular sensor. The triangulation method for estimating the pose of an object is elaborated and the C-Track as a binocular sensor is introduced. In order to remove the noise from the C-Track's measurements, the

analysis on the measured data is carried out. A root mean square (RMS) method is used to achieve reliable pose estimation.

Next, the DPC strategy is designed and simulated for an industrial robot, Fanuc M20-iA. This strategy can correct the position and orientation of the robot end-effector by using position-based visual servoing. A proportional-integral-derivative (PID) controller is proposed to achieve the dynamic pose correction. The algorithm does not need the kinematic and dynamic model of the robot. The controller is fully tested in Matlab/Simulink with robotic toolbox where Fanuc M20-iA is simulated. The simulation results validated the effectiveness of the proposed DPC.

The final research work is dedicated to the experimental testing of the proposed DPC on Fanuc M20-iA. The pose estimated from the C-Track serves as the feedback and the output of the DPC is given to the robot controller through Fanuc dynamic path modification (DPM) function. As a result, the robot is guided to the desired pose in real time. The experimental results demonstrate that the robot can achieve the position accuracy $\pm 0.05\, mm$ and orientation accuracy ± 0.05 degree.

Acknowledgments

By completing this dissertation, my Masters degree takes the final step towards its conclusion. While conceiving these few words, I realized how many people I owe for their support, so acknowledging all of them within this limited space is not possible. Therefore, I wish to express my sincere gratitude to all those who have helped me in attaining such a goal specially my family and friends.

In addition, it is my pleasure to express my deepest gratitude to my supervisor, Prof. Wen-Fang Xie, for her constant support, direction and guidance. I would like to thank her for sharing all of her talent and experience, which I learned from, and for all the opportunities she gave me during my research. I particularly appreciated the freedom she accorded to me during my research and her trust in my capabilities. She always encouraged me every step of the way, which I am extremely grateful for. I am aware that spending these years under her supervision was an invaluable opportunity in my professional and scientific career.

In addition, I would like to thank Prof. Ilian Bonev of École de Technologie Supérieure, for being a constant source of motivation and for giving his precious and kind advice regarding my research.

Hearty thanks also go to my colleagues and friends who accompanied me along these years: Ms. Tingting Shu, Mr. Alexandre Filion, Dr. Ahmed Joubair, Dr. Amir Hajiloo, Dr. Mohammad Keshmiri, Mr. Abolfazl Mohebbi and Mr. Ahmad Ghasemi. I am also grateful

to everyone who has given me support, but I did not have a chance to thank them.

I would like to acknowledge the financial support throughout the course of this study from the Consortium for Research and Innovation in Aerospace in Quebec (CRIAQ).

I would also like to thank my two sisters for their continuous support throughout the years. Whether it be encouraging me or giving me advice, they were always available to help.

Last, but not least, my gratitude and sincere thanks go to my parents for their support, kindness and always being there regardless of my choices. I would not be where I am today without them.

To my beloved parents

Mahmood** and **Marzieh

and my lovely sisters

Soha** and **Sadaf

Contents

List of Figures

List of Tables

Nomenclature

${}^{S}P = [X\ Y\ Z]$	Coordinates of a point represented in sensor frame
P = [u, v]	Coordinates of a point represented in image plane (pixel dimension)
f	Camera focal length
ρ	Pixel to Metric transformation constant
${}^{B}_{A}H$	Homogeneous Transformation Matrix between Frame B and A
W	Rotation Angle about X
P	Rotation Angle about Y
R	Rotation Angle about Z
$\vec{P}_{des}$	End-Effector Desired Pose
$\vec{P}_{Cur}$	End-Effector Current Pose
K_p	Proportional Gain
K_i	Integral Gain
K_d	Derivative Gain
$R_x(\theta)$	Pure Rotation about X axis
$D_x(d)$	Pure Translation along X axis
J(q)	Robot Jacobian Matrix

$\vec{\dot{q}}_{6\times1}$	Robot Joint Space Velocity
$\vec{\dot{P}}_{6\times1}$	Robot Cartesian Space Velocity
{C}	Cube Frame
{M}	Model Frame
{L}	Left Camera Frame
{R}	Right Camera Frame
{S}	Camera Frame
{T}	Tool Frame
{U}	User Frame
DO	Digital Output
DI	Digital Input
PR	Position Register
T_s	Sampling Interval
T_c	Control Interval
ARC	Adaptive Robot Control
ATC	Automatic Tool Changer
CMM	Coordinate Measuring Machines
CRIAQ	Consortium for Research and Innovation in Aerospace in Quebec
D-H	Denavite-Hartenberg
DPC	Dynamic Pose Correction
DPM	Dynamic Path Modification
EGM	External Guided Motion
ÉTS	École de technologie supérieure

FOV	Field of View
HVS	Hybrid Visual Servoing
IBVS	Image Based Visual Servoing
ILC	Iterative Learning Control
ISE	Integral of Squared Error
ISO	International Standards Organization
MPC	Model Predictive Controller
PBVS	Position Based Visual Servoing
PCDK	PC Developer's Kit
PID	Proportional-Integral-Derivative
RMS	Root Mean Square
SCARA	Selective Compliance Assembly Robot Arm
TCP	Tool Center Point
TPP	Teach Pendant Program

Chapter 1

Introduction

Currently, robots are widely used in different industries, from simple packaging companies to complicated aerospace applications. Most of the manufacturing industries use robots for inspection, laser cutting, assembly, welding, drilling, palletizing and fastening. In addition, robotic systems are employed in military, urban and exploratory applications [1]. Industrial robots can be considered as the inseparable part of the automated manufacturing system in performing repetitive tasks [2]. The main existing challenge is the relative low absolute accuracy of standard industrial robots [3].

Robot calibration is an option to increase the accuracy of industrial robots. This is done by minimizing the differences between the real kinematic model and the nominal one of the robot in the controller. Although robot calibration can improve the accuracy of the robots with the help of complex mathematical algorithms and metrology equipment, the provided accuracy appears to be insufficient and the procedure is costly according to [4–6].

In addition to the calibration, vision data can be integrated into the robot control system to enhance the performance and flexibility of the robot to deal with unstructured environments [1]. Using vision information in a feedback control system to control the robot manipulators, i.e. "Visual Servoing systems", has attracted great attention of the researchers

in the past decades.

The main objective of this project is to develop a dynamic pose control algorithm using the position and orientation feedback measurement from Creaform's C-Track to apply on-line corrections to guide the robot to the desired pose with better accuracy than that achieved in [4–6]. In order to implement the mentioned method, a user-friendly software is developed in Microsoft Visual Basic and the algorithm is tested on the Fanuc robot in the lab.

1.1 Overview

In the following, a brief introduction about industrial robots is presented. Different types of robots used in industry and their programming methods are briefly explained. Also, the main sources of errors which affect the precision of the industrial robots are addressed.

1.1.1 Industrial Robots

A general definition of an industrial robot as defined by International Standards Organization (ISO) is an automatically controlled, programmable, multi-purpose manipulator, which is able to perform specific tasks in a repeatable manner [7]. The industrial robots can be categorized into the following five main types: Cartesian, SCARA, Cylindrical, Parallel and Articulated.

Cartesian robots also known as gantry robots have three prismatic joints allowing movement along three axis (X,Y,Z). These types of robots are mostly used for pick and place, assembly, handling and arc welding tasks [12]. Selective Compliance Assembly Robot Arm (SCARA) presented in 1981 has two parallel rotary joints allowing compliance in a plane [13]. Assembly, pick and place and handling are some examples of the applications for this type of robots.

(a) (b) (c)

(d) (e)

Figure 1.1: Different industrial robot types (a) Cartesian robot [8] (b) SCARA robot [9] (c) Cylindrical robot [10] (d) Delta robot [9] and (e) Serial robot [11]

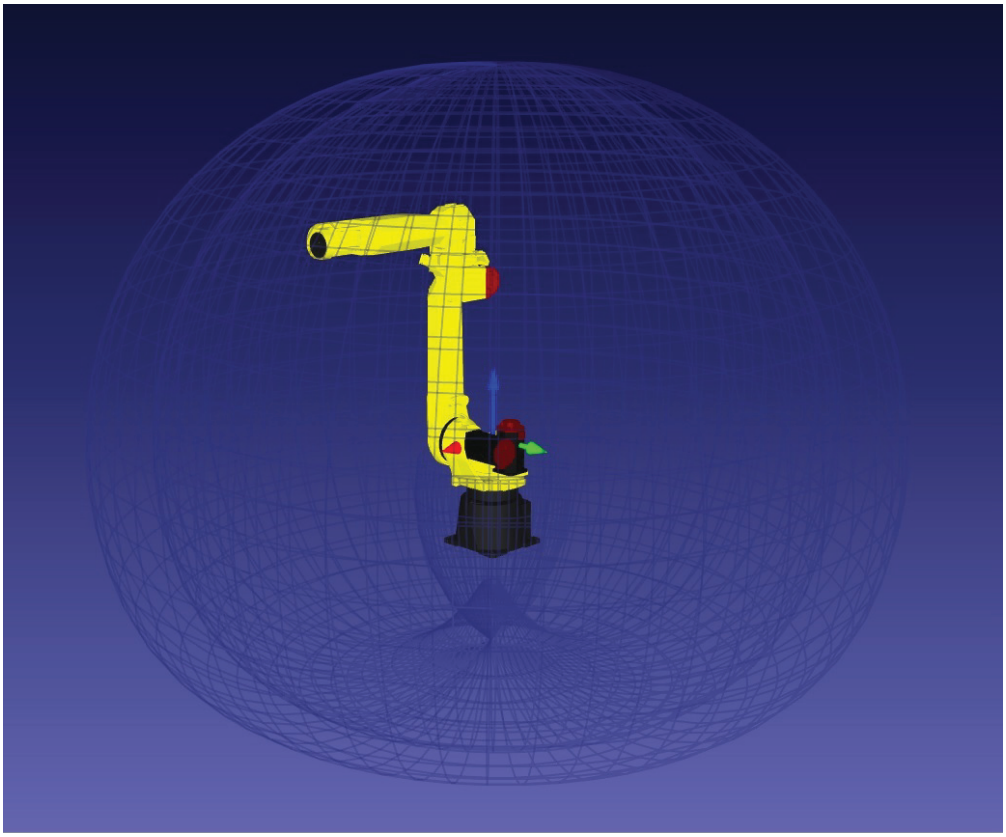


Figure 1.2: FANUC M20-iA workspace

Cylindrical robots are mainly used for assembly, spot welding and handling. They usually consist of one rotary joint and 2 prismatic joints allowing horizontal and vertical movements and providing a cylinderical workspace [12].

Parallel robots are another category of industrial robots which have concurrent prismatic or rotary joints connecting a platform to a base. The linkage joints are independent which are working in parallel. Parallel robots have limited workspace due to the linkage of the joints and mostly are used for handling platforms. Delta robot, also known as spider robot, is a well-known type of parallel robots consisting of parallel joints connected to the base. Delta robots are mostly used in food and electronic industry for rapid pick and place. The last robot in the above mentioned category is Articulated robots also known as Serial robots

which consist of two or more axes connecting the robot base to the end-effector by rotary joints. These types of robots are used in welding, painting, palletizing and most of the manufacturing industries [12]. Figure 1.1 shows an example of each type of the robots used in industry.

The robot used in the experiments for this project is an articulated robot (FANUC M20-iA), which has six degrees-of-freedom with six rotary joints. The robot weighs $250\,kg$, with a maximum load capacity $20\,kg$ and the maximum reach for this type of robot is $1811\,mm$. Figure 1.3 shows the FANUC M20-iA in the lab in Concordia University. The repeatability of this robot is $\pm 0.08\,mm$, while the absolute position accuracy can be worse than $7\,mm$ [11, 14]. Losing accuracy due to the abrasive wearing of transmission parts, workload and temperature change in the environment, is one of the main problems of this robot.

An industrial robot consists of different frames including Base Frame, Tool Frame and User Frame. In serial robots, the base frame is located at the base of the manipulator, which it is mounted to the floor. The tool frame's origin is located at the Tool Center Point (TCP) of the tool, which is attached to the robot. The base frame is the start of the robot kinematic chain, and the tool-frame is the end of robot kinematic chain. The user frame is an arbitrary frame defined by the user in the workspace of the robot for programming purposes.

Serial robots can be modeled by the Denavit-Hartenberg parameters. The kinematic model of a serial robot is obtained by assigning standard frames at each joint of the robot and finding the relation between the assigned frames from the base to the tool [15]. The kinematic model of the robot is used to obtain the pose of the end-effector according to the joint angles known as forward kinematics. On the other hand, inverse kinematics is used to extract the joint angles according to a given pose of the end-effector. The transitions between joint space and cartesian space are realized through the forward kinematic and inverse kinematic as shown in Figure 1.4.

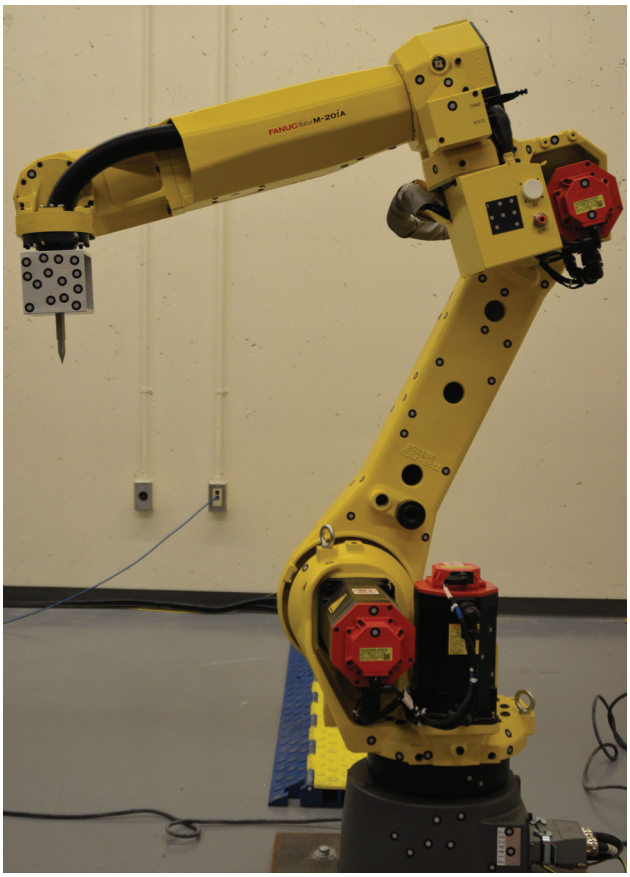


Figure 1.3: FANUC M20-iA robot

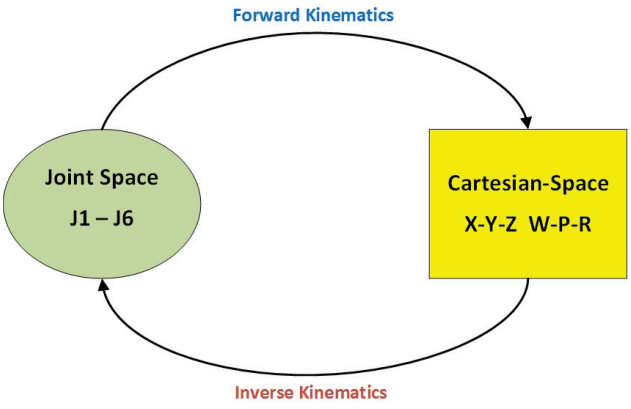


Figure 1.4: Relation between Joint-Space and Cartesian-Space

Figure 1.5: Fanuc robot control cabinet and teach pendant

1.1.2 Robot Programming

Industrial robots can be programmed either by the Teach Pendant(reading the encoder data) or off-line programming method. The first method directly uses the robot controller to read and record the position and orientation of the end-effector with respect to the robot base frame or any user-defined frame. A Teach Pendant Program (TPP) consists of the commands which move the robot from one pose to another pose in space.

In order to record (teach) a pose in the space, the robot should be jogged to the desired point first. Considering a very massive program including dozens of specific poses, the programming procedure takes too much time. Moreover the robot cannot be used for production during the programming.

On the other hand, the second method does not interfere with the production. In this method the robot program is written separately in a simulation software (such as Robotguide for FANUC robots) based on the nominal kinematic model of the robot.

The behaviour of the robot can be evaluated based on the simulation to check the singularities, collisions and generally the performance of the robot. The program can be uploaded to the real robot after finding the simulation results are satisfactory [16]. Since the nominal kinematic model of the robot might be different from the real one [17], the simulation results are different from the behaviour of the real robot and the task may not be performed accurately.

Robots are normally programmed based on the point to point movements, which can be performed as a linear, joint, or circular motion specified by L, J, C respectively. For the linear motion, all the joints are actuated so that the robot end-effector trajectory from point A to B is a straight line. On the other hand, for the joint motion there is no control on the trajectory of the robot end-effector since all the motors are actuated so that the joints start and finish the movement at the same time. In this case, all the joints will do the minimum motion from point A to point B. Circular motion is another option which allows the robot to move on a circular trajectory between three points. Figure 1.6 shows the difference of three motion types from point A to B.

1.1.3 Sources of Error in Industrial Robots

Enhancing the accuracy of the industrial robots have always been a main concern for both manufacturers and operators. On one hand, the new applications in automotive industries and aerospace need high accuracy robots which can perform the tasks precisely. On the other hand, there are many factors which affect the accuracy of a robot [18] such as manufacturing process, size of the robot and environment, etc.

The components of a robot and the environment where the robot is operated can be considered as the main sources of errors in industrial robots. According to [4], the sources of errors for an industrial robot can be categorized into three main classes including geometric

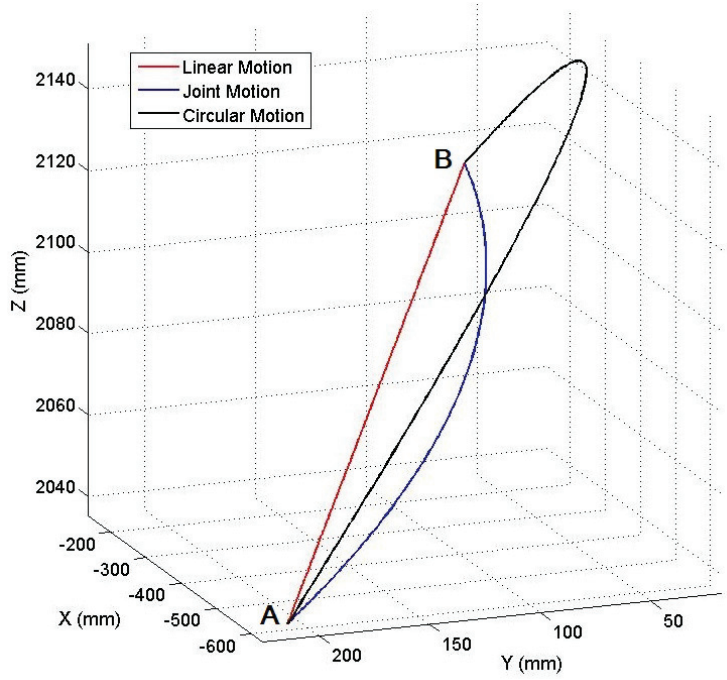


Figure 1.6: Different motion types from point A to point B

errors, thermal errors and system errors.

Geometric errors are due to the manufacturing process, tolerances and inevitable inaccuracies of the components in an industrial robot. As a result, the nominal mathematical model of the robot in the robot controller is not exactly the same as the manufactured one. The difference between the mathematical model of the robot in the controller and the real one affects the accuracy of the industrial robots [19].

In addition to tolerances and inaccuracies of the manufacturing process, the components of a robot are affected by the temperature change which is the source of thermal errors. The temperature change might be due to the environment changes, heating from the motors, bearings, drive mechanisms and electronic components. Among the mentioned heating sources, the effect of environment change is more uniform than the others because the heat source is not localized [17]. System errors normally consist of sensor and encoder inaccuracies, structural deformations, wear, frictions and backlash in the gears of the motors.

Calibration of an industrial robot aims to find the accurate kinematic model parameters of the robot and implementing the identified parameters in the robot controller. Calibration process requires very accurate external measurement instruments to provide accurate pose of the end-effector at any configuration. To name a few, MaxSHOT-3D from Creaform and FARO laser tracker, are the renowned measurement sensors used for robot calibration during the past years [20].

1.2 Motivation

In order to perform a specific task precisely using the industrial robots, it is necessary to compensate the geometric and non-geometric sources of errors. The first solution for this issue is to calibrate the robot by recalculating the exact kinematic model parameters of the robot and modifying the nominal kinematic parameters of the robot in the controller.

To the best of our knowledge, current calibration packages provided by the industrial robot manufacturers (ABB, FANUC, KUKA) or other companies (Nikon Metrology, Dynalog) are time consuming, costly and, in some cases, not satisfactory for the customers in terms of the accuracy [20]. Furthermore, the calibration process needs to be repeated approximately every year (depending on the work load of the robot). Another drawback of this solution is that production must be suspended while the robot is being calibrated.

The best reported position accuracy achieved so far, is greater than $0.1\,mm$ according to [3, 5, 21]. On one hand, the growing demand for industrial robots with higher absolute accuracy and, on the other hand, the lack of an accurate robot pose control algorithm is the main motivation of this research.

1.3 Contribution

In this research, a novel and practical control method is developed, which can handle the sources of errors by using online measurement instruments and performing dynamic corrections. Using this method, not only the position, but also the orientation, is controlled and the accuracy is improved to $\pm 0.05\, mm$ and $\pm 0.05\, deg$ for the position and orientation, respectively.

This method is completely independent of the kinematic chain parameters and is applicable to either un-calibrated or calibrated industrial robots, therefore, eliminating the need for annual calibration. In addition, the effect of the operator's skills on handling the measuring instruments is eliminated due to the automatic measurement procedures developed in the software.

Using the proposed strategy, the robot is guided to the desired pose by measuring the pose of the tool dynamically, and comparing the acquired pose information with the desired ones. This strategy has been extensively tested on the Fanuc M20-iA robot whose repeatability in the best condition is $0.08\, mm$.

The methodology of using photogrammetry sensors to increase the robot's absolute accuracy, can be applied to the other categories of the robots, such as parallel robots, etc. It will render the current industrial robots high-end ones without going through costly calibration procedure or resorting to high-end encoders. Also the developed control strategy can be integrated with the current industrial robot controller and achieve the high accuracy without suspending the robot production process.

1.4 Thesis Outline

This thesis has seven main chapters, which are organized as follows; Chapter 1 consists of an introduction about industrial robots, programming methods, the main sources of errors and the contribution of this research. In Chapter 2, a literature review on visual servoing and accuracy enhancement of industrial robots are addressed.

The pose estimation methods and C-Track measurement sensor are introduced in Chapter 3. The basic theory for Fanuc Dynamic Path Modification and PID controller, as well as the simulation setup and results are presented in Chapter 4.

Chapter 5 is devoted to describing the experimental setup including the robot tool and the developed software. The techniques for calculating the offsets are also explained in this chapter.

The details of the experiments and the results are presented in Chapter 6. Finally, the conclusion of the thesis and potential future works are presented in Chapter 7. The references cited in this thesis are sorted in the bibliography section.

Chapter 2

Literature Review

In this chapter, literature review on visual servoing and classifications of different visual servoing systems are presented. The various research on accuracy enhancement, as well as the available solutions and software for accuracy enhancement of industrial robots are introduced. Also, a brief comparison between the C-Track and the laser tracker is given in this chapter.

2.1 Visual Servoing

Visual servoing control is referred to controlling the motion of a robot using the feedback data from a camera. In the early 1970s, Shirai and Inoue [22] described how the accuracy can be improved by using vision system in the feedback. The term "Visual Servoing" has been introduced for the first time by Hill and Park [23] but, prior to that, the term "Vision Feedback" was generally used.

The major application of Visual Servoing is in the robotics to control the position and orientation (pose) of the robot end-effector to the desired onces. Spot and arc welding,

Figure 2.1: Application of visual servoing in surgery robotic system [26]

assembling, painting and grasping objects are some examples of the tasks which can be performed by the robotic systems [24]. Visual servoing is also used in medical applications, such as surgery, by either positioning instruments or performing operations which has attracted great attention of the researchers [25].

Depending on the configuration of the camera, visual servoing is divided into two main categories. If the camera is mounted on the robot's end-effector and moves with the robot, it is called eye-in-hand configuration. If the camera is installed and fixed in the workspace of the robot and observe the robot and the workspace, it is called eye-to-hand configuration [27]. These configurations are shown in Figure 2.2.

Based on the application and the workspace, a visual servoing system can have one camera, known as monocular system, two cameras, known as binocular system (stereo vision), or multiple cameras. Each of these categories can be used in both eye-in-hand and eye-to-hand configuration. A comprehensive survey on each category is reported by Kragic and

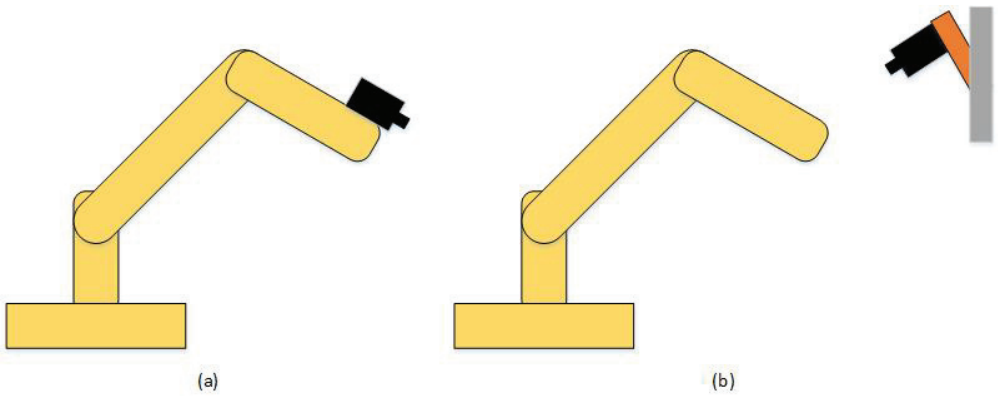


Figure 2.2: Camera configuration in visual servoing control, (a) Eye-in-hand Configuration (b) Eye-to-hand Configuration

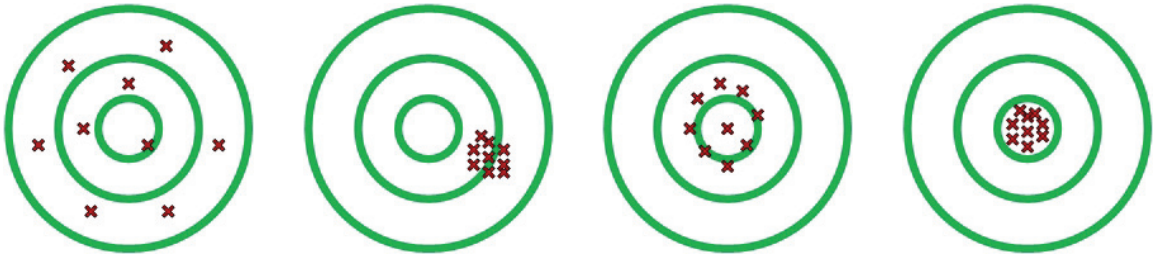
Christensen [28].

Visual servoing is classified into three main strategies according to the type of vision feedback data including Position Based Visual Servoing (PBVS), Image Based Visual Servoing (IBVS) and Hybrid Visual Servoing (HVS) [29].

If the extracted feature from the image captured by the camera is the position and orientation of the robot's end-effector, this strategy is called Position Based Visual Servoing. In this strategy, the desired pose of the end-effector is compared with the actual one measured by the camera, and the error is computed as the input to the controller. On the other hand, in IBVS, image features are used as the feedback measurement and the difference between the desired features and the actual ones (captured by the camera) creates the error signal for visual servoing control.

HVS is introduced combining the IBVS and PBVS strategies. $2\frac{1}{2}D$ method is one popular example of hybrid strategy in which the robot is controlled by decoupling the end-effector translational and rotational motion control [30].

It is noticed that a lot of research works on PBVS are focused on relative pose correction in the eye-in-hand configuration. Very few papers deal with the absolute pose correction of industrial robots by using PBVS in the eye-to-hand configuration. The reasons could be



(a) (b) (c) (d)

Figure 2.3: (a) Bad Repeatability-Bad Accuracy (b) Good Repeatability-Bad Accuracy (c) Bad Repeatability-Good Accuracy (d) Good Repeatability-Good Accuracy

summerized as follows;

The first reason could be due to the lack of reliable 3D measurement sensor, which can provide high accuracy pose estimation. The second reason is the difficulty in integrating the measured pose with the current industrial robot controller seamlessly. The third one could be the difficulties in defining the tool frame with respect to the user frame to realize the absolute pose correction.

Besides, in order to reconstruct 3D information from 2D image information, accurate camera calibration is required.

2.2 Accuracy Enhancement

The precision of an industrial robot is specified by pose repeatability and accuracy. According to [6], "Pose repeatability is a measure of ability of the robot to move back to the same position and orientation" and "pose accuracy is defined as the ability of the robot to precisely move to a desired position in 3D space". Figure 2.3 illustrates the difference between pose accuracy and repeatability.

It is worth mentioning that the repeatability of the current industrial robots can be less than $0.1\,mm$ due to high resolution built-in encoders, while the pose accuracy can be

worse than $7\,mm$ [14] when the robots are programmed off-line. Although the problem of robot inaccuracy is difficult to be solved completely, it can be improved and reduced to an acceptable range depending on the application.

A lot of research has been carried out to improve robots positioning accuracy [3, 19, 31–33]. In 1985, Hayati and Mirmirani considered only geometric errors and presented a general method to estimate the errors of the link parameters for any manipulator [32]. Chen *et al.*, considered the geometric and non-geometric errors to improve the positioning accuracy. They concluded that understanding and implementing the details of error can reduce the mean of the error from $5.9\,mm$ to $0.28\,mm$ [33]. Further work which has been done to improve the accuracy of the robots can be found in [34–37]

Accuracy enhancement of the industrial robots can be performed in three different methods. The first method is to calibrate the robot by developing the accurate mathematical model of the robot which represents the real robot more precisely. This model can approximates the relation between the pose of the end-effector and the joint variables as accurately as possible.

Calibration process is performed by measuring the pose of the end-effector in different configurations using sophisticated measurement equipments and identifying the calibration parameters. In [5], an ABB IRB 1600 robot is calibrated by considering all possible geometric errors using laser tracker. In this work, 29 error parameters are identified using least square optimization. As a result, the maximum position errors reduced from $2.158\,mm$ to $0.69\,mm$ after calibration.

Robot calibration is offered by most of the leading industrial robot manufacturers to their customers; however, it is an expensive and time consuming service for the customers. Sometimes, the calibration can be performed on site, but in most cases the robot needs to be

sent to the manufacturer and their highly-skilled engineers who can work with 6D metrology measurement equipments. In addition to shipping and calibration service expenses, the production stop should also be considered as one of the disadvantages of this method. Considering the fact that the industrial robots needs to be calibrated frequently (once a year), these procedures will impose a significant annual cost for the company. For instance, GE Aviation pays approximately $7000 to send their robot to Detroit to be calibrated with a laser tracker, according to the reports.

In addition to calibration services from robot manufacturers, there are some companies who provide software packages for robot calibration. Dynalog, for example, has been operating exclusively in this field for more than 25 years. DynaCal is a calibration package offered by Dynalog which costs about $\$50,000$ [38]. The software is capable of being used in conjunction with a 3D measurement device such as laser tracker. Rocal is another calibration package offered by Nikon Metrology based on their 6D measurement sensors [20].

Considering the total expense for an accurate measurement sensor and the software, this solution is feasible only for large companies with hundreds of robots to be calibrated each year. Moreover, the existing third-party software for calibration requires filtering the robot programs before being uploaded to the robot which is automatically performed in case of manufacturer calibration [6].

The second solution is to install extra high-accuracy encoders at each joint of the robot. The secondary encoders are installed to eliminate the effect of backlash, elasticity and non-linearities in the gearbox [39]. In [21], the position error of a KUKA KR500/L340 robot is improved to less than $0.16\,mm$ by installing the secondary encoders on each axis. Figure 2.4 shows an example of the secondary encoder installed on a KUKA robot. Due to the differences among industrial robots in terms of the size and application, this method needs to be customized for each robot. Therefore, for companies with numerous robots, this method

Figure 2.4: Secondary encoder installed on a KUKA robot [21]

can be costly and infeasible.

The third method is to use an external measurement device to guide the robot end-effector to the desired pose dynamically. In this method, the external sensor provides the current pose of the robot on-line and the obtained pose information is compared with the desired value in a third-party software. Nikon Metrology offers a software package called Adaptive Robot Control (ARC) which uses this method to guide the robot to the desired pose. The software costs $\$50,000$ and is only compatible with Nikon Metrology's optical CMMs which costs more than $\$70,000$. Moreover, the best reported position accuracy achieved using ARC is $0.1\,mm$.

Currently most of the industrial robots are calibrated with laser trackers. From the economical point of view, the industry is looking for the most efficient method of the robot

calibration with the minimum costs. Among all measurement instruments available in the market, C-Track has the lowest price (less than $\$50,000$) compared to the laser trackers and Nikon Metrology's optical CMM which cost more than $\$100,000$ and $\$70,000$ respectively. In addition, the reflector targets for the C-Track are inexpensive (few cents each), while the reflectors for the laser tracker cost more than $1000 each. Moreover the C-Track is easier to use compared to laser tracker which needs high-skilled operators. The C-track can also measure the position of all the targets simultaneously but the laser tracker can only measure one point at a time. Vibration, air turbulence and temperature variation, humidity and also operators' skills and experience affect the laser tracker measurements. For instance achieving the best accuracy from laser tracker in the environments where heavy robots generates intense vibration is not always feasible [20].

Laser trackers have been commercialized and used in industry for more than 25 years. So far there was no reduction of their price while the C-Track launched in 2008 is less expensive than the laser tracker and has been used in various companies around the world [40].

Nubiola *et al.*, in [20] compared the C-Track and FARO laser track in a calibration process and concluded that the efficiency of the C-Track is equivalent to that of the laser tracker in calibrating the ABB IRB 120, while the C-Track is less expensive. After calibration the mean and maximum position errors of an ABB IRB 120 robot are reduced from 3 and $5\,mm$ to 0.15 and $0.5\,mm$ respectively.

Considering the above mentioned robot's accuracy enhancement methods, the robotic manufacturing companies, such as GE Aviation and Coriolis Inc, seek the help from robotic academic research teams, notably, Concordia and ÉTS, as well as and Creaform Inc, who provides various photogrammetry measurement sensors to find a cost-effective way to increase the absolute accuracy of the industrial robots.

Such method will use the C-Track as the feedback sensor to measure the pose of the

(a) (b)

Figure 2.5: On-line Photogrammetry systems (a) C-Track [40] and (b) FARO Laser Tracker [41]

robot end-effector and apply the dynamic pose correction to the robot controller in real-time. This method does not rely on the high-end encoders. Also, it does not need to intrusively modify the built-in kinematic model of the robot in the controller. It is expected that the proposed method be easily integrated with the current robot controller and satisfy the industry required precision.

2.3 Summary

In this chapter, a brief introduction to basic concepts and different categories of visual servoing system is presented. The efforts on enhancing the accuracy of the industrial robots including different techniques and solutions are mentioned. The comparison between two renowned 6D measurement sensors used for the robot calibration process is presented.

Chapter 3

Pose Estimation

In order to implement the position based visual servoing to enhance the accuracy of industrial robots, one has to obtain the accurate pose of the end-effector of the robot in real-time. This chapter is dedicated to the principles of the pose estimation using parallel and non-parallel binocular vision system. Also, the pose measurement sensor used in this research, i.e. C-Track is introduced. In addition, the VXelements software and analysis of the C-Track measurement data are presented.

3.1 Pose Calculation Using Dual Camera Sensor

In order to extract the position of a point from the image captured by a single camera, it is necessary to understand the geometric aspects of imaging process. According to [30], each camera consists of a lens that creates a 2D projection of the scene on the image plane. The projection cause loss of the depth information. Therefore some additional information is required to determine the coordinates of a point from an image. This information can be obtained using multiple cameras, multiple views using a single camera or using the geometric relationship between several feature points.

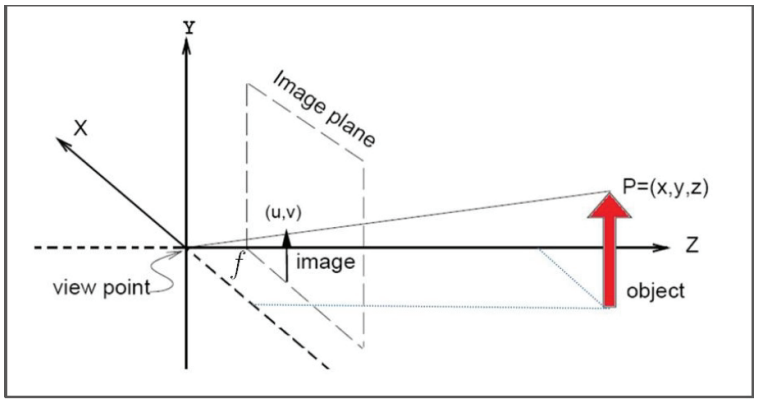


Figure 3.1: Coordinate frame for single camera system [30]

The coordinate frame for a single camera system is shown in Figure 3.1. In this figure, the z axis is considered perpendicular to the image plane. The camera coordinate frame is located at the distance of f from the image plane where f is the focal length of the camera.

According to Figure 3.1, for a point in a three dimensional space, ${}^{S}P = [X, Y, Z]^T$ whose coordinates are expressed with respect to the camera coordinate frame, $\{S\}$, the 2D projection coordinates in the image plane in pixel dimension is $p = [u, v]^T$. According to the perspective projection we have:

$$x = \frac{X}{Z} = \frac{(u - \sigma_u)}{f}.\rho_1 \tag{3.1a}$$

$$y = \frac{Y}{Z} = \frac{(v - \sigma_v)}{f}.\rho_2 \tag{3.1b}$$

where $[x, y]^T$ are the normalized coordinates from $p = [u, v]^T$ and $(\sigma_u, \sigma_v, f, \rho_1, \rho_2)$ are the camera intrinsic parameters [30]. σ_u and σ_v are the coordinates of the camera principal points, f is the focal length and ρ_1 and ρ_2 are the transformation constants from pixel dimensions to metric dimensions. As mentioned in Section 3.2, camera intrinsic parameters in this project are obtained automatically using the VXelements software and through the

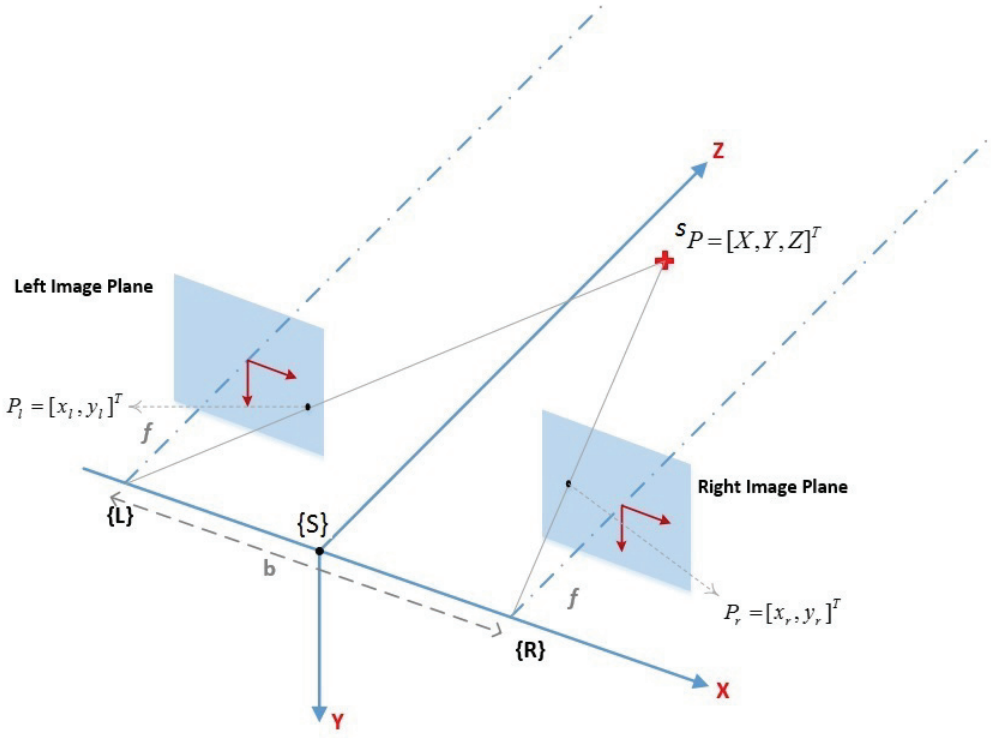


Figure 3.2: Parallel stereo vision system

calibration process.

Using the basic information for a monocular system, this approach can be extended to dual camera systems. Dual camera systems, also known as binocular systems, are composed of two parallel or non-parallel cameras. The focal points of two cameras are apart at distance $\frac{b}{2}$ with respect to the origin of sensor frame, $\{S\}$ (origin of the sensor frame is the midpoint of the cameras focal points) as shown in Figure 3.2.

According to Figure 3.2 the image plane for each camera is located at the distance of f from the camera focal point and orthogonal to the optical axis. Considering {L} and {R} corresponding to frames of the left camera and right camera respectively, the image coordinates of a 3D point P, observed by both cameras can be written as:

$$p_l = [x_l, y_l]^T \tag{3.2a}$$

$$p_r = [x_r, y_r]^T \tag{3.2b}$$

Now we can use the following equations to project the observed point into the right and left image planes:

$$x_l = \frac{X + b/2}{Z} = \frac{(u_l - \sigma_u)}{f^*\alpha} \tag{3.3a}$$

$$y_l = \frac{Y}{Z} = \frac{(v_l - \sigma_v)}{f^*} \tag{3.3b}$$

$$x_r = \frac{X - b/2}{Z} = \frac{(u_r - \sigma_u)}{f^*\alpha} \tag{3.3c}$$

$$y_r = \frac{Y}{Z} = \frac{(v_r - \sigma_v)}{f^*} \tag{3.3d}$$

where, f^* is the focal length described in pixel dimension and α is the ratio of the pixel dimensions obtained from the following equation:

$$\alpha = \frac{dy}{dx} \tag{3.4}$$

Therefore, the 3D coordinates of the observed point P with respect to the sensor frame can be calculated using the stereo vision projection equations:

$$X = \frac{b(x_l + x_r)}{2(x_l - x_r)} \tag{3.5a}$$

$$Y = \frac{by_l}{x_l - x_r} = \frac{by_r}{x_l - x_r} \tag{3.5b}$$

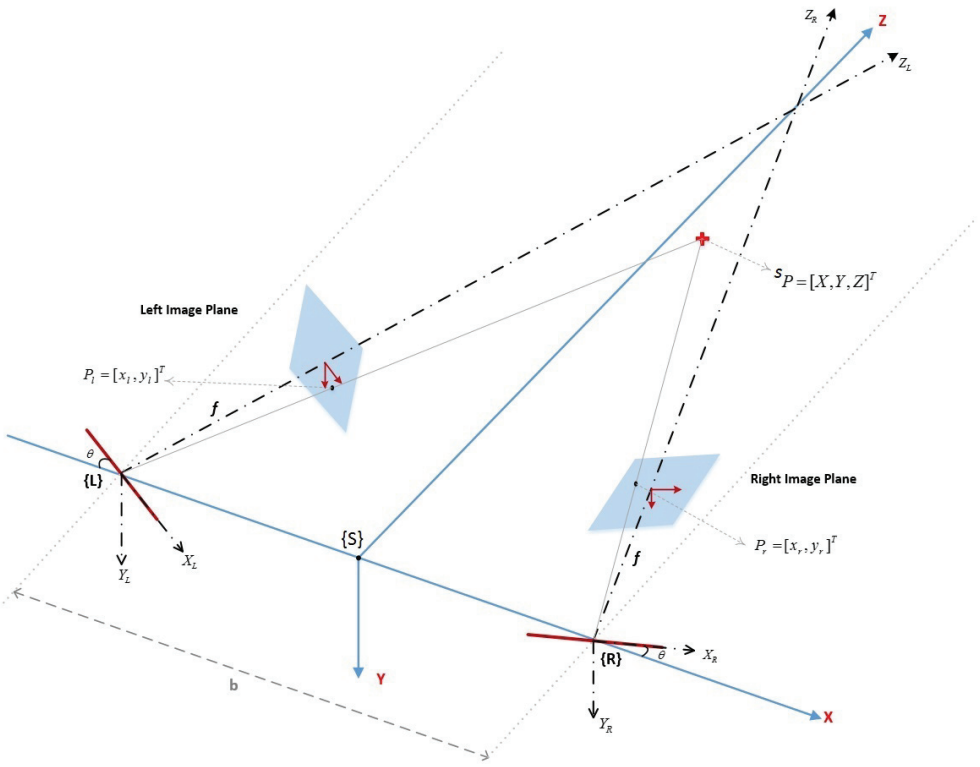


Figure 3.3: Non-parallel stereo vision system

$$Z = \frac{b}{x_l - x_r} \tag{3.5c}$$

The second case for binocular systems is the non-parallel configuration as shown in Figure 3.3. In this configuration the cameras are tilted by θ around Y-axis and the focal points of two cameras are apart at distance $\frac{b}{2}$ with respect to the origin of the sensor frame, $\{S\}$. The image planes for the left and right cameras are located at distance f from the focal points and orthogonal to the optical axis [1].

Considering a point ${}^{S}P = [X, Y, Z]$ observed by both cameras, the perspective camera model can be used to project the point into the right and left image planes:

$$x_l = \frac{X_l}{Z_l} = \frac{(u_l - \sigma_u)}{f^{*}\alpha} \tag{3.6a}$$

$$y_l = \frac{Y_l}{Z_l} = \frac{(v_l - \sigma_v)}{f^*} \tag{3.6b}$$

$$x_r = \frac{X_r}{Z_r} = \frac{(u_r - \sigma_u)}{f^*\alpha} \tag{3.6c}$$

$$y_r = \frac{Y_r}{Z_r} = \frac{(v_r - \sigma_v)}{f^*} \tag{3.6d}$$

where $p_l = [x_l, y_l]^T$ and $p_r = [x_r, y_r]^T$ are the normalized image coordinates from $[u_l, v_l]^T$ and $[u_r, v_r]^T$, f^* is the focal length in pixel dimension and α is the ratio of the pixel dimensions obtained from (3.4).

Using the homogeneous transformation matrix between the frames, the sensor frame can be transformed to the right and left camera frames using the following equations:

$$^{S}P = {}^{S}_{r}H \times {}^{r}P \tag{3.7}$$

where $^{S}P = [X, Y, Z]$ and $^{r}P = [X_r, Y_r, Z_r]$ and ${}^{S}_{r}H$ can be calculated according to the rotation matrix along Y axis which can be calculated as follows:

$${}^{S}_{r}H = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) & \frac{b}{2} \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

and the equations for the left camera can be obtained by:

$$^{S}P = {}^{S}_{l}H \times {}^{l}P \tag{3.9}$$

where ${}^{l}P = [X_l, Y_l, Z_l]$ and ${}^{S}_{l}H$ are calculated from the following equation:

$$ {}^{S}_{l}H = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & -\frac{b}{2} \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10} $$

By substituting (3.8) and (3.10) into (3.7) and (3.9) respectively, we can write the following equations for the right and left cameras:

$$ X = X_r \cos(\theta) - Z_r \sin(\theta) + \frac{b}{2} \tag{3.11} $$

$$ Y = Y_r \tag{3.12} $$

$$ Z = X_r \sin(\theta) + Z_r \cos(\theta) \tag{3.13} $$

$$ X = X_l \cos(\theta) + Z_l \sin(\theta) - \frac{b}{2} \tag{3.14} $$

$$ Y = Y_l \tag{3.15} $$

$$ Z = -X_l \sin(\theta) + Z_l \cos(\theta) \tag{3.16} $$

Substituting X, Y and Z from (3.11), (3.12) and (3.13) into (3.14), (3.15) and (3.16) respectively, the following equations are obtained:

$$Z_r + Z_l = \frac{X_r - X_l}{\tan(\theta)} + \frac{b}{\sin(\theta)} \tag{3.17}$$

$$Y_l = Y_r \tag{3.18}$$

$$Z_r - Z_l = (X_r - X_l)\tan(\theta) \tag{3.19}$$

Using (3.6), the following relations are obtained:

$$\frac{Z_r}{Z_l} = \frac{y_l}{y_r} \tag{3.20}$$

$$\frac{X_r}{X_l} = \frac{x_r y_l}{x_l y_r} \tag{3.21}$$

Using the above equations, θ, ${}^{l}P$ and ${}^{r}P$ can be found as follows:

$$\tan\theta = \frac{y_r - y_l}{x_r y_l + x_l y_r} \tag{3.22}$$

$$X_l = \frac{b}{\sin(\theta)} \frac{x_l y_r (y_r - y_l)}{((y_r^2 - y_l^2)) - ((x_r y_l)^2 - (x_l y_r)^2)} \tag{3.23}$$

$$Z_l = \frac{b}{\sin(\theta)} \frac{y_r (y_r - y_l)}{((y_r^2 - y_l^2)) - ((x_r y_l)^2 - (x_l y_r)^2)} \tag{3.24}$$

$$X_r = \frac{b}{\sin(\theta)} \frac{x_r y_l (y_r - y_l)}{((y_r^2 - y_l^2)) - ((x_r y_l)^2 - (x_l y_r)^2)} \tag{3.25}$$

$$Z_r = \frac{b}{\sin(\theta)} \frac{y_l(y_r - y_l)}{((y_r^2 - y_l^2)) - ((x_r y_l)^2 - (x_l y_r)^2)} \tag{3.26}$$

$$Y_l = Y_r = \frac{b}{\sin(\theta)} \frac{y_r y_l(y_r - y_l)}{((y_r^2 - y_l^2)) - ((x_r y_l)^2 - (x_l y_r)^2)} \tag{3.27}$$

By substituting X_l, Y_l, Z_l and θ into (3.9) and simplifying the equations, the exact position of the point with respect to sensor frame, ${}^S P$, can be obtained as follows:

$${}^S P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \frac{b}{2} \frac{(x_r y_l - x_l y_r)^2 + (y_r - y_l)^2}{(y_r^2 - y_l^2) - ((x_r y_l)^2 - (x_l y_r)^2)} \\ \frac{b y_r y_l (x_r y_l - x_l y_r) \left(\frac{(y_r - y_l)^2}{(x_r y_l - x_l y_r)^2} + 1 \right)^{1/2}}{(y_r^2 - y_l^2) - ((x_r y_l)^2 - (x_l y_r)^2)} \\ \frac{b y_r y_l (x_r + x_l)}{(y_r^2 - y_l^2) - ((x_r y_l)^2 - (x_l y_r)^2)} \end{bmatrix} \tag{3.28}$$

By finding the position of the each target by triangulation method, the orientation of a model which consists of at least 4 targets, can be calculated. Figure 3.4 shows a model which consists of four targets. Frame $\{M\}$ is the known model frame which is attached to the created model. Since the model is considered to be rigid, the coordinates of each target with respect to the model frame is known and fixed.

$${}^M P_i = [x_i, y_i, z_i]^T \tag{3.29}$$

where ${}^M P_i$ represents the position of target i with respect to frame $\{M\}$.

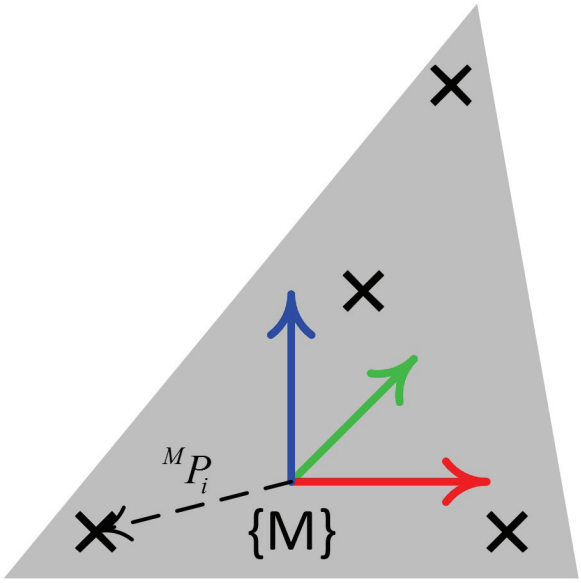


Figure 3.4: A model created using 4 targets

In addition, according to Figure 3.5, the position of each target with respect to the camera frame $\{S\}$ can be obtained using (3.28) or (3.5).

$$ {}^{S}P_i = [X_i, Y_i, Z_i]^T \tag{3.30} $$

the transformation matrix between the model frame $\{M\}$ and the camera frame $\{S\}$ can be calculated using the following equation:

$$ {}^{S}P_i = {}^{S}_{M}H \times {}^{M}P_i \tag{3.31a} $$

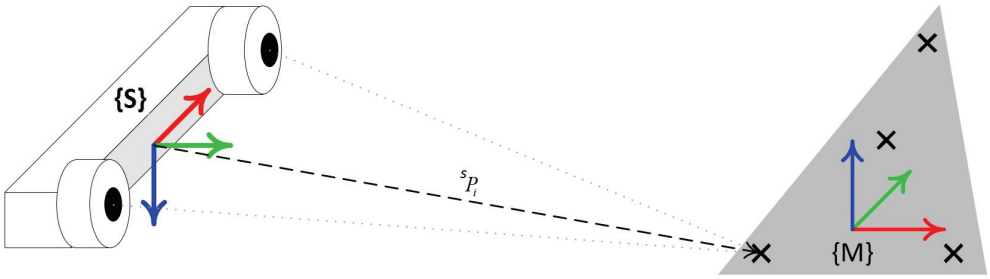


Figure 3.5: Obtaining orientation using binocular visual system

$$ {}^{S}_{M}H = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{S}_{M}R_{3\times3} & {}^{S}_{M}t_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{3.31b} $$

where ${}^{S}_{M}R$ is the rotation matrix between model frame and camera frame and ${}^{S}_{M}t$ is the coordinates of the origin of model frame in the camera frame. Substituting the coordinates of 4 targets in (3.31), the 12 unknown parameters of ${}^{S}_{M}H$ can be obtained.

$$ \begin{bmatrix} {}^{S}X_i \\ {}^{S}Y_i \\ {}^{S}Z_i \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} {}^{M}x_i \\ {}^{M}y_i \\ {}^{M}z_i \\ 1 \end{bmatrix} \tag{3.32} $$

Once ${}^{S}_{M}H$ is determined, the orientation of frame $\{M\}$ with respect to frame $\{S\}$ can be extracted from the rotation matrix ${}^{S}_{M}R$. According to [15], the rotation matrix between two

frames based on the Euler angles is given by:

$$
{}^{S}_{M}R = \begin{bmatrix} c(R)c(P) & c(R)s(W)c(P) - s(R)c(W) & c(R)c(W)s(P) + s(R)s(W) \\ s(R)c(P) & s(R)s(W)s(P) + c(R)c(W) & s(R)c(W)s(P) - c(R)s(W) \\ -s(P) & c(P)s(W) & c(P)c(W) \end{bmatrix} \tag{3.33}
$$

where $c(R) = \cos(R)$ and $s(R) = \sin(R)$. By comparing (3.33) with the obtained ${}^{S}_{M}H$, the Euler angles which represent the orientation of the model frame with respect to the camera frame can be calculated.

The value of $\cos(P)$ can be obtained by taking square root of the sum of the squares of r_{11} and r_{21}. According to [15], as long as $\cos(P) \neq 0$, the general solution for W-P-R can be found by:

$$P = \arctan 2(-r_{31}, \sqrt{r_{11}^2 + r_{22}^2}) \tag{3.34a}$$

$$R = \arctan 2(\frac{r_{21}}{\cos(P)}, \frac{r_{11}}{\cos(P)}) \tag{3.34b}$$

$$W = \arctan 2(\frac{r_{32}}{\cos(P)}, \frac{r_{33}}{\cos(P)}) \tag{3.34c}$$

As mentioned earlier, this solution is valid as long as $P \neq \pm 90$. If $\cos(P) = 0$ then we can only compute the summation or difference of W and R [15]. In this situation R is considered to be zero and W can be calculated as follows:

$$R = 0 \tag{3.35a}$$

$$P = -r_{31} \times \frac{\pi}{2} \tag{3.35b}$$

$$W = -r_{31} \times \arctan 2(r_{12}, r_{22}) \tag{3.35c}$$

This algorithm is also used to find the orientation offset values for the pose correction strategy which will further be explained in Section 5.3.

3.2 Optical Measurement Sensor C-Track

The processing of the images and the pose estimation algorithm using regular cameras are time consuming. Additionally, the accuracy of the estimated pose depends on the camera's parameters. These issues affect the information flow and cause delays in the performance of the visual servoing task. The time delay is one of the main causes for difficulties in the real-time correction process. Thus, using a sensor that is able to provide accurate pose of the objects dynamically can help the system avoid such problems.

Optical coordinate measuring machines (CMMs) are the sensors which use image-based triangulation to calculate the pose of the model. These sensors consist of accurate video cameras providing dynamic tracking capabilities. C-Track, a commercial product of Creaform, is a dual camera optical CMM sensor which measures the position of the reflector targets in its workspace by performing continuous image acquisition at the rate of maximum $29\,Hz$. Figure 3.6 shows the C-Track used in this research.

There are three main calculation steps to implement the dynamic referencing and optical measurement functionalities for optical CMMs. The first step is to accurately assess the image projections in the stereo sensor images by an image processing procedure. The next step is to estimate the target coordinates (position of each target) with respect to the sensor reference frame using the two images of the stereo vision. This step is performed by triangulation method explained in Section 3.1. The last step is to extract the pose of the models created by the targets whose positions are known by the sensor cameras [42].

Figure 3.6: Dual camera sensor C-Track

As mentioned earlier, in order to estimate the pose of the models, it is important to know the model of the sensor's cameras. According to [42], the cameras can be modelled using perspective projections and other parameters that take into account geometric aberrations generated by the camera, commonly known as radial and tangential distortions. These parameters also known as camera intrinsic parameters include the focal length, image sensor format, and principal point which can be found through a calibration procedure.

Camera calibration is the process of using a standard bar with several reflector targets at known distances, also known as calibration bar, to identify the camera parameters. Figure 3.7 shows the Creaform calibration bar. The operator can use this bar in different configurations within the workspace of the CMM and then the intrinsic parameters are identified automatically according to the obtained positions of the reflector targets on the calibration bar [43]. As an example, the calibration of the C-Track is performed by the VXelements software and the standard scale bar provided by the Creaform. C-Track calibration procedure is shown in Figure 3.8. Since the camera parameters have a significant effect on

Figure 3.7: Camera calibration scale bar

the measurements and accuracy of the sensor, it is recommended that the operator repeats the camera calibration regularly. This will guarantee consistent precision during the entire lifecycle of the product [40].

C-Track can be used for dynamic measurements and probing inspection (associating with HandyProbe). In addition, it can be used to measure the pose of several reference models (created by several targets) simultaneously, continuously and with high accuracy.

One of the key advantages of the C-Track compared to other optical measurement sensors such as laser-trackers, is the lower price. Moreover, compared to the laser-trackers which need high-skilled technicians and specific trainings, C-Track is easier to operate. The measuring volume of the C-Track varies from 3.8 to 14.8 m^3 [40].

In this project, C-Track is used as the measurement sensor because of the on-line tracking capabilities, portabilities and low price compared to laser trackers [20]. Using the C-Track as the feedback sensor in the position based visual servoing control system, the pose of the robot tool is extracted and compared with the desired pose through the developed Dynamic Pose Correction Software (DPC software) which is addressed in Section 5.2.

3.3 VXelements Software

VXelements is the software provided by Creaform for communicating with the manufactured optical CMMs. This software consist of different modules for different applications including

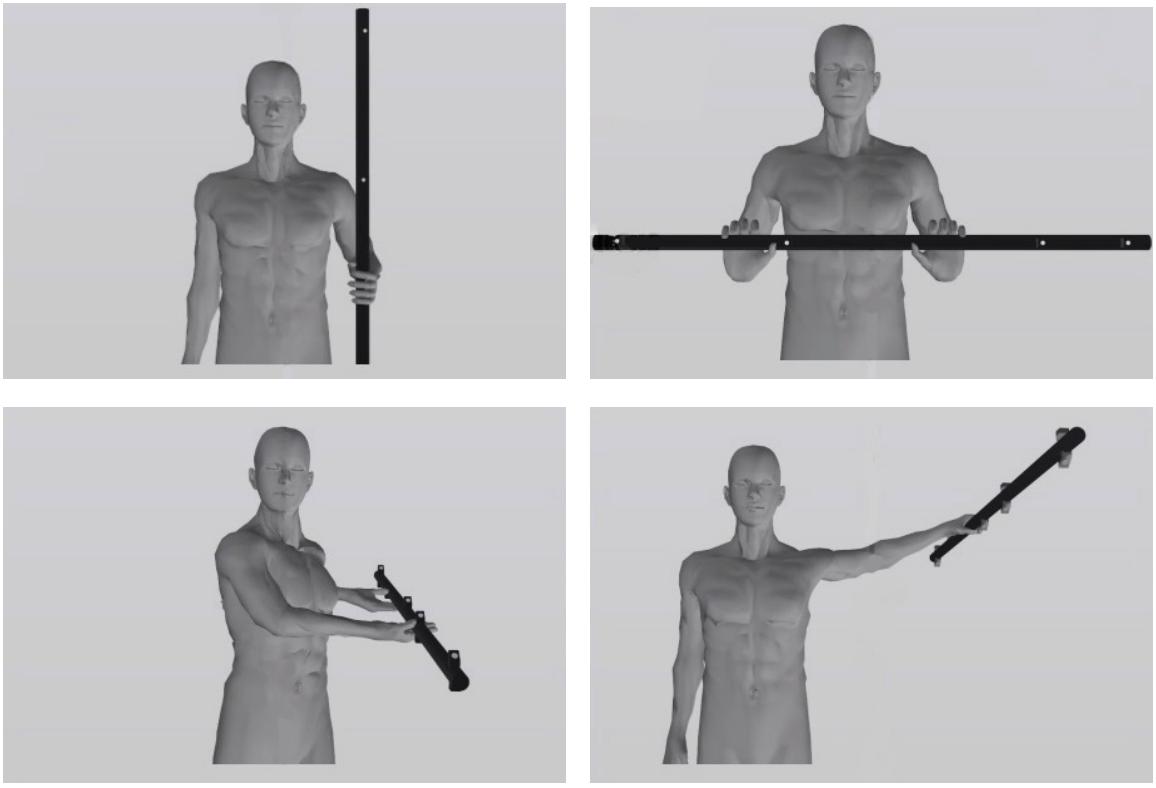


Figure 3.8: C-Track calibration procedure

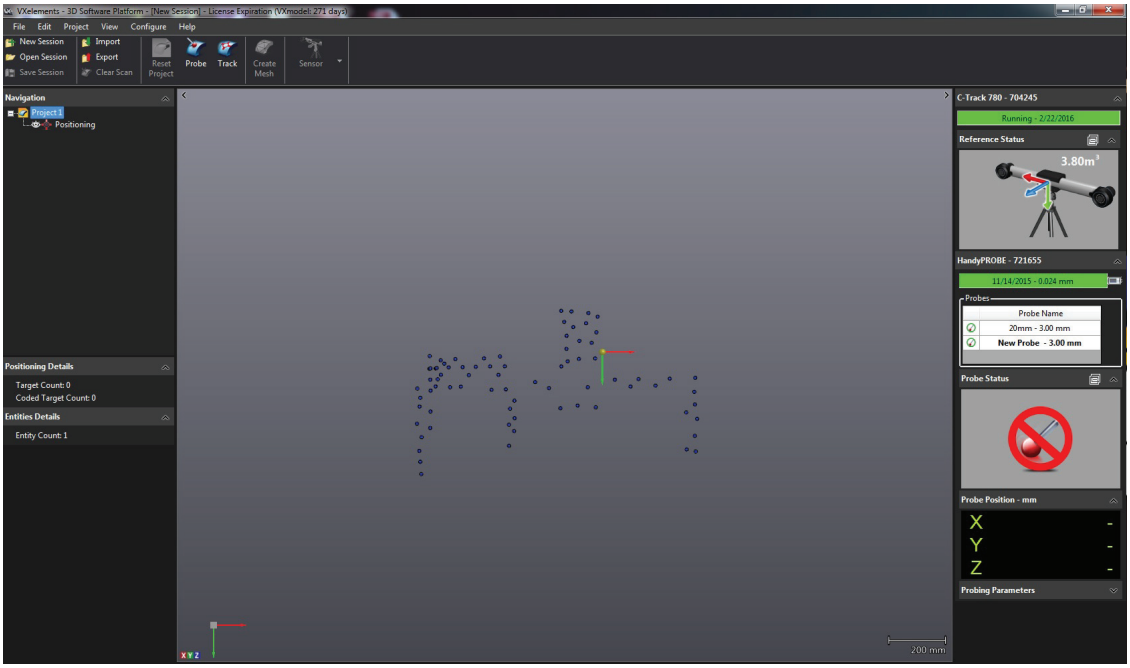
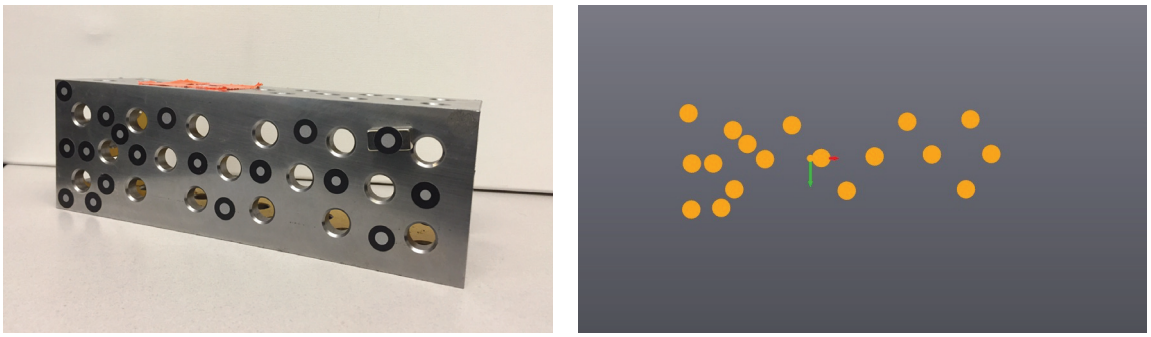


Figure 3.9: VXelements software environment

probing, scanning and tracking associating with HandyProbe, MaxSHOT and C-Track respectively. In this project the probing and tracking modules are mostly used. The software environment is shown in Figure 3.9.

Through the software, the user can calibrate C-Track and HandyProbe, as well as probing the pre-defined entities such as points, axes, planes and circles. Using this software, it is also possible to create models and dynamic references from the optical reflectors seen by the camera. The user can also assign reference frames to the created models, track the models and record the real-time data when the models are moving in the workspace of the C-Track.

Dynamic reference is used to eliminate handling errors specially in the environments with vibrations. This reference also make it possible to move the C-Track without affecting the precision of the measurements. The probe module, provides the environment where the HandyProbe is used to create different standard entities. This module is used in this research to identify the robot original tool frame which is explained in Section 5.1.1.

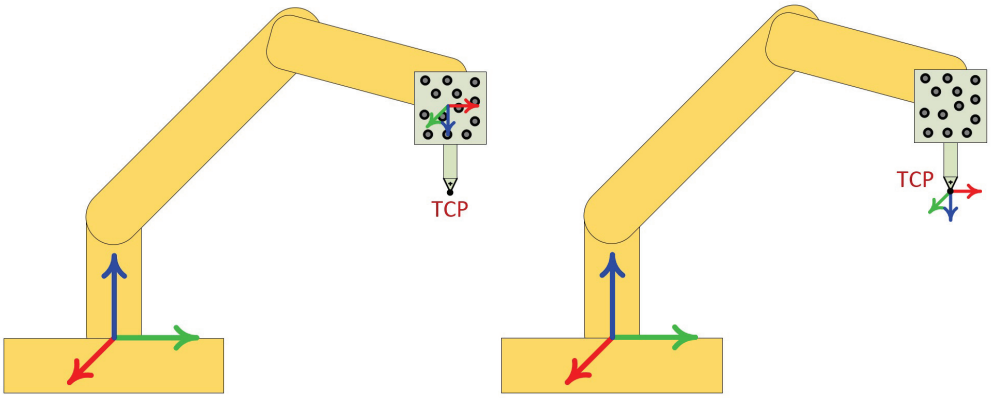


(a) (b)

Figure 3.10: Creating model using the software (a) object with reflector targets and (b) created model in the software

Each model consists of at least four reflector targets. As can be seen in Figure 3.10 the model of the cube is created in VXelements using 18 reflector targets. After creating a model, the software calculates the geometric center of the targets and assigns a frame to the created model. This frame, also known as "Origin-Offset", is used to calculate the position and orientation of the model with respect to the sensor frame as explained in Section 3.1. As long as the C-Track observes the model, the position and orientation of this frame can be measured by the C-Track.

The origin-offset of a model can be modified by the user. Since each model is identified by its origin-offset, defining an appropriate one is essential. For instance, in robotic applications, the tool frame of the robots are defined at the TCP which is not necessarily at the center of reflector targets. Therefore, it is important to modify and change the origin-offset of the tool model from target's center to the TCP as shown in Figure 3.11. In Section 5.1.1, the detailed procedures for creating the accurate origin-offset of the tool model at TCP is explained.



(a) (b)

Figure 3.11: Tool model origin-offset (a) at center of targets (by default) (b) at TCP

3.4 C-Track Data Analysis

One of the major drawbacks of the C-Track is the relative low accuracy of the measurements compared to laser trackers. Moreover, the noise associated with the measurements affect the pose calculation procedure. Figure 3.12 shows the position of the robot end-effector measured by the C-Track for 34 seconds (almost 1000 measurements) when the robot is fixed at a specific position in front of the cameras.

As can be seen in Figure 3.12, the peak to peak difference for X, Y, Z coordinates are approximately 0.06 , 0.05 and 0.12 mm respectively. Therefore, achieving accuracy better than 0.12 mm would be impossible with the current measurement. Since the desired precision in our experiments (0.05 mm) is higher than the C-Track measurement precision, a solution must be considered to overcome this issue.

In order to analyze the measured data taken from the C-Track, the first 100 measurements is used. Figure 3.13 shows the data for the first 3.4 seconds.

Root mean square (RMS) is one of the techniques which is used to remove the noise from

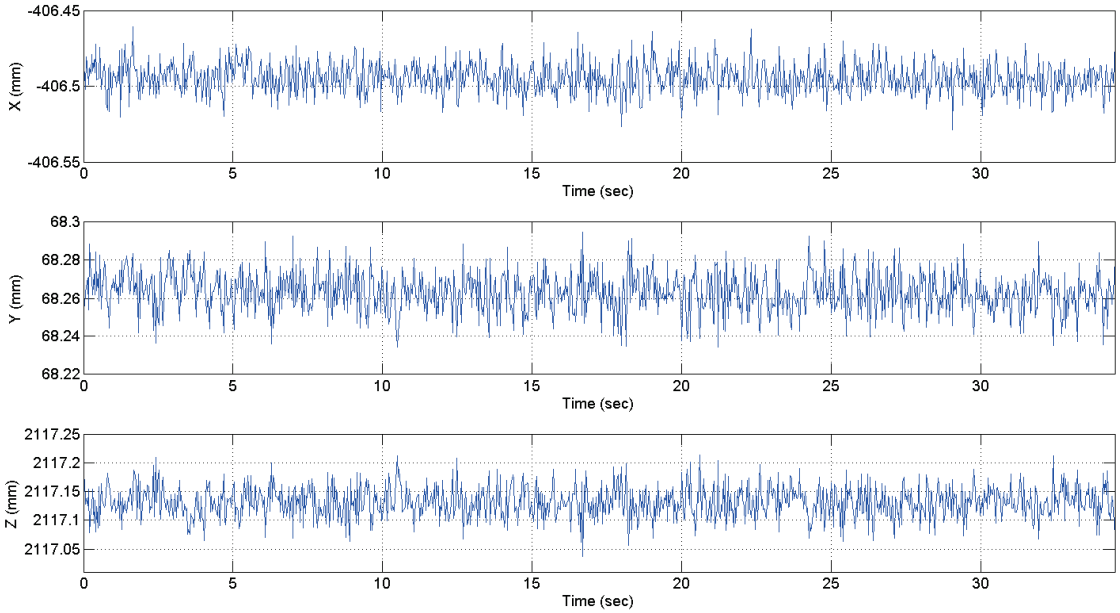


Figure 3.12: Position measurements taken from a fixed model by C-Track

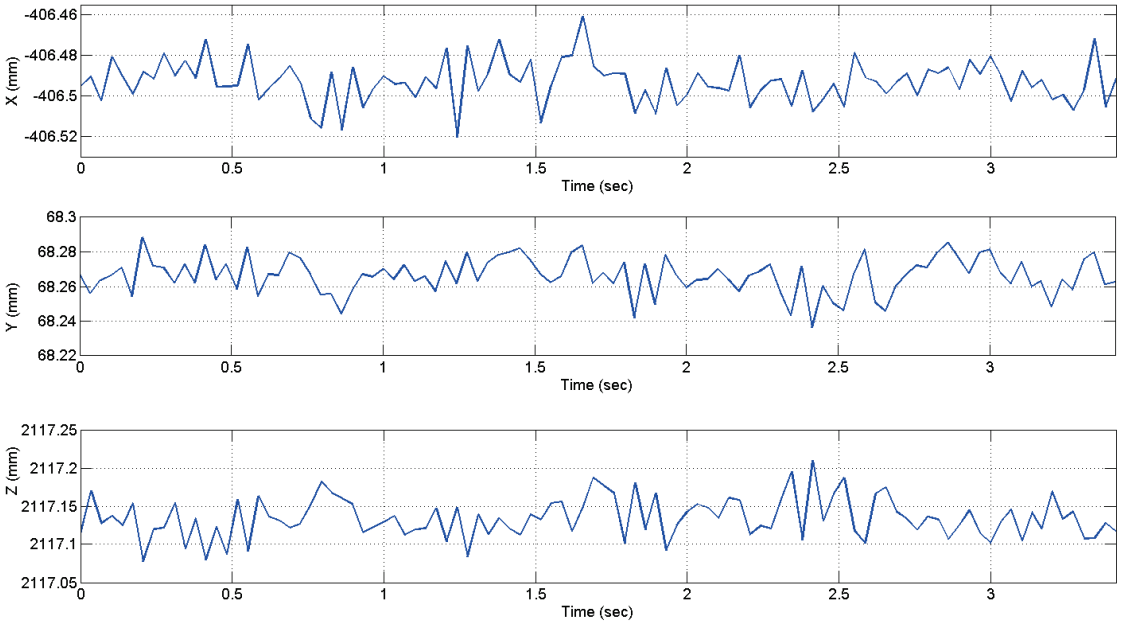


Figure 3.13: Selecting 100 measurements from position of a fixed model measured by C-Track

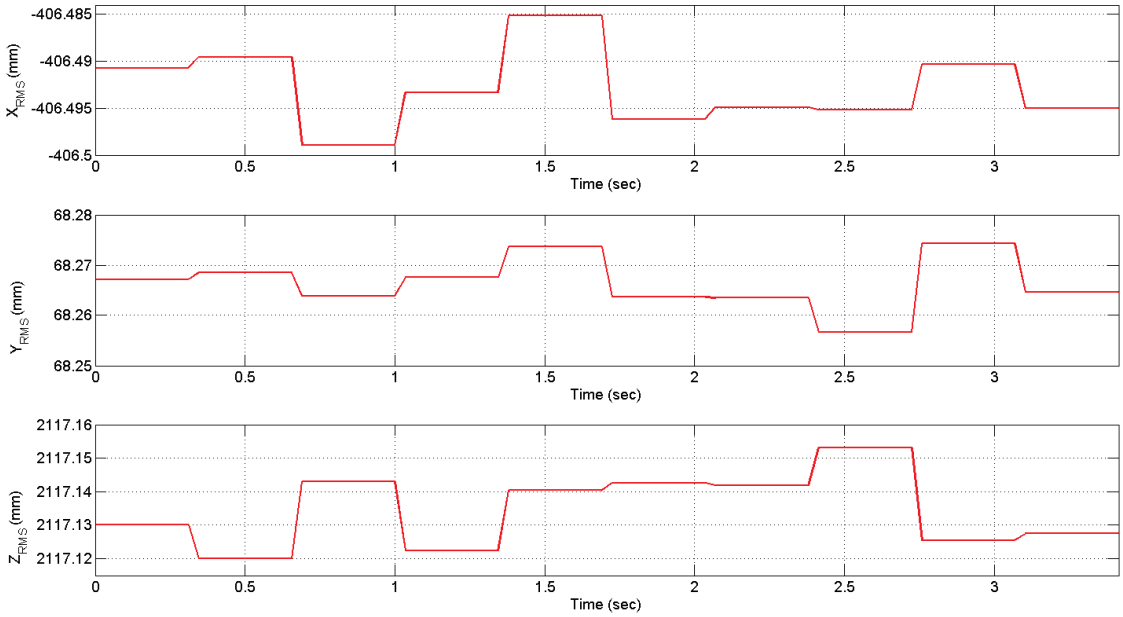


Figure 3.14: RMS value of the first 100 measurements

the C-Track measurements. The RMS value of the model pose measured by C-Track can be obtained by the following equation:

$$\vec{P}_{RMS} = sgn(\vec{P}_i)\sqrt{\frac{\sum_{i=1}^{N} \vec{P}_i^2}{N}} \tag{3.36}$$

where $\vec{P}_i$ is the pose of the model measured by the C-Track at each interval, N is the RMS length and $\vec{P}_{RMS}$ is the calculated pose of the model based on the RMS value of the N measurements.

Although each measurement is different from the previous one, it is proved that the difference among RMS values of each 10 measurements is negligible. Figure 3.14 shows the RMS values of the same 100 measurements.

As shown in the Figure 3.14 the maximum deviation of RMS values is 0.03 on Z direction which is 75% improvement in the C-Track measurements. This measurement technique

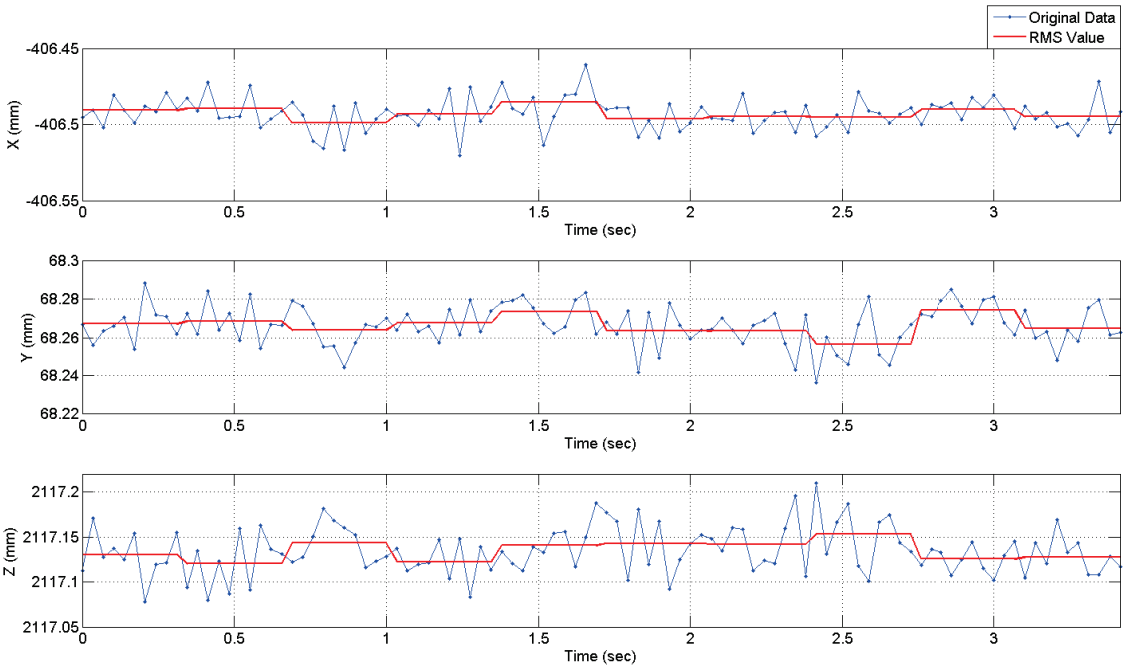


Figure 3.15: Comparison of original measurement data and RMS

provides sufficient accuracy for the measured data and can be used as the feedback signal. Figure 3.15 shows both original data and RMS together for the first 100 measurements.

It is important to mention the RMS calculations can not be used while the robot is moving since the RMS data might not be reliable. In this case, a possible solution is to use a filter for removing the noise from the C-Track measurements.

The RMS length in our experiments is considered to be 10, i.e. after taking 10 measurements by the C-Track, which takes 0.344 seconds, the RMS value of X-Y-Z-W-P-R coordinates are calculated. Through the developed software, this method is implemented by a function which takes 10 measurements from the C-Track and calculates the root mean square value of the data.

The calculated RMS values are compared with the desired pose of the robot and the difference is used as the controller input. The controller provides the actuating signal which is applied to the robot as position and orientation offsets along X, Y, Z directions. In the

software a function is considered for *Wait* after applying offsets to make sure the robot consumes the whole offsets applied. After that the new C-Track measurements are used to calculate RMS value of the new pose. This procedure repeats until the required precision in all directions is satisfied.

3.5 Summary

In this chapter, the methods and algorithms for estimating the pose of an object from the dual camera sensor are presented and mathematically discussed. C-Track is introduced as the 6D optical measurement sensor used in this research which provides dynamic measurements of the position and orientation of the objects. The calibration procedure, the cons and pros of the C-Track, as well as the brief introduction to the VXelements software are outlined. The proposed solution to improve the accuracy of the C-Track measurements is also introduced at the end of this chapter.

Chapter 4

Dynamic Pose Correction (DPC) Controller

After introducing the principles of the pose estimation using the binocular visual system, one can use the measured pose as the feedback signal in the DPC controller to complete the position based visual servoing task.

The first part of this chapter is dedicated to the introduction of the Dynamic Path Modification (DPM) option for Fanuc robot followed by the DPC controller design. In addition, the simulation results are presented to prove the feasibility of the proposed pose control algorithm on the Fanuc M20-iA robot.

4.1 Fanuc Dynamic Path Modification Option

In position-based visual servoing, an external sensor is used to estimate the pose of the end-effector. The estimated pose is compared with the desired pose and the difference is used as the controller input. The output signal from the controller is fed into the robot controller as the offsets along the axes X, Y, Z for position and orientation. Figure 4.1 illustrates the

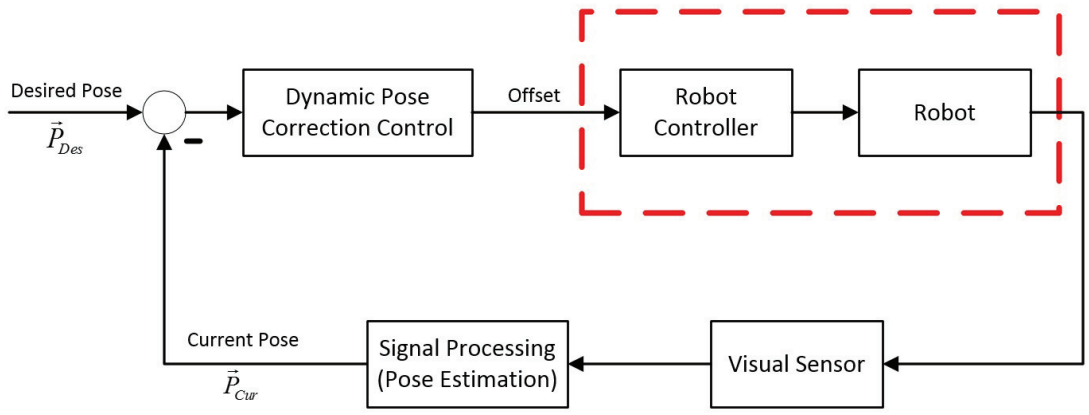
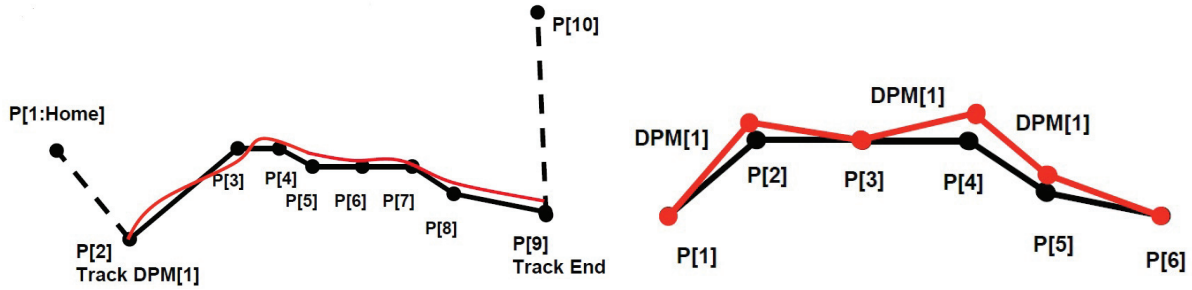


Figure 4.1: Block diagram of the PBVS



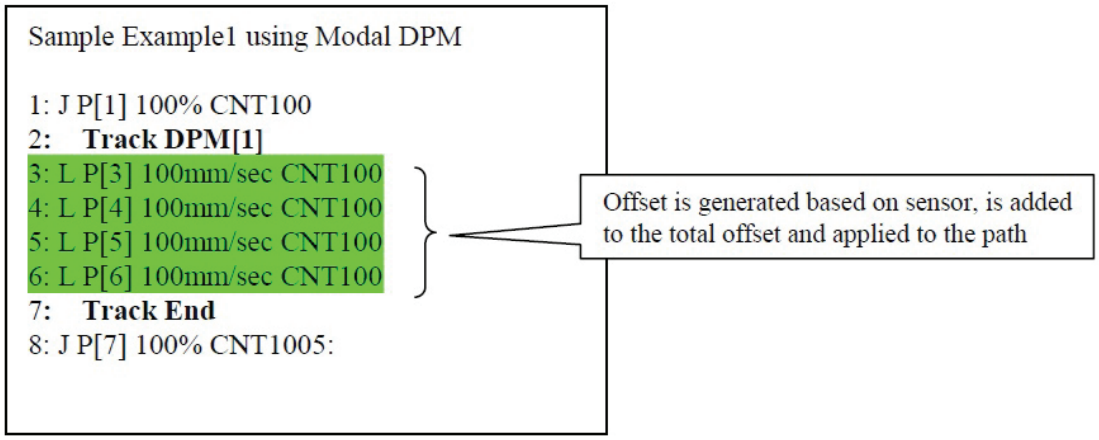
(a) (b)

Figure 4.2: Two different DPM modes (a) Modal DPM (b) Inline DPM

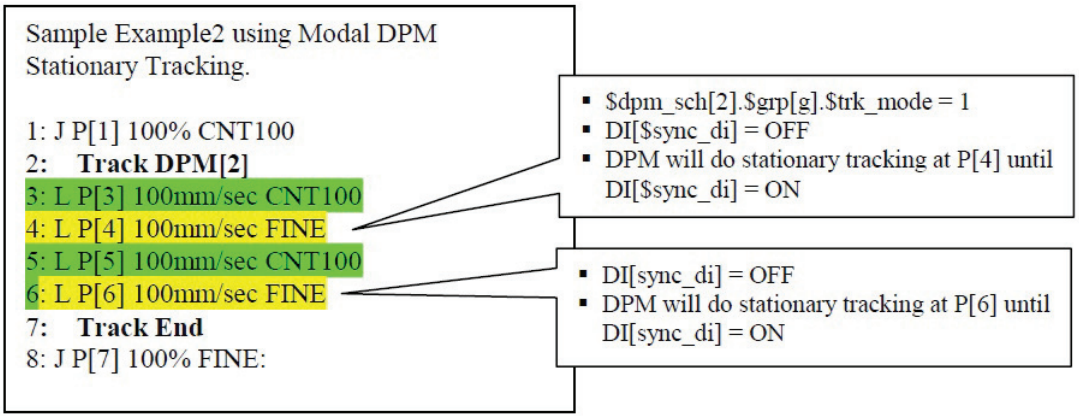
principles of the position based visual servoing system used in this research.

Dynamic Path Modification (DPM) option offered by Fanuc, allows the operator to modify the pose of the robot end-effector dynamically. Using this option, dynamic corrections can be applied to the robot through a third party interface. According to the Fanuc documentations, the corrections can be applied along the entire path or only at the destination position for each segment of the motion, known as Modal DPM and Inline DPM respectively [11]. Figure 4.2 illustrates the mentioned modes of the DPM.

Using the DPM, the operator can apply offsets with respect to different reference frames including user frame, tool frame and along the path of the robot trajectory. In addition to the



(a)



(b)

Figure 4.3: (a) TP program example using modal DPM, (b) TP program example using stationary tracking option

mentioned built-in modes for DPM, stationary tracking is an additional option which allows the operator to apply corrections while the robot is stopped at one specific pose. This mode needs synchronization signals to start and finish correction. As long as the synchronization signal has not been changed, the controller can not move to the next line of the program. Figure 4.3 shows two examples of TP program using DPM option.

It is worth mentioning that other robot manufacturing companies provide the same option as DPM. For instance, External Guided Motion (EGM) is an option offered by ABB which enables the external sensors to control the robot. Using this service, the operator can input absolute positions and modify the path of the robot according to the sensor information at every $4\,ms$ [9].

In this research, the modal DPM is used and the corrections are applied with respect to the user frame. The details of the strategy are presented in Chapter 5.

4.2 Controller Design

The PBVS controller used in this project is a PID controller. A PID controller is a non-model based feedback controller, which is robust to the noise and uncertainties. PID controller tries to eliminate the difference between the desired input and measured output.

As shown in Figure 4.1, in the PBVS system, the feedback sensor provides the current pose of the robot end-effector, $\vec{P}_{Cur}$, and the user sets the desired pose, $\vec{P}_{Des}$. The controller input is the error signal which is the difference between the current pose and desired pose of the robot TCP.

$$\Delta\vec{P} = \vec{P}_{Des} - \vec{P}_{Cur} \tag{4.1}$$

where $\Delta\vec{P}$ is the error consisting of the translations along X,Y,Z axes for position and rotation about X,Y,Z axes for orientation in Cartesian space. The generated error signal is

fed into the PID controller. The general equation for a PID controller is

$$u(t) = K_p \Delta\vec{P} + K_i \int_0^t \Delta\vec{P} d\tau + K_d \frac{d}{dt} \Delta\vec{P} \tag{4.2}$$

The coefficients K_p, K_i, K_d in (4.2) are the proportional, integral and derivative gains respectively, which have to be tuned to obtain the desired performance of the system. The output of the controller, $u(t)$, is the actuating signal sent to the controlled system, i.e. the industrial robot.

In this research, the actuating signal is the offset along each axis. The offsets can be directly applied to the robot using the DPM option and guide the robot TCP to the desired pose through the real-time process.

$$u(t) = \begin{bmatrix} \delta X \\ \delta Y \\ \delta Z \\ \delta W \\ \delta P \\ \delta R \end{bmatrix} \tag{4.3}$$

4.3 System Modeling and Simulation

In order to evaluate the performance of the designed controller and feasibility of the proposed method for dynamic pose correction, the proposed PBVS and the kinematic model of robot are implemented in Matlab Simulink prior to applying to the real industrial robot. In order to simulate the manipulators, the kinematic model of the robot is required. Here we assume that the robot system has an ideal inner joint position control loop. Hence the robot dynamics

Table 4.1: D-H table of Fanuc M20-iA

i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0.0	0.0	525.0	θ_1
2	150.0	-90.0	0.0	θ_2
3	790.0	0.0	0.0	θ_3
4	250.0	-90.0	835.0	θ_4
5	0.0	90.0	0.0	θ_5
6	0.0	-90.0	100.0	θ_6

is not considered in this research. The kinematic model of a serial robot is obtained using the Denavit-Hartenberg parameters (D-H parameters).

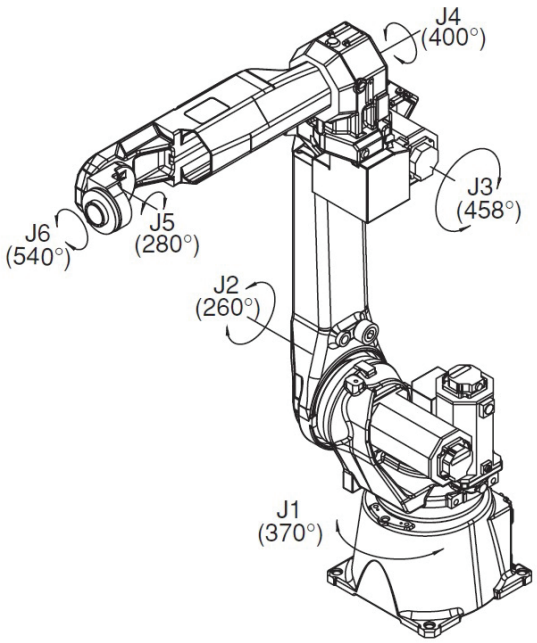
The D-H parameters of the FANUC M20-iA are calculated based on the drawings of the robot (Figure 4.4) by assigning the standard frames at each joint based on [15]. According to the documentations of the FANUC M20-iA, the base frame is located at the center of rotation of joint one and at same level of joint two. The D-H parameters of the FANUC M20-iA robot are shown in Table 4.1.

According to [15], the transformation matrix, ${}_{i}^{i-1}H$ which defines frame $\{i\}$ relative to frame $\{i-1\}$ can be obtained using the D-H parameters and the following equation:

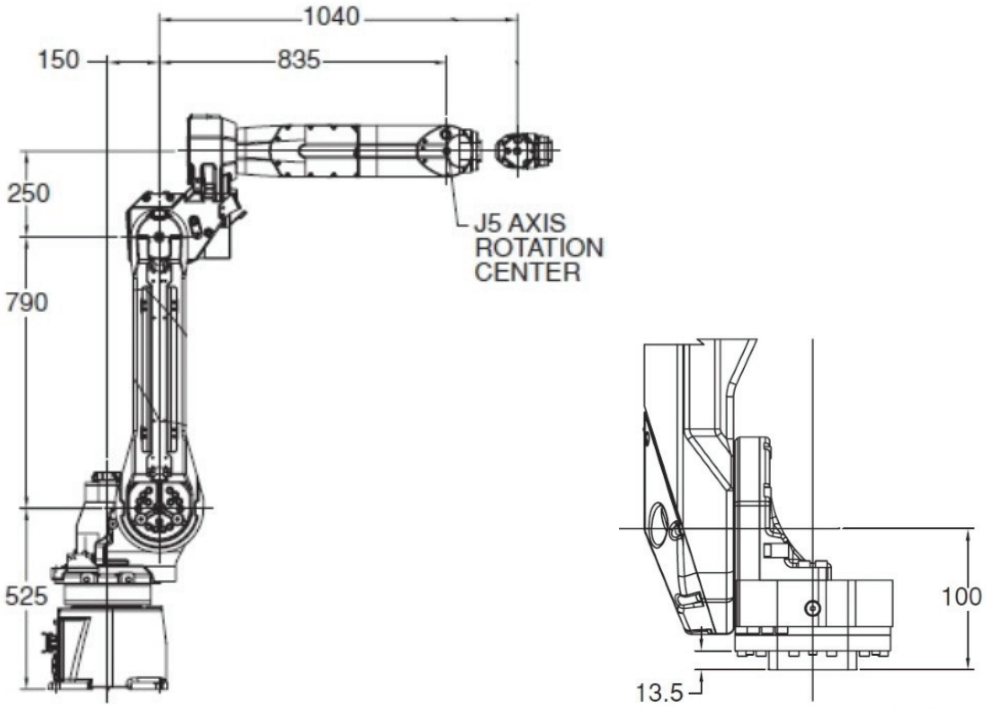
$$ {}_{i}^{i-1}H = R_X(\alpha_{i-1})D_X(a_{i-1})R_Z(\theta_i)D_Z(d_i) \tag{4.4} $$

where

$$ R_X(\alpha_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & 0 \\ 0 & \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5a} $$



(a)



(b) (c)

Figure 4.4: Fanuc M20-iA (a) Isometric view, (b) 2D sketch of the side and (c) 2D sketch of the wrist

$$D_X(a_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5b}$$

$$R_Z(\theta_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5c}$$

$$D_Z(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5d}$$

By substituting (4.5) in (4.4), the following homogeneous transformation matrix between two frames is achieved:

$${}^{i-1}_{i}H = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

where $c\theta_i$ and $s\theta_i$ represent $\cos\theta_i$ and $\sin\theta_i$ respectively. The forward kinematic chain equation which defines the relation between the robot end-effector (joint 6) and the base frame

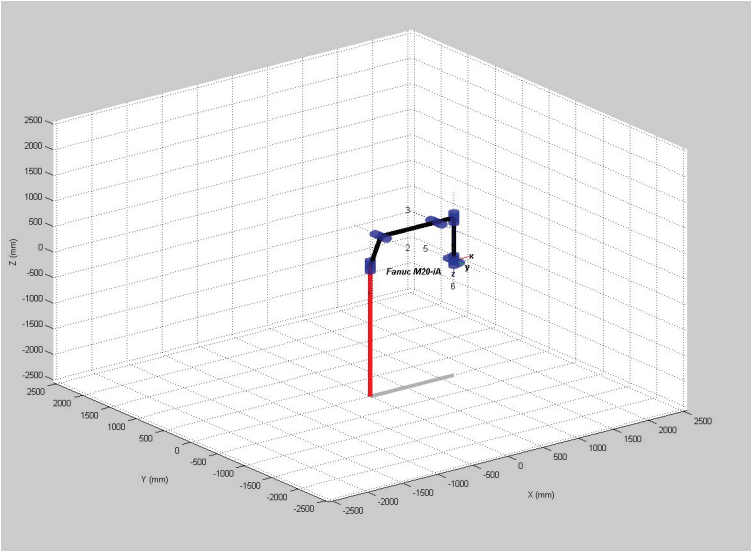


Figure 4.5: Fanuc M20-iA 3D model

of the robot can be calculated using the following equation:

$$ {}^{0}_{6}H = {}^{0}_{1}H\,{}^{1}_{2}H\,{}^{2}_{3}H\,{}^{3}_{4}H\,{}^{4}_{5}H\,{}^{5}_{6}H \tag{4.7} $$

where ${}^{0}_{6}H$ specifies the position and orientation of the end-effector with respect to the base of the robot.

After finding the D-H parameters of the robot, the kinematic model of the robot is implemented in Matlab and the forward kinematic equations and Jacobian matrix are obtained from (4.7) and using the Robotic Toolbox [44]. The 3D model of the robot created in Matlab is shown in Figure 4.5.

Figure 4.6 shows the Simulink block diagram for the simulation. As shown in Figure 4.6, the current pose of the robot is compared with the desired pose at each time interval ($34.48\,ms$), and the difference is fed into the PID controller block.

According to Section 4.2, the output of the controller is the offsets given to the robot controller. Since the calculated offsets are in Cartesian space, they need to be converted into

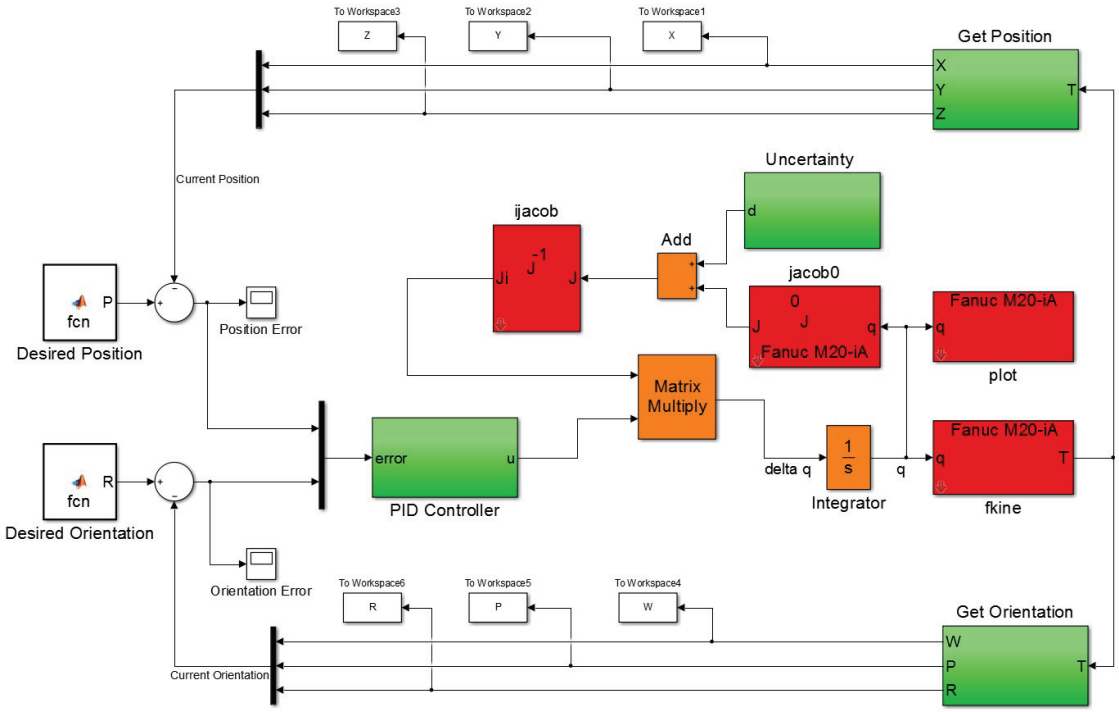


Figure 4.6: Block diagram of the Simulation

the joint space in order to be applied to the robot. This conversion is implemented using the Jacobian matrix of the robot.

The Jacobian matrix of a given manipulator is defined as a $6 \times n$ matrix that relates the joint space velocity and Cartesian space velocity.

$$^{O}\dot{\vec{P}}_{6\times 1} = {}^{O}J(q)_{6\times n}.\dot{\vec{q}}_{n\times 1} \tag{4.8}$$

where n is the number of the joints of the manipulator, $\dot{P}$ is the linear and angular velocity of the robot end-effector with respect to the base of the robot (frame $\{O\}$), $\dot{q}$ is the velocity of each joint and $J(q)$ is the Jacobian matrix of the robot. The superscript O to $\dot{P}$ and $J(q)$ is to emphasize that both of them are expressed with respect to the base frame of the robot.

As explained in Section 1.1.1, Fanuc M20-iA has 6 rotary joints ($n = 6$). Multiplying

both sides of the (4.8) by inverse of Jacobian matrix, J^{-1}:

$$\dot{\vec{q}}_{6\times1} = J^{-1}_{6\times6}.\dot{\vec{P}}_{6\times1} \tag{4.9}$$

which can be expressed in the following form:

$$\Delta\vec{q}_{6\times1} = J^{-1}_{6\times6}.\Delta\vec{P}_{6\times1} \tag{4.10}$$

By introducing the above-mentioned PID controller, the final offset sent to the joints is obtained as:

$$\Delta\vec{q}_{6\times1} = J^{-1}_{6\times6}.[K_p\Delta\vec{P} + K_i\int_0^t \Delta\vec{P}d\tau + K_d\frac{d}{dt}\Delta\vec{P}] \tag{4.11}$$

Therefore, by using the Jacobian matrix, the calculated offsets in the Cartesian space can be converted into the joint offsets. The robot controller uses the built-in dynamic equations to actuate each joint which results in the dynamic correction of the end-effector.

In order to show this procedure in Simulink, the integral block is used to add the calculated Δq to the initial joint angles of the robot and guide the robot to the desired pose according to the calculated difference. The Jacobian matrix of the robot is calculated according to the joint angles of the robot in each iteration. The *fkine* block consist of the forward kinematic chain equations which calculates the transformation matrix between the last joint and the base frame, ${}^0_6H$, using each joint angle q_i.

It is worth mentioning that due to the limitations of simulating the C-Track, *Get Position* and *Get Orientation* blocks are used to extract the position and orientation of the robot from the transformation matrix. The uncertainty block is used to affect the original Jacobian matrix of the robot. Using this method, the calculated joint angles for a specific position would be different from the ideal experiment. This technique is used to model the robot

Table 4.2: PID controller gains

	K_p	K_i	K_d
X	1.45	0.7	0.3
Y	1.45	0.9	0.3
Z	1.45	0.9	0.3
W	1.3	0.02	0.25
P	1.6	0.02	0.25
R	1.5	0.02	0.25

inaccuracies in the simulation. This block generates random numbers in a 6×6 matrix that is added to the Jacobian matrix. The purpose of this simulation is to assess the effect of the PID controller on eliminating the error between the desired pose and the current pose of the robot end-effector while the uncertainty is added to the Jacobian matrix at each time interval. The control interval and sampling interval are set to $34.48\,ms$ which is the same as the C-Track sampling rate. The controller gains are presented in Table 4.2. The desired pose and uncertainty are as follows:

$$\vec{P}_{Des} = \begin{bmatrix} X \\ Y \\ Z \\ W \\ P \\ R \end{bmatrix} = \begin{bmatrix} 579.7 \\ 165.0 \\ -300.0 \\ 60 \\ -60 \\ 45 \end{bmatrix} \tag{4.12a}$$

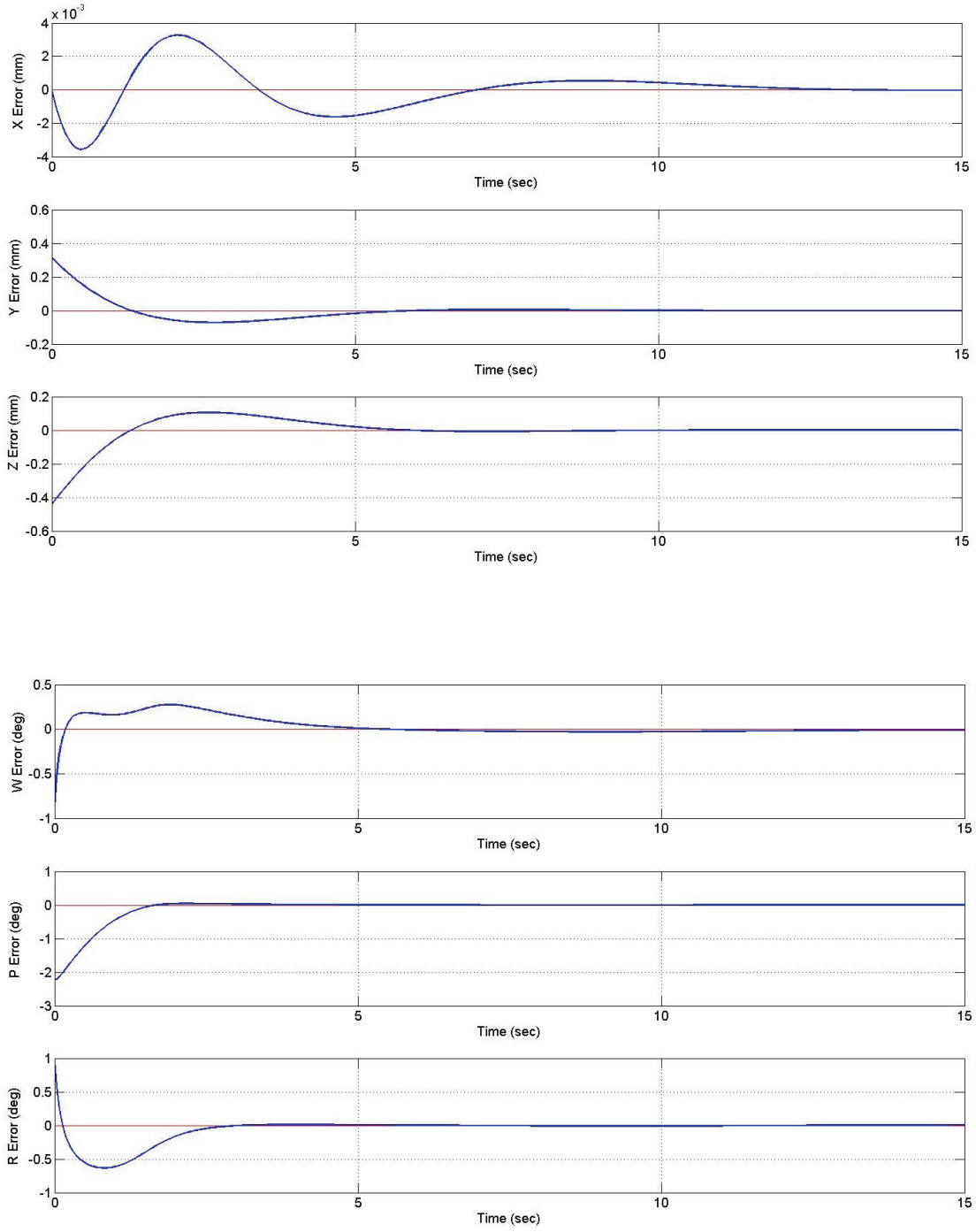


Figure 4.7: Position and orientation error

$$Uncertainty = \begin{bmatrix} 0.0500 & 0.0500 & 0.0500 & 0 & 0 & 0 \\ 0.0500 & 0.0500 & 0.0500 & 0 & 0 & 0 \\ 0.0500 & 0.0500 & 0.0500 & 0 & 0 & 0 \\ 0.0500 & 0.0500 & 0.0500 & 0.0500 & 0.0500 & 0.0500 \\ 0.0500 & 0.0500 & 0.0500 & 0.0500 & 0.0500 & 0.0500 \\ 0.0500 & 0.0500 & 0.0500 & 0.0500 & 0.0500 & 0.0500 \end{bmatrix} \tag{4.12b}$$

In this simulation, the robot is moved in Y-Z plane and the X is considered to be fixed. Figure 4.7 shows the error signal for the position and orientation. The Integral of Squared Error (ISE) for each axis is represented in Table 4.3.

As can be seen in Figure 4.7, although the uncertainty affect the joint angles of the robot, the controller can eliminate the generated error and the robot can reach the destination accurately.

The simulation results demonstrate that the total ISE has been improved 40%, which prove the feasibility of the proposed method for compensating the errors (uncertainties).

Table 4.3: ISE Analysis

	Integral of Squared Error (ISE)	
	Without PID Controller	With PID Controller
X	2.9460 e-5	2.403 e-5
Y	0.0498	0.0370
Z	0.1027	0.0726
W	0.4659	0.0203
P	2.8780	1.938
R	0.5253	0.393

4.4 Summary

In this chapter, Fanuc dynamic path modification option to apply dynamic corrections is introduced. The fundamental of the PID controller for PBVS is given. Finally, the simulation results are presented to verify the feasibility of the proposed strategy.

Chapter 5

Experimental Setup

Once the dynamic correction task between two poses is fully tested in the simulation, the proposed dynamic pose correction algorithm can be implemented using the C-Track on the real industrial robot.

This chapter covers the prerequisites and general requirements for implementing the proposed control algorithm. The manufactured tool for the robot as well as the design criteria are presented in the first part of this chapter. The developed software for implementation is introduced and general information, as well as the key features of this software are explained. In addition, the techniques used in this project for calculating the offsets are presented in the last section of this chapter.

5.1 Robot Tool

Industrial robots need appropriate tools to perform their tasks. Therefore, the first step is to design a tool that meets the requirements of the experiment. Generally speaking, industrial robots need to use different tools to perform different tasks. An automatic tool changer (ATC) is usually used to improve the production and tool carrying capacity of the robots.

ATC changes the tool of the robot very fast, which saves valuable time compared to when the tool is manually replaced [45].

In order to implement the proposed algorithm, the robot tool must be inserted into the hole with a high accuracy. According to the mentioned task, the tool of the robot should consists of a needle and a flat surface to attach reflector targets (to create end-effector model). In addition, the effect of the weight of the tool should be considered in the dynamics of the robot.

According to the mentioned specifications, the initial design of the tool is performed in CATIA. In order to avoid the effect of the load on joint 6, aluminium is selected as the material for the tool. Since at the beginning of this research, the robot was not equipped with the tool changer, the compatibility of the designed tool with both joint 6 flange and tool-changer is considered as one of the main design criteria.

According to the drawings of joint 6 flange and the tool changer, the tool base and the needle are designed. Figures 5.1 and 5.2 shows the drawings for the designed tool. The isometric view of the designed tool, needle and the assembled tool is also shown in Figure 5.3.

The tool is manufactured at Concordia University according to the drawings. Figure 5.4 shows the manufactured tool attached to the robot.

5.1.1 Creating Tool Frame at TCP

Once the tool is manufactured and attached to the robot, the next step is to define the TCP and to assign the tool frame at the TCP of the robot. The tool frame needs to be defined in the robot controller as the active tool frame and also in the VXelements as the origin-offset for the tool model. The procedures for creating the tool frame for both are explained in the following.

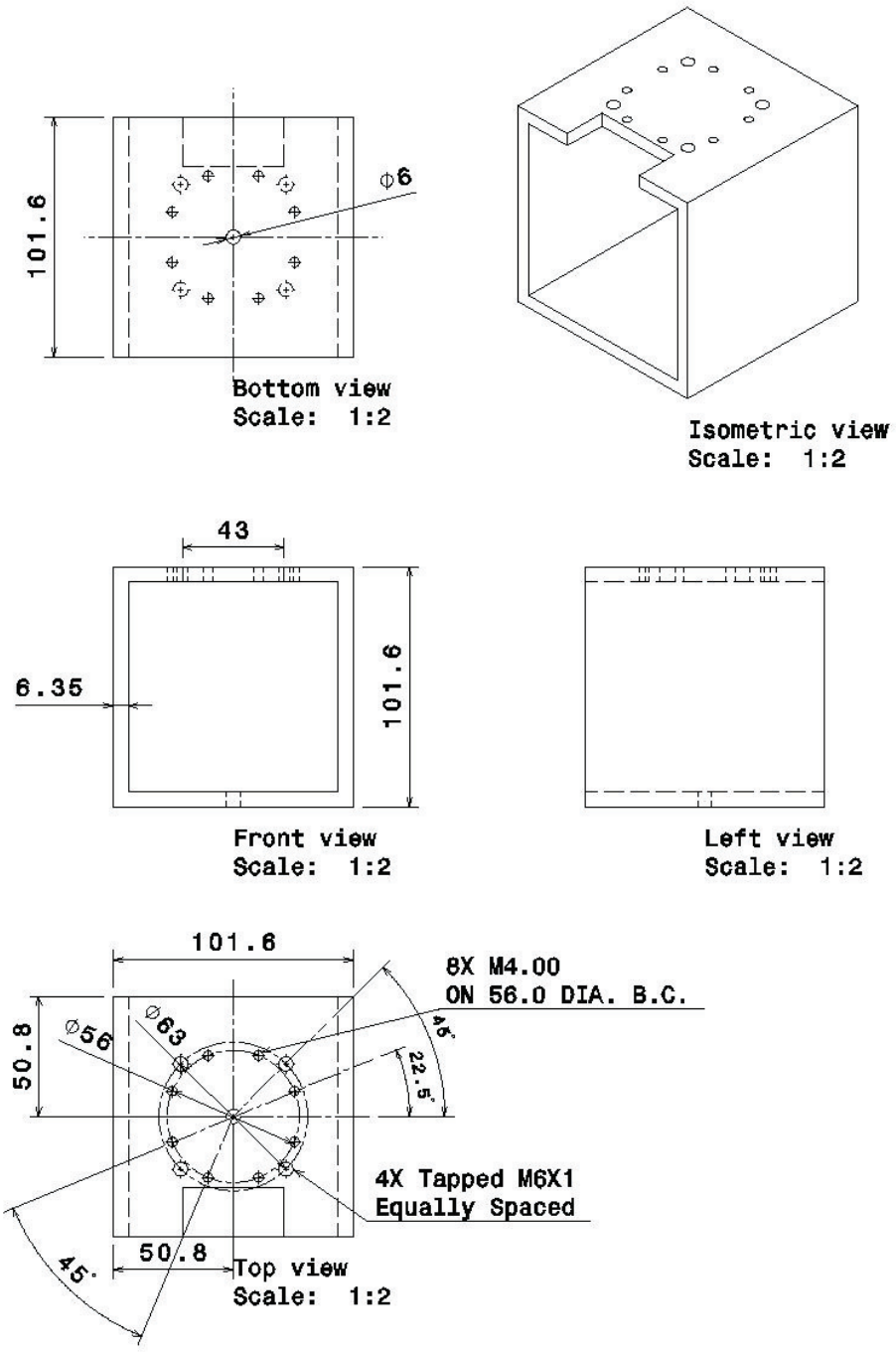


Figure 5.1: Drawing of the tool

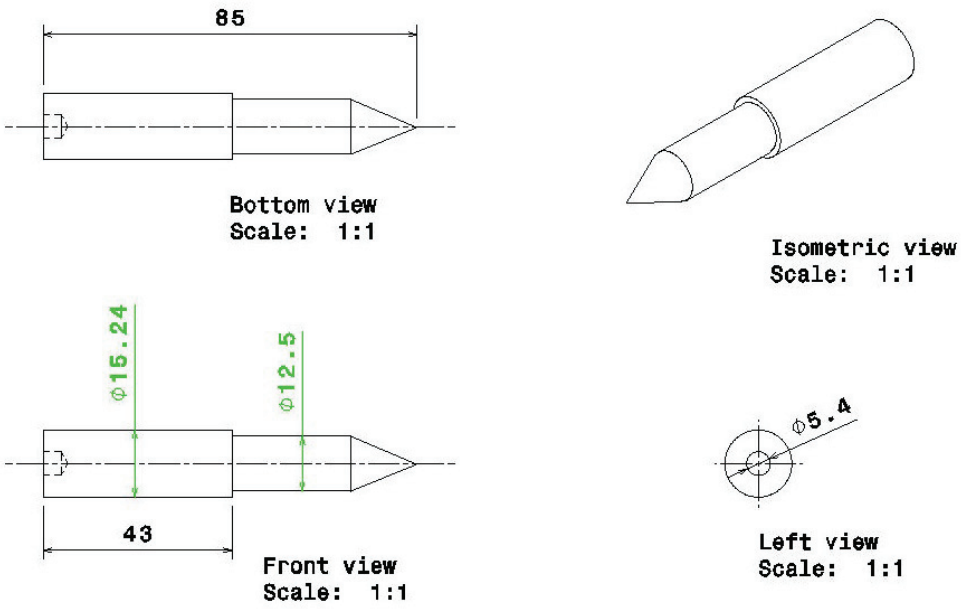


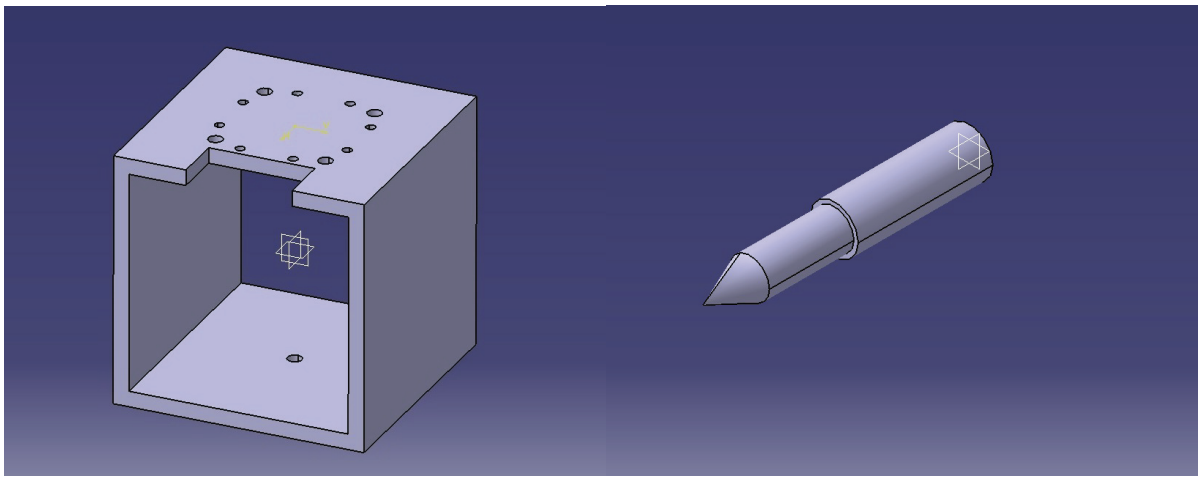
Figure 5.2: Drawing of the Needle

Tool Frame in Robot Controller

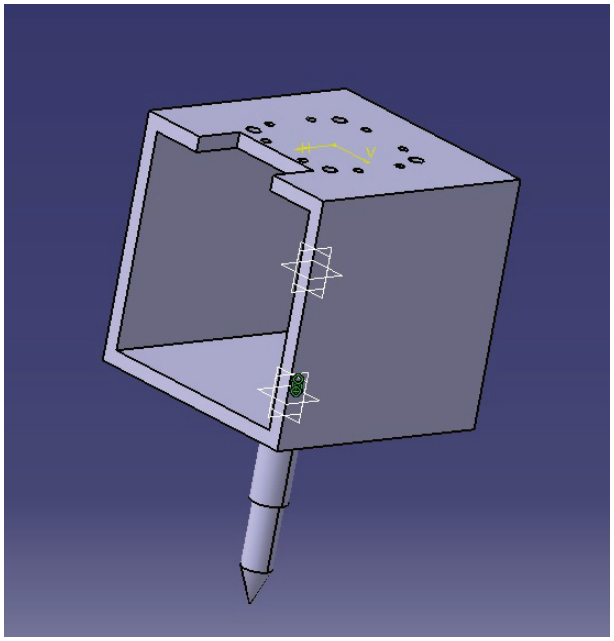
By default, the tool frame of the robot is at the center of the joint 6 flange as shown in Figure 5.5.

In order to find the exact location of the TCP in the robot controller, the three-point method is used [11]. In this method, the robot is jogged to the tip of a fixed tapper cone in three different configurations using the teach-pendant. The procedure is shown in Figure 5.6. After recording the coordinates of the three points by the teach pendant, the robot controller calculates the position of the needle tip (TCP) with respect to the default tool frame. The controller automatically creates a frame at TCP where the axes are parallel to the default tool frame. Figure 5.7 shows the new tool frame of the robot.

The position of the TCP with respect to the default tool frame will be used in the next step for creating the origin offset for the tool model in VXelements.

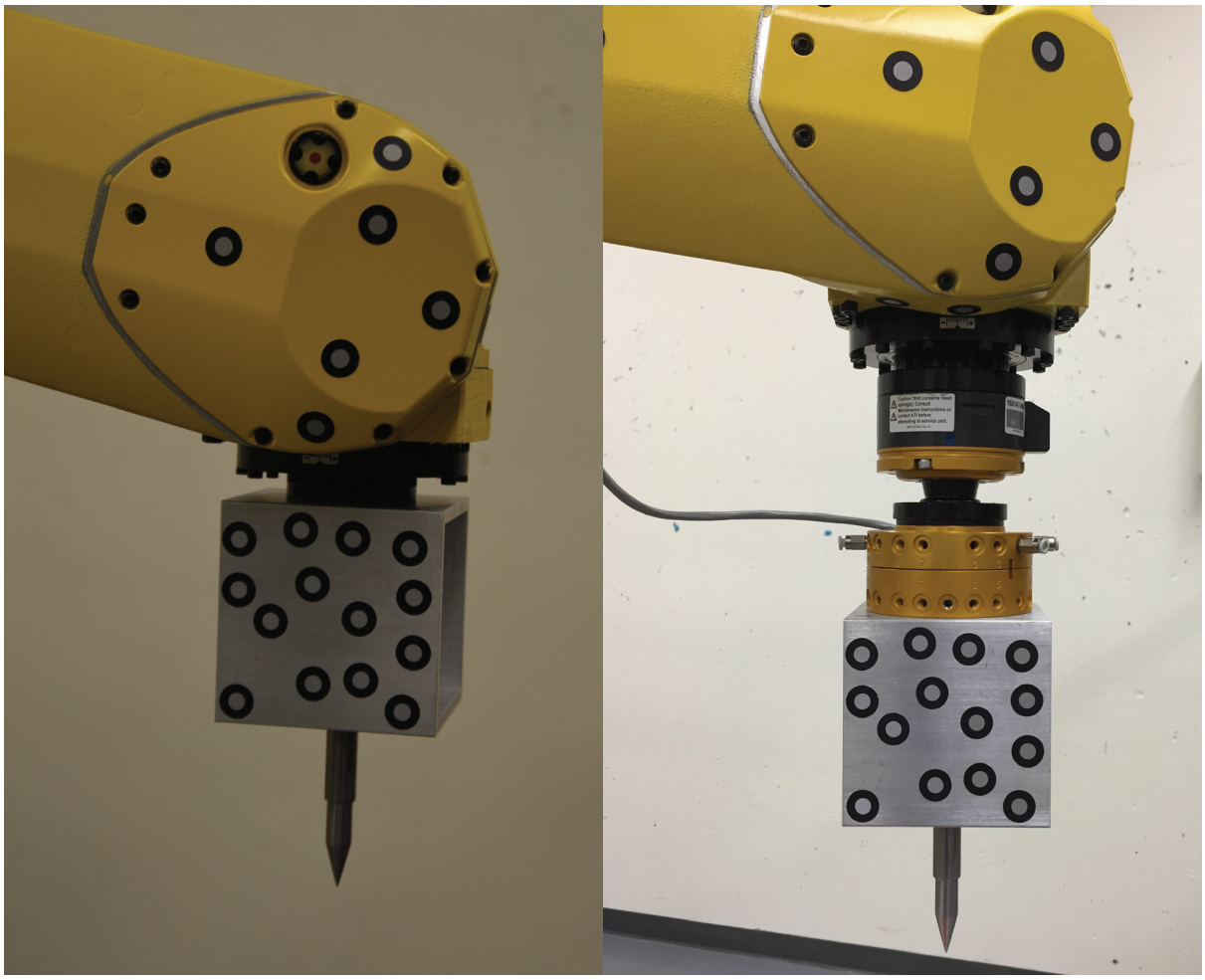


(a) (b)



(c)

Figure 5.3: (a) Tool Base compatible with Tool-Changer and Robot Flange, (b) Needle, and (c) Assembled Tool.



(a) (b)

Figure 5.4: (a) Tool attached to the robot flange, (b) Tool attached to the tool-changer

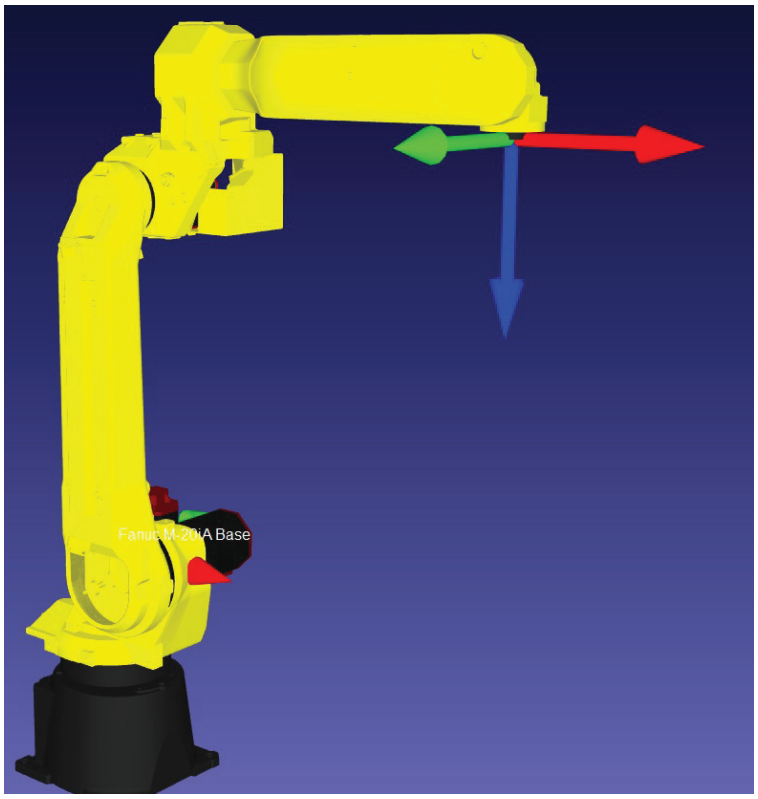


Figure 5.5: The default tool frame of the robot

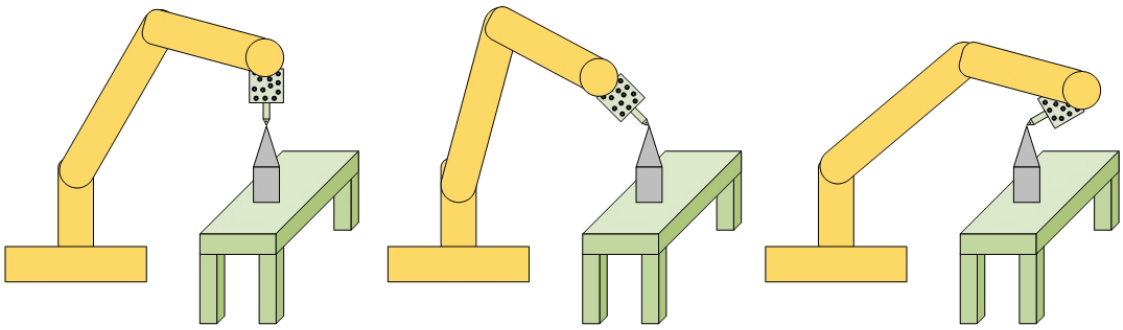


Figure 5.6: Finding the TCP by using three-point method

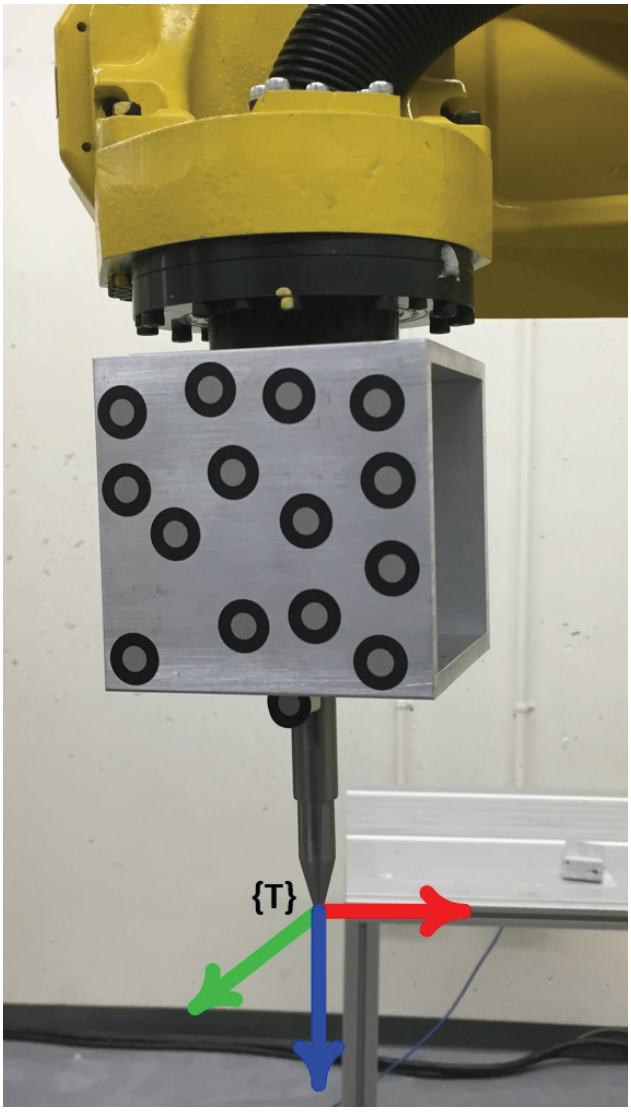
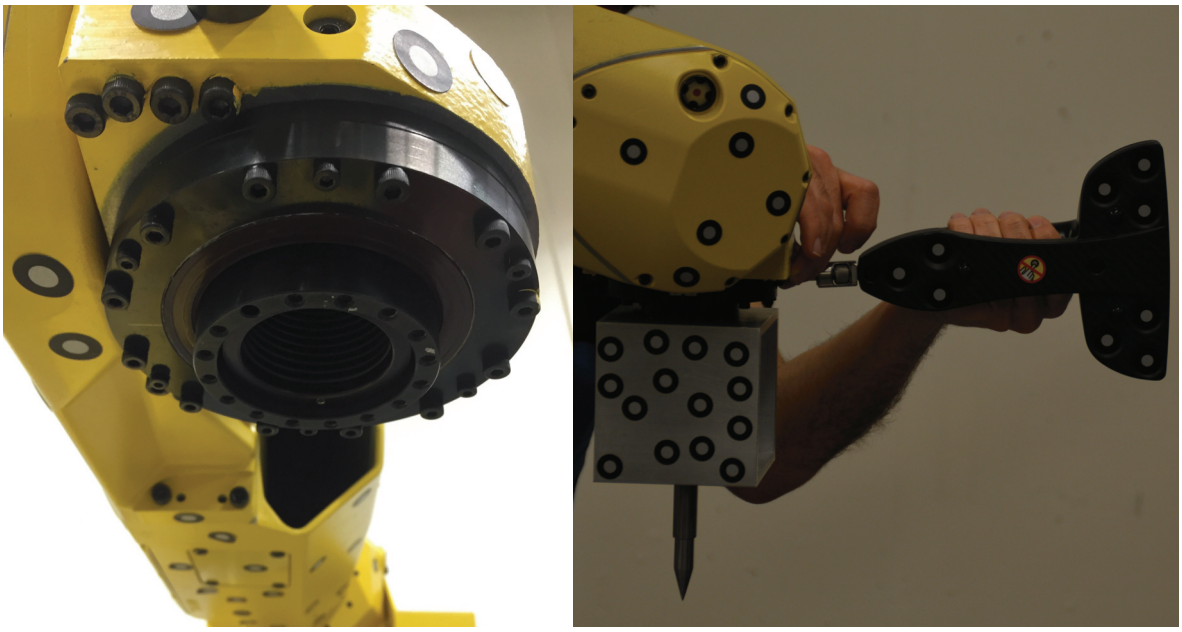


Figure 5.7: New tool frame of the robot at TCP



(a) (b)

Figure 5.8: (a) Robot flange and (b) probing flange to find original tool frame

Tool Frame in VXelements

After creating the new tool frame in the robot controller, it is necessary to create the same frame as the origin-offset of the tool model in VXelements software. A frame in VXelements can be created using one point as the origin and two axes. The technique used in this part of the research includes creating the default tool frame of the robot using the HandyProbe and then, using the values calculated by the robot to offset the created frame into the TCP.

In order to create the default tool frame of the robot in VXelements, HandyProbe is used to probe the measurable features of joint 6. As shown in Figure 5.8, the plane and the smooth circular surface of the flange are probed. The center of the circle is found as the origin of the default tool frame and the normal axis of the probed circle (passing through the origin) assigned to the Z direction. The second axis is obtained using the origin point and probing the zero sign of joint six.

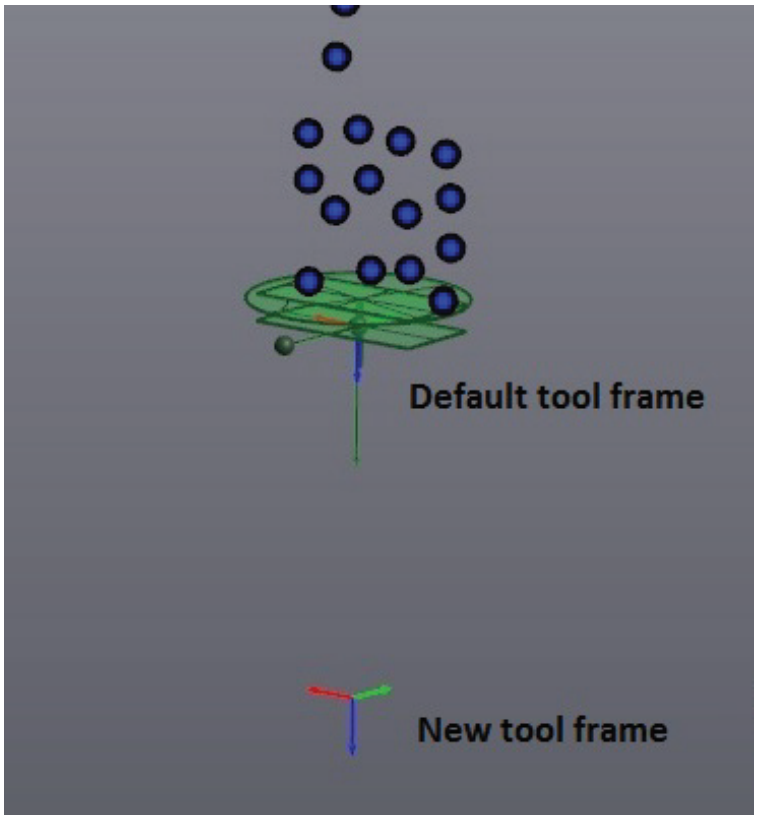


Figure 5.9: Default tool frame and new tool frame in VXelements

By creating the origin and two axes, the default tool frame of the robot is created. Using the coordinates of the TCP with respect to the original tool frame from the previous step, the new tool frame can be obtained in VXelements. The probed entities and the created frames are shown in Figure 5.9.

In Figure 5.9 the green circle and rectangle are the probed circular surface and the plane surfaces on the flange respectively. In addition, the default tool-frame and the new tool-frame are shown.

After finding the new tool frame at TCP, this frame can be attached to the tool model as origin offset. As a result, the C-Track will detect the pose of the TCP as long as the model is in the C-Track's field of view (FOV).

5.2 Developed Software

In order to process the measurements, calculate the errors, implement the control strategy and apply real-time corrections, an interface is designed in Microsoft Visual Basic. The developed software consists of two major parts. The first part is the robot communication, which is designed to access the robot controller variables and to apply dynamic corrections to the robot. The second part is the C-Track communication, which receives the C-Track measurements according to the created models. These parts are designed separately to check the performance and then are combined to perform the PBVS task as explained in Figure 4.1.

5.2.1 Robot Communication Interface

The communication with the robot controller is realized by Fanuc PC Developer's Kit (PCDK) through the Ethernet Cable and TCP/IP protocol. In the designed interface, the user can access the robot controller variables and the current pose of the robot with respect to the base frame or user frame. Using this interface, the user can also apply manual offsets in X,Y,Z,W,P,R directions using the DPM variables. The designed interface for simple applications such as reading the pose and applying manual offsets to the robot are shown in Figure 5.10.

5.2.2 C-Track Communication Interface

VXelements API is used to communicate with the Creaform instruments. The designed interface for the C-Track allows the user to connect to the camera, to create referential targets and models (for tracking), to load the created reference targets and models, to add the relation between the created models and to check the status of the C-Track whether it

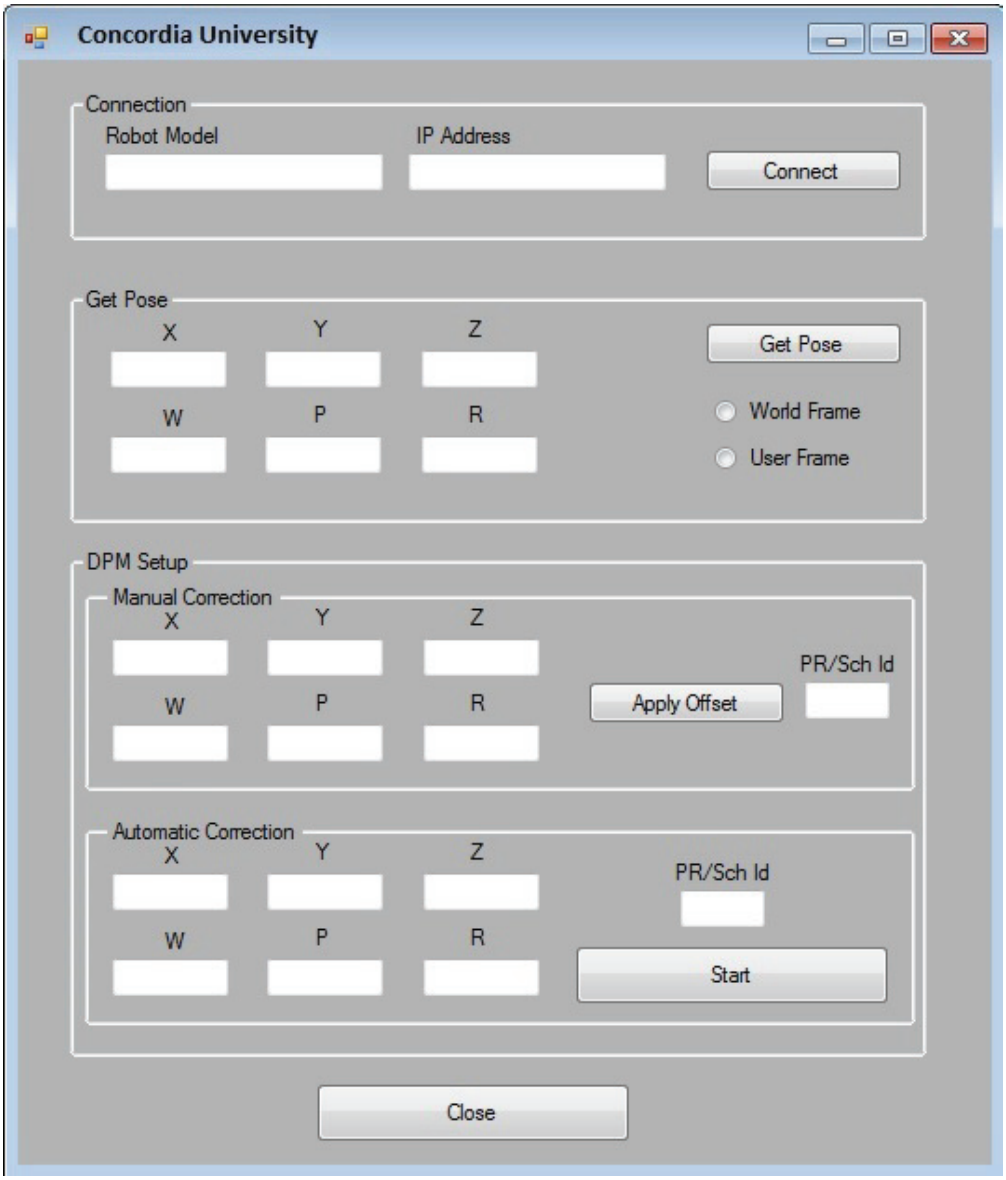


Figure 5.10: Robot Controller Interface

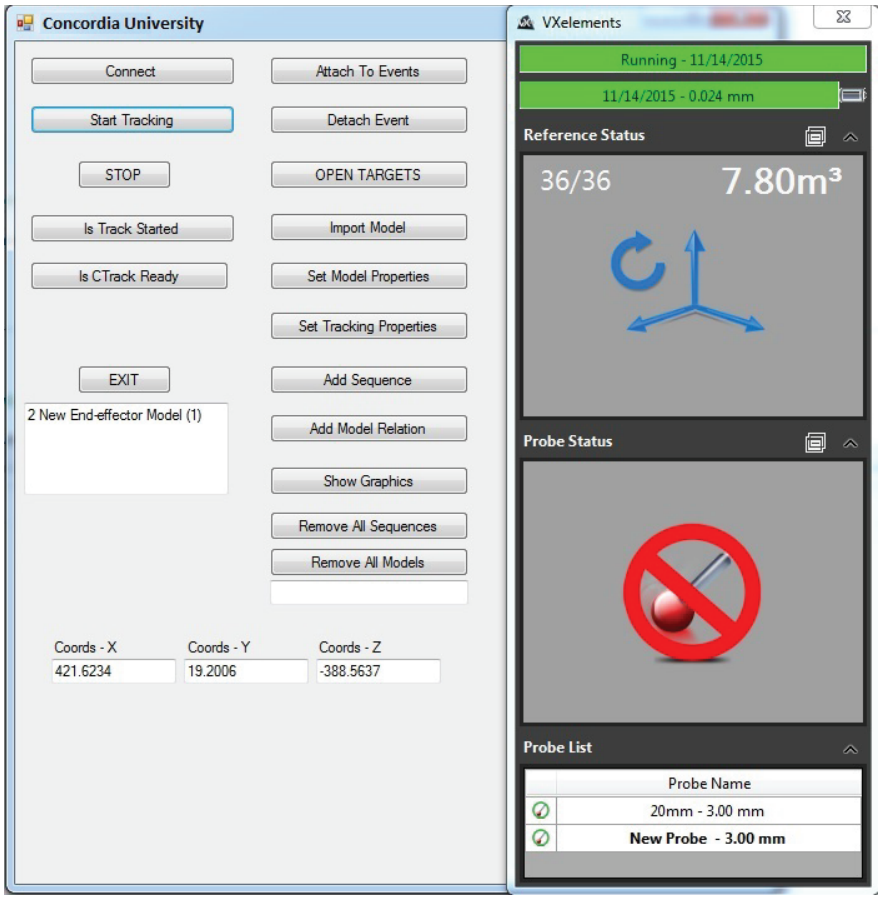


Figure 5.11: C-Track Interface

is working or not. The tracking information can be shown graphically using *show graphics* button and the real-time position and orientation of the models can be displayed. Figure 5.11 shows the designed interface for the C-Track.

5.2.3 Final Interface

In order to use the C-Track data as the feedback measurement for the control system, both interfaces in Figures 5.10 and 5.11 have to be combined in a single interface. The final version of the designed interface in which both robot and camera work together is shown in Figure 5.12.

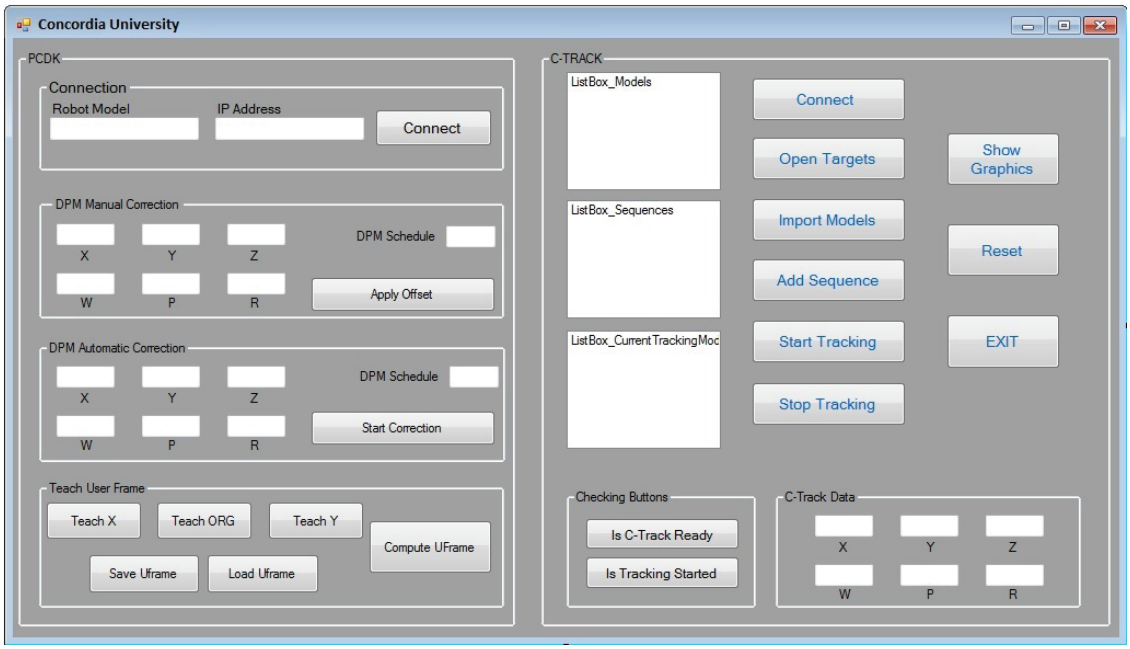


Figure 5.12: Designed Dynamic Pose Control Software Interface

The user can connect to both robot controller and C-Track simultaneously. The reference targets and models can be either created or loaded to the software. The reference status in the software shows the number of the reflectors which can be seen by the C-Track. The tracking properties such as frequency of the C-Track can be set using the software as well as the model properties and model relations. Once the "start tracking" button is pressed by the user, the real-time position and orientation of the models will be displayed according to the frequency of the C-Track.

Figure 5.13, shows the physical connection between the C-Track, the computer and the robot controller. As can be seen in this figure, the C-Track provides the current pose of the robot. The software receives the provided measurements and compares the measured pose with the desired one. The difference is applied to the robot controller to correct the position and orientation of the end-effector as shown in Figure 5.14

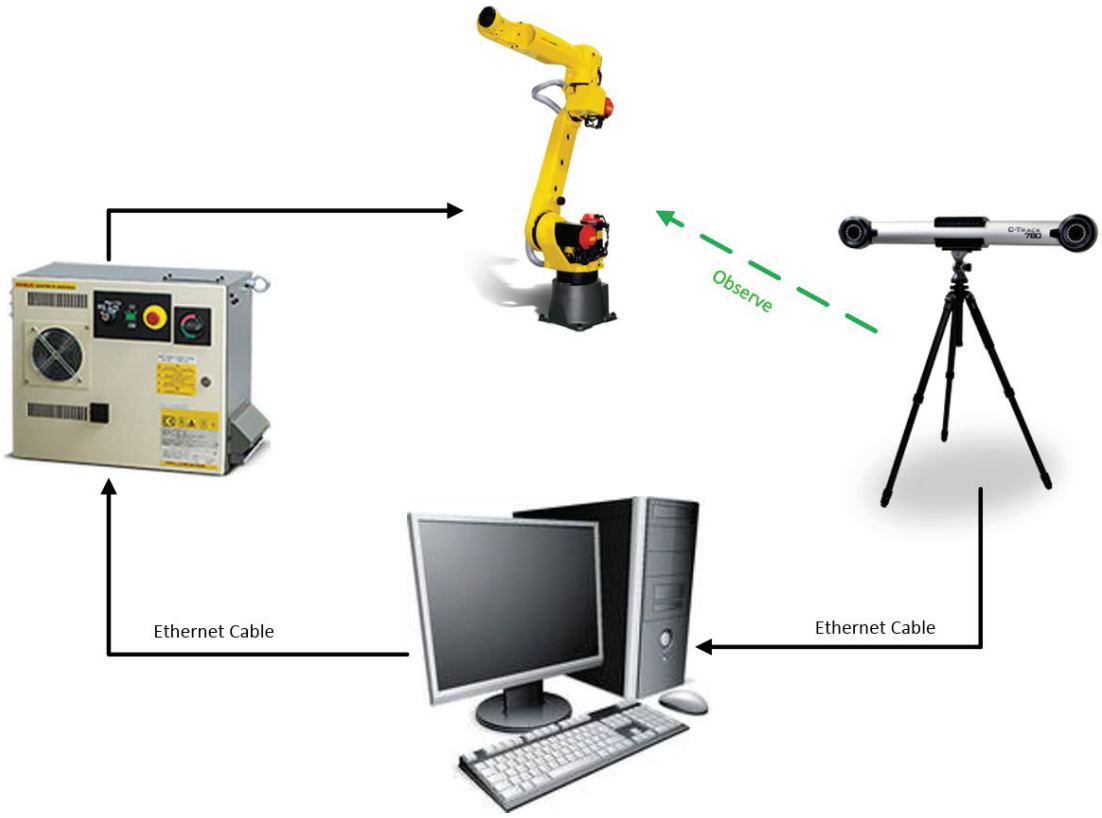


Figure 5.13: Schematic of the connections

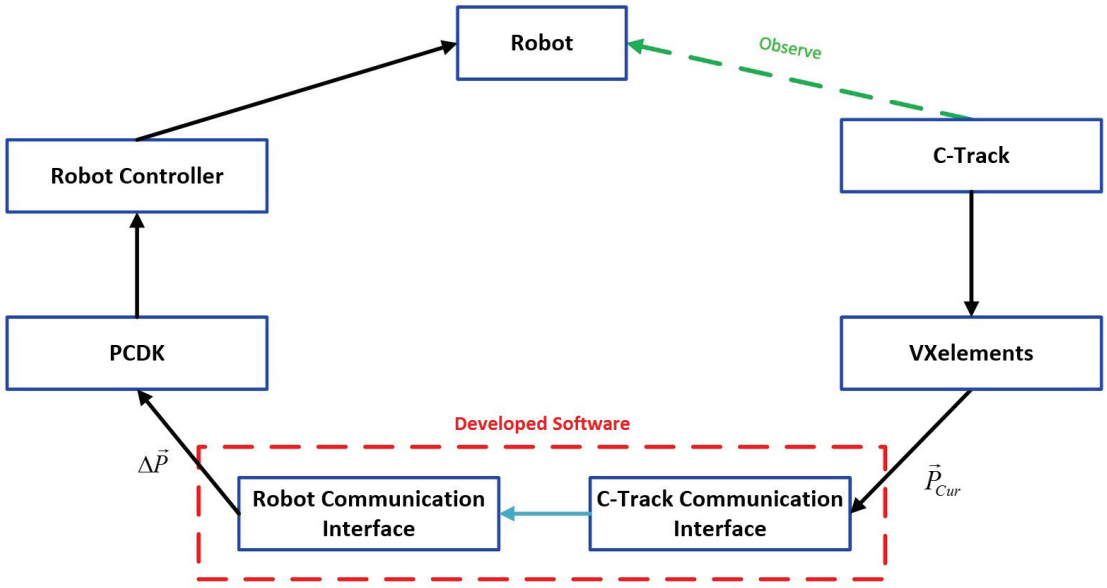


Figure 5.14: Schematic of the relation between the interfaces

5.2.4 Teach User-Frame

The *Teach User-Frame* section in the designed software interface is used to create the same user-frame of the robot controller in the software. This frame is used as a reference frame to calculate the corrections. Moreover, the DPM option uses the user-frame as a reference for applying offsets as explained in Section 4.1. Therefore, it is necessary to create the same user-frame in the software for calculating and applying the corrections.

In order to create the same user-frame in the software, the robot can be jogged in the user frame coordinates of the robot manualy by the teach pendant. Using the C-Track and created tool model, the position and orientation of the tool model can be recorded dynamically. Figure 5.15 shows the strategy of creating the user frame.

According to Figure 5.15, the robot is jogged to an arbitrary position in the C-Track's FOV and the coordinates of this point are recorded using the *Teach X* button (point A). After recording the first point, the robot is jogged in the $-x$ direction with respect to the robot active user frame using the teach pendant. This point is recorded as the origin of the user-frame (point B). Finally, the robot is jogged in +y direction and the last position is recorded (point C).

Since the measurements taken by the C-Track are noisy and considering only one measurement for the pose is not reliable, as explained in Section 3.4, the RMS value of 5 to 10 measurements is used for each point as a way to find the accurate measurement of the tool model.

Considering the 3 recorded points, we can create 2 intersecting axis $\overrightarrow{BA}$ and $\overrightarrow{BC}$ that $\overrightarrow{BA}$ is in the X direction of robot user frame and $\overrightarrow{BC}$ is parallel to the Y direction of the robot user frame.

The perpendicularity of the created axes is verified by importing the coordinates of points A, B, C into VXelements. The angle between $\overrightarrow{BA}$ and $\overrightarrow{BC}$ can be measured by the software.

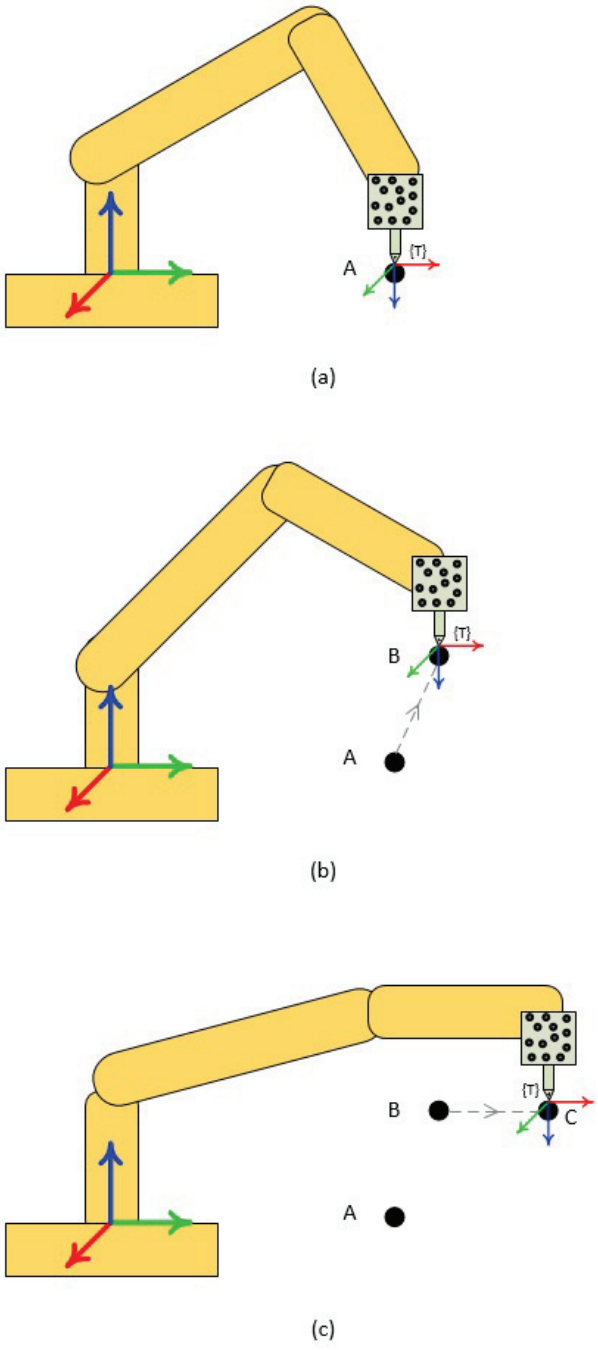


Figure 5.15: Three steps to define the user frame (a) record the first point A (b) Jog the robot in -x direction of the user-frame and record point B (c) Jog the robot in +y direction of the user-frame and record point C

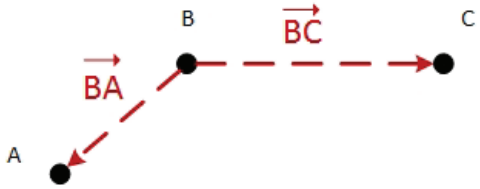


Figure 5.16: Creating $\overrightarrow{BA}$ and $\overrightarrow{BC}$ using the robot and C-Track

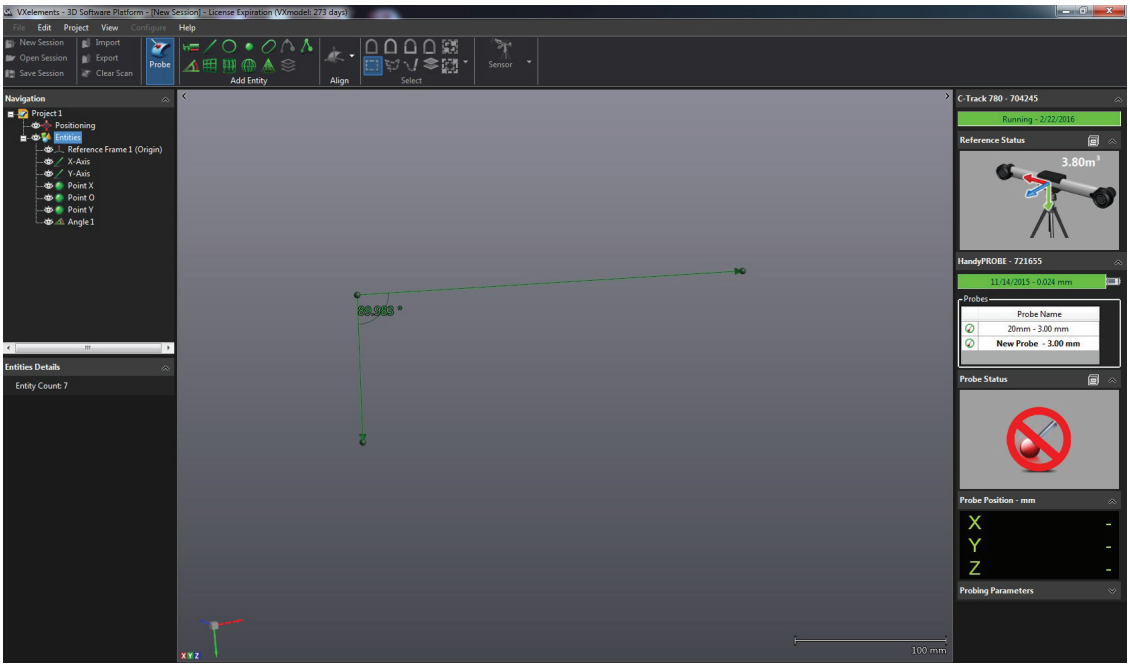


Figure 5.17: Verification of the perpendicularity of the created X and Y axes of the user-frame using VXelements software

Figure 5.17 shows that the angle is 89.983deg. The Z axis of the user-frame can be calculated by the cross product of X and Y.

As mentioned earlier, this user frame is used as the reference frame in the software for applying dynamic corrections. The corrections are applied with respect to the robot active user frame. If the calculated offsets and the applied ones are not represented with respect to the same reference frame, the coupling effect will affect the behaviour of the robot and the control strategy.

In order to clarify the coupling effect, assume the total error calculated by the software

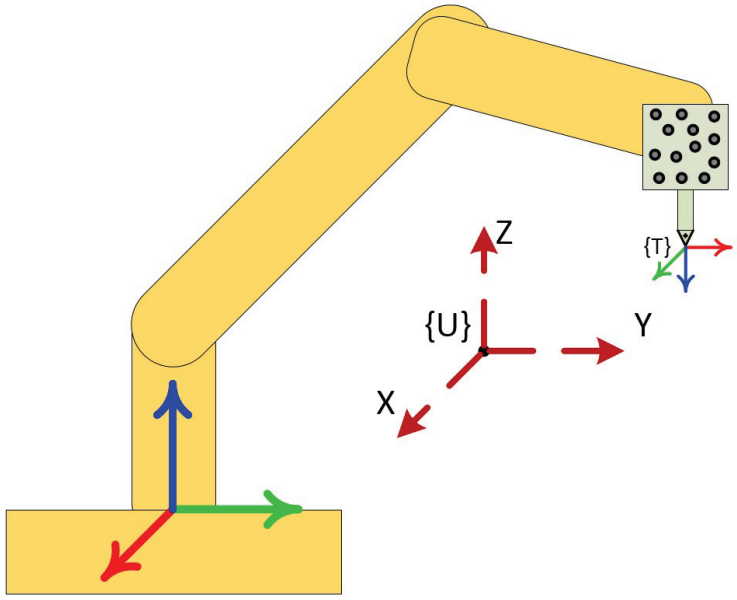


Figure 5.18: Final user-frame created in the software

with respect to the user frame is 2 mm in X direction. The correction value, calculated by the software, is given to the robot controller and is applied with respect to the active user frame defined in the robot controller. If the axes of these two frames are not parallel, the 2 mm offset in X will also affect the values of Y and Z. Consequently the robot cannot be guided directly to the desired pose.

Since the software is calculating the difference between the desired pose and current pose of the robot and the robot controller is applying the difference to the robot, these two frames need to have parallel axes only. Therefore, the origin of these frames should not necessarily be the same.

The software is also capable of saving the taught user frame as the transformation matrix between the user frame $\{U\}$ and the C-Track frame $\{S\}$ using the *Save Uframe* button (${}^{S}_{U}T$). This frame can be loaded and used in the next experiments if the C-Track position is not changed. This technique can save the time for teaching and calculating the frame after each experiment. The mathematical procedures for calculating the user frame using the recorded

points are as follows:

$$\vec{X} = \vec{BA} = \vec{A} - \vec{B} = [X_x, Y_x, Z_x] \tag{5.1a}$$

$$\vec{Y} = \vec{BC} = \vec{C} - \vec{B} = [X_y, Y_y, Z_y] \tag{5.1b}$$

$$\vec{Z} = \vec{X} \times \vec{Y} = [X_z, Y_z, Z_z] \tag{5.1c}$$

Using the cross product of X and Y, the Z axis elements can be computed:

$$X_z = Y_x Z_y - Z_x Y_y \tag{5.2a}$$

$$Y_z = Z_x X_y - X_x Z_y \tag{5.2b}$$

$$Z_z = X_x Y_y - Y_x X_y \tag{5.2c}$$

The unit vector for each axis can be found by dividing by the norm of each vector.

$$\hat{X} = \frac{\vec{X}}{|\vec{X}|} = \begin{bmatrix} \hat{X}_x & \hat{Y}_x & \hat{Z}_x \end{bmatrix} \tag{5.3a}$$

$$\hat{Y} = \frac{\vec{Y}}{|\vec{Y}|} = \begin{bmatrix} \hat{X}_y & \hat{Y}_y & \hat{Z}_y \end{bmatrix} \tag{5.3b}$$

$$\hat{Z} = \frac{\vec{Z}}{|\vec{Z}|} = \begin{bmatrix} \hat{X}_z & \hat{Y}_z & \hat{Z}_z \end{bmatrix} \tag{5.3c}$$

Using the unit vectors and the coordinate of the frame origin, the user frame transformation

matrix relative to the C-Track frame can be obtained as follows:

$$ {}^{S}_{U}H = \begin{bmatrix} \hat{X}_x & \hat{X}_y & \hat{X}_z & X_B \\ \hat{Y}_x & \hat{Y}_y & \hat{Y}_z & Y_B \\ \hat{Z}_x & \hat{Z}_y & \hat{Z}_z & Z_B \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4} $$

where X_B, Y_B and Z_B are the coordinates of the origin of the user frame (point B) with respect to the sensor frame obtained from step (b) in Figure 5.15. This transformation matrix is used to transform the corrections from C-Track frame $\{S\}$ into user frame $\{U\}$.

5.3 Offset Calculations

Considering the position error vector as the difference of the desired position and current position, the transformation matrix between two frames can be used to transform the error vector into any known reference frame. For instance, in (5.5b), the error represented in frame S (sensor frame) is transformed into frame U (user frame) by multiplying the transformation matrix, ${}^{S}_{U}H^{-1}$.

$$ {}^{S}\Delta P = {}^{S}P_{Des} - {}^{S}P_{Cur} \tag{5.5a} $$

$$ {}^{U}\Delta P = {}^{S}_{U}H^{-1} \times {}^{S}\Delta P \tag{5.5b} $$

These equations are used in the developed software to calculate the position correction with respect to the user frame. The calculated corrections can be applied to the robot directly by the DPM option.

In order to correct the orientation of the robot, a strategy based on the equivalent angle axis method is developed and implemented. The equivalent angle axis method uses a vector

and an angle to define the orientation between two frames. The vector is called equivalent axis and the angle is the rotation angle along the equivalent axis based on the right hand rule. The equivalent vector and the angle can be calculated from the rotation matrix between two frames [15]. Consider the rotation matrix between Frame B and A as ${}^{A}_{B}R$ represented by:

$$ {}^{A}_{B}R = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{5.6} $$

The equivalent rotation angle, θ, between the two frames can be calculated from the following equation:

$$ \theta = \arccos\left(\frac{a_{11} + a_{22} + a_{33} - 1}{2}\right) \tag{5.7} $$

and the equivalent axis $\hat{K}$ is obtained by:

$$ \hat{K} = \frac{1}{2\sin\theta} \begin{bmatrix} a_{32} - a_{23} \\ a_{13} - a_{31} \\ a_{21} - a_{12} \end{bmatrix} = \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix} \tag{5.8} $$

On the other hand, according to the equivalent angle and equivalent axis, the rotation matrix between two frames can be calculated by the following equation:

$$ R_K(\theta) = \begin{bmatrix} k_x k_x v\theta & k_x k_y v\theta - k_z s\theta & k_x k_z v\theta + k_y s\theta \\ k_x k_y v\theta + k_z s\theta & k_y k_y v\theta + c\theta & k_y k_z v\theta - k_x s\theta \\ k_x k_z v\theta - k_y s\theta & k_y k_z v\theta + k_x s\theta & k_z k_z v\theta + c\theta \end{bmatrix} \tag{5.9} $$

where k_x, k_y and k_z are the elements of the equivalent axis vector $\hat{K}$, $s\theta = \sin\theta$, $c\theta = \cos\theta$

and $v\theta = 1 - \cos\theta$.

Once the rotation matrix between two frames is known, the Euler angles can be extracted according to the method explained in Section 3.1 where we calculate the orientation from the rotation matrix.

In this research, this algorithm is used to find the orientation correction between the current pose of the robot and the desired one. Through the software, the equivalent angle and axis between the desired pose and the current pose of the robot is calculated with respect to the sensor frame. Since the corrections are applied with respect to the user frame, all the calculations are transformed into user frame. Therefore, the equivalent axis vector is transformed into user frame by using the homogeneous transformation matrix. After finding the new equivalent axis, the new rotation matrix can be obtained from (5.9). By comparing the rotation matrix with (3.33), the W-P-R values are extracted based on the explained algorithm. The calculated values can be applied to the robot using the DPM option. The calculation procedure for orientation corrections is presented in details in Section 6.2.

5.4 Software General Setup

The first step for each experiment is to set-up a stable environment so that the measurements from the C-Track can be stable and reliable. This set-up consists of dynamic referential targets, user frame and accurate tool model. The targets used for dynamic reference are recommended to be scattered in the space. This results in taking more accurate and less noisy measurements from the C-Track.

In this project, the targets attached to the aluminium bar and extensions are used for dynamic reference. Figure 5.19 shows the dynamic reference including 36 reflector targets. The model of the tool frame is created according to the targets attached to the tool base of the robot (Figure 5.7) and the TCP is at the tip of the tool needle as explained in Section

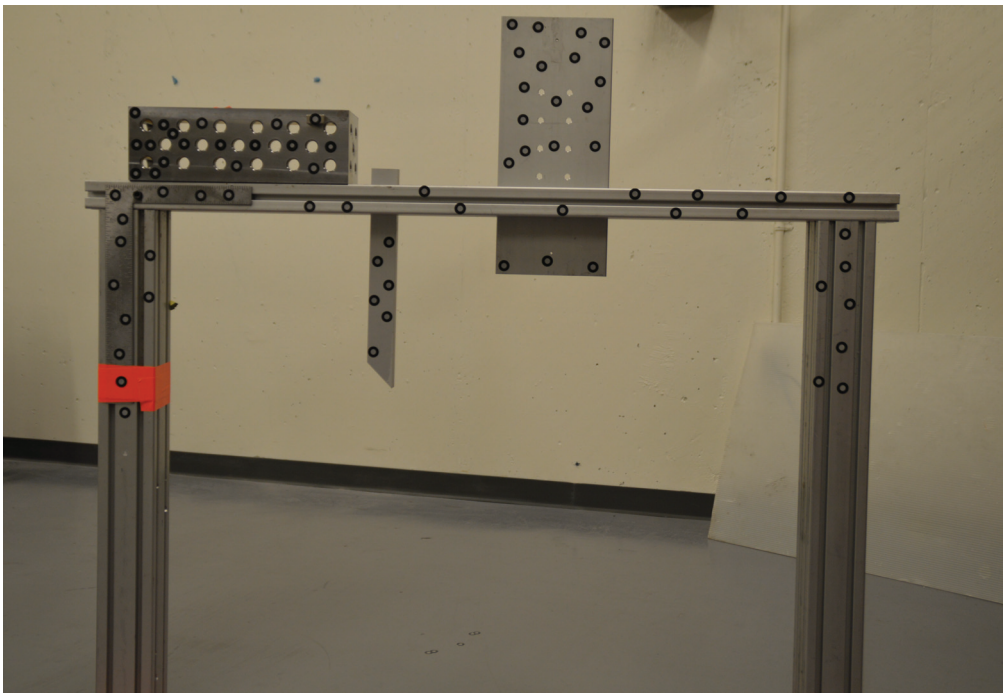


Figure 5.19: Dynamic Reference consists of the targets attached to the aluminium bar

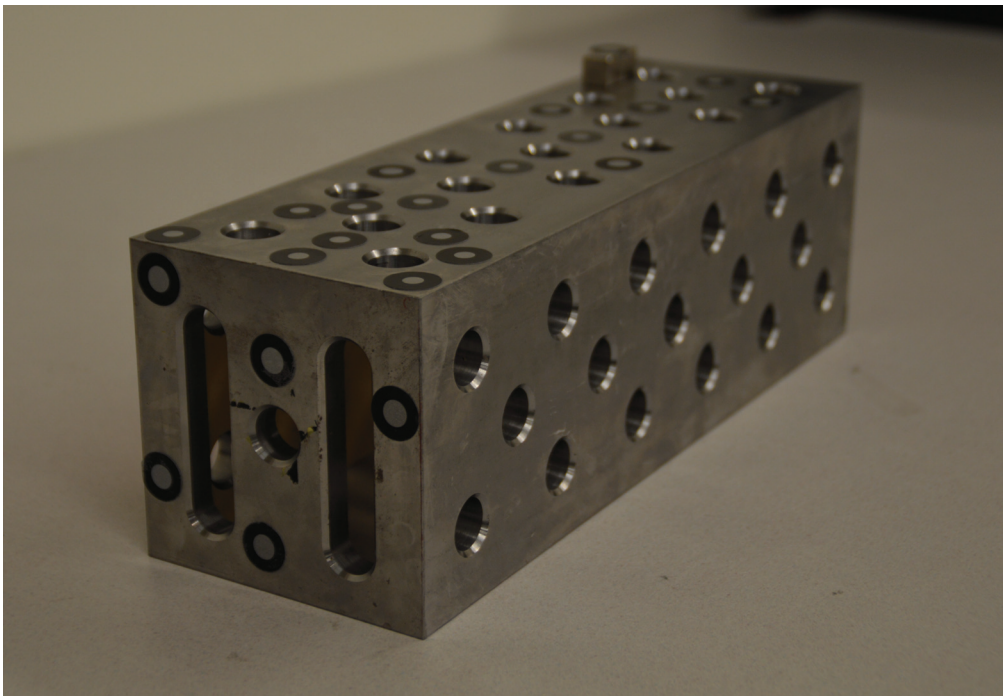


Figure 5.20: Standard cube used in this research

5.1.1. The user frame can also be identified as illustrated in Section 5.2.4.

The cube used in this project is a standard cube provided by ÉTS. The holes on the cube are equally spaced on each side and the diameter of each hole is 0.620 inches (15.748 mm). As explained in Section 5.1, the diameter of the needle is designed to be $15.24\,mm$. The cube is shown in Figure 5.20.

5.5 Summary

In this chapter, the preliminary setup and installations for the robot and experiments are explained. The developed software and the designed communication interfaces are introduced. The strategy for calculating the position and orientation offsets are also presented in this chapter.

Chapter 6

Experimental Results

Dynamic pose correction of the robot end-effector at a specific point can be used in different applications such as spot welding, drilling and riveting.

In this chapter, the proposed dynamic pose correction algorithm for enhancing the accuracy of Fanuc M20-iA robot is tested. The first experiment is to guide the robot end-effector to insert a series of holes on a fixed cube by controlling only the position of the tool (X , Y, Z). The analysis of the experiment results are given. Furthermore, the limitations of this experiment are explained. In order to eliminate the limitations of the first experiment, the second experiment is designed and tested. The analysis of the results for this experiment is presented. Finally, the performance of the proposed control strategy for the pose correction is assessed and the results are presented.

6.1 Experiment 1: Dynamic Position Correction to insert the holes on a fixed cube

This experiment is devoted to evaluate the performance of the robot when correcting only the position of the end-effector. Therefore the orientation of the tool remains fix during this experiment.

The main goal is to develop the dynamic position correction control algorithm to guide the robot end-effector to insert a series of holes on different sides of the cube with the precision of $\pm 0.05\,mm$ for each axes. In this experiment the accurate coordinates of each hole center are first being taught by using the HandyProbe and C-Track as shown in Figure 6.1.

By probing at least four points for each hole and creating the circle, the coordinates of the hole center is computed by the VXelements software automatically. The calculated coordinates for each hole center is loaded to the robot controller. The control structure for this experiment is shown as a flowchart in Figure 6.2. Also, a part of the TP program is shown in Figure 6.3. The configuration of the aluminium bar, cube and the robot for this experiment is shown in Figure 6.4. As can be seen, the cube is attached and fixed to the platform. Since the coordinates of each hole are obtained by the HandyProbe, the measurements are very sensitive to the vibrations. Once the centers of the probed circles are calculated, the cube should not be moved. Otherwise, the measured data is not valid and cannot be used.

As shown in Figure 6.3, the TP program consists of different commands including *Track*, *Tool_Offset* and *DO[i]=ON*. The robot will start the program from *P[2]* which is an arbitrary position in the workspace. The *Track DPM[i]* illustrates all the specifications regarding the DPM option. The DPM mode, maximum offset limit, rate of correction and synchronization

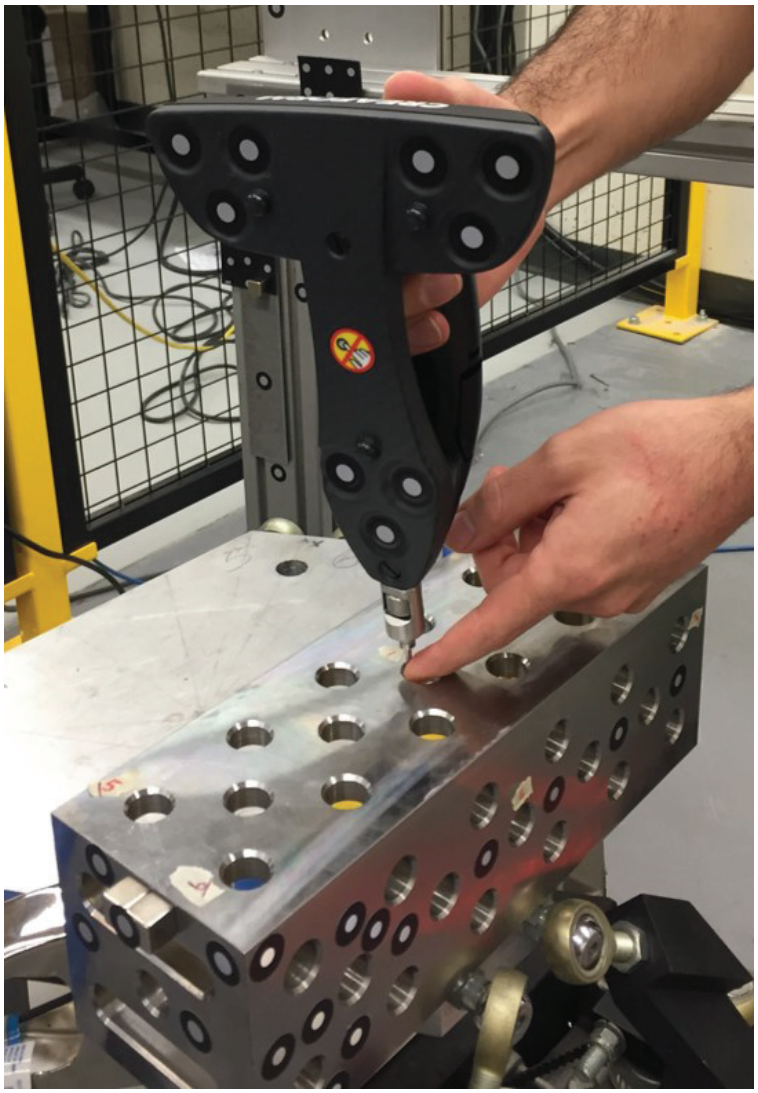


Figure 6.1: Measuring the accurate location of the Hole Center using the HandyProbe

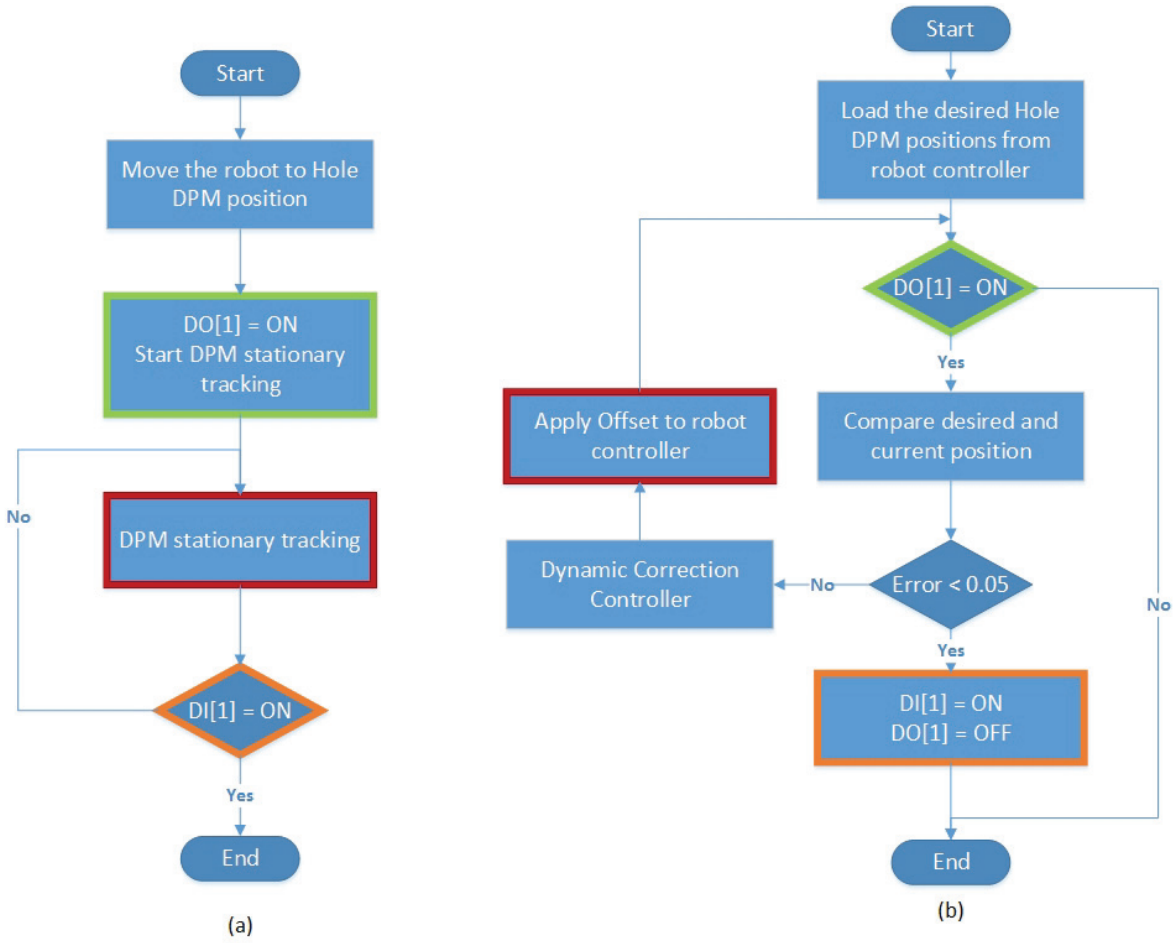


Figure 6.2: Flowchart of the first experiment algorithm (a) Teach-Pendant program (b) Developed pose control software

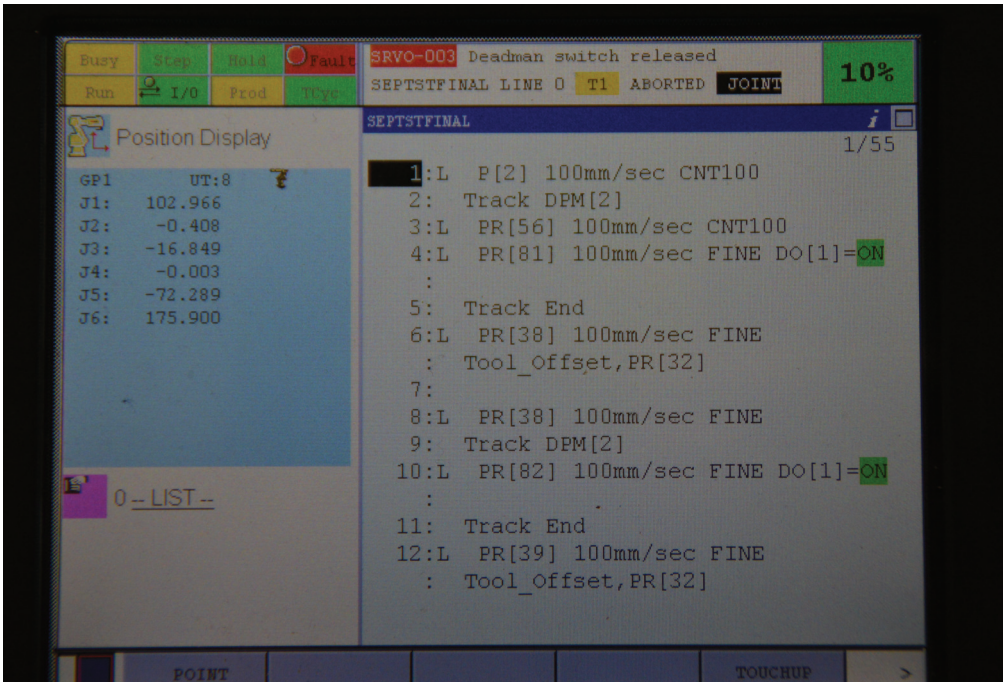


Figure 6.3: Teach-Pendant program for the first experiment

signal for stationary tracking are the examples of these specifications which are automatically set through the developed software.

This experiment consists of three steps. The first step is to go to the tracking point. The second step is to measure the position of the robot dynamically and guide the robot to the desired position by applying real-time corrections. This step is repeated until the error on each axes satisfies the required precision ($\pm 0.05\,mm$). The third step is to insert the hole by *Tool_Offset* command. Using this command, the robot is moved in the active tool frame coordinates of the robot according to the values given in PR[32]. In this experiment, this command consists of only movement in Z direction in order to insert the hole.

The DPM option used in this experiment is the Modal DPM in stationary mode. Therefore the robot will stop at the position register specified with the FINE term in the TP program (PR[81] and PR[82] in Figure 6.3). These positions are the taught positions from the HandyProbe.

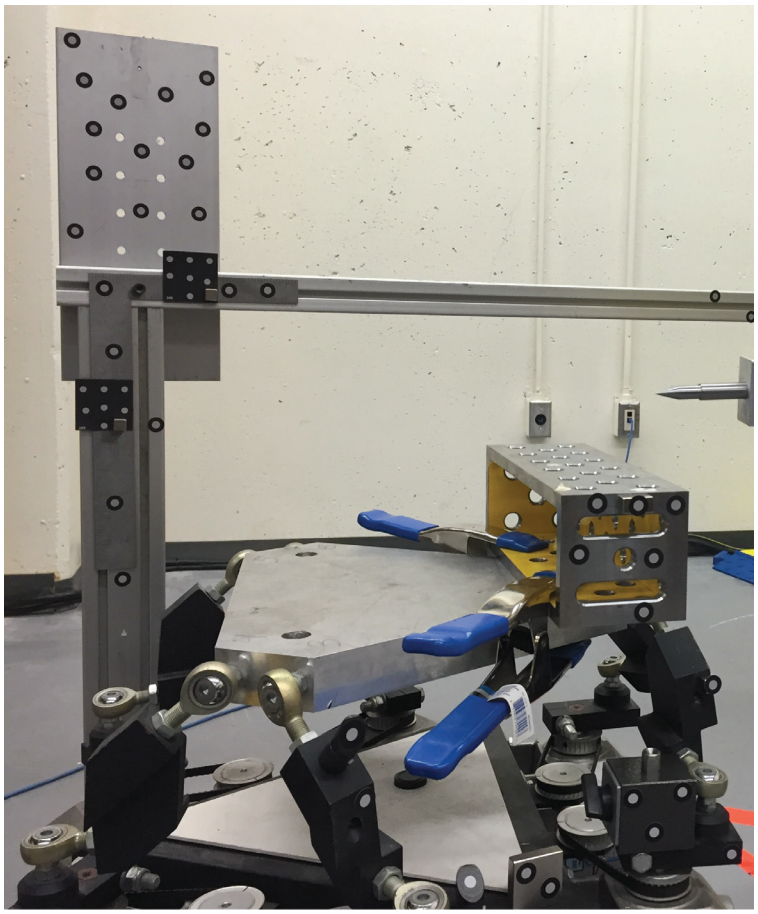


Figure 6.4: Setup for experiment 1

When the robot reaches to the PR[81], the Digital Output 1 switches to ON and the software uses this flag to start taking measurements by the C-Track. Since the stationary tracking is activated, the robot stops and waits for the correction commands as long as the signal has not been changed.

The measurements taken from the C-Track are compared with the desired values, which are already obtained by the HandyProbe. If the difference for each axis is greater than the required precision ($\pm 0.05\, mm$), the controller corrects the position of the robot end-effector by applying offsets along X,Y,Z directions.

After satisfying the desired accuracy, the software stops sending corrections to the controller, the digital output is switched to OFF and the last corrected position is recorded and loaded to the robot controller (PR[38] in Figure 6.3). This technique is called the automatic learning which will be explained in Section 6.1.1.

In this experiment the sampling interval is equal the C-Track measurement rate and the control interval is after taking 10 measurements:

$$T_s = 0.034\,ms \tag{6.1a}$$

$$T_c = 0.34\,ms \tag{6.1b}$$

where T_s and T_c represent the sampling interval and control interval respectively.

Dynamic position correction control algorithm is implemented on the designed task where the robot should insert 4 holes on different sides of the cube. The required precision for this task is $\pm 0.05mm$ on each axis. The task is repeated twice. In the first round, the robot controller uses the C-Track information and corrects the position of the robot end-effector through the developed software. In the second round, the program uses the learned positions obtained from the first round and repeats the same task. Figures 6.5 – 6.8 show the position error for each hole the first round.

As can be seen in the figures, due to the large initial error on each axis, the robot could not insert the hole without touching the cube. The insertion can be performed only after satisfying the desired precision.

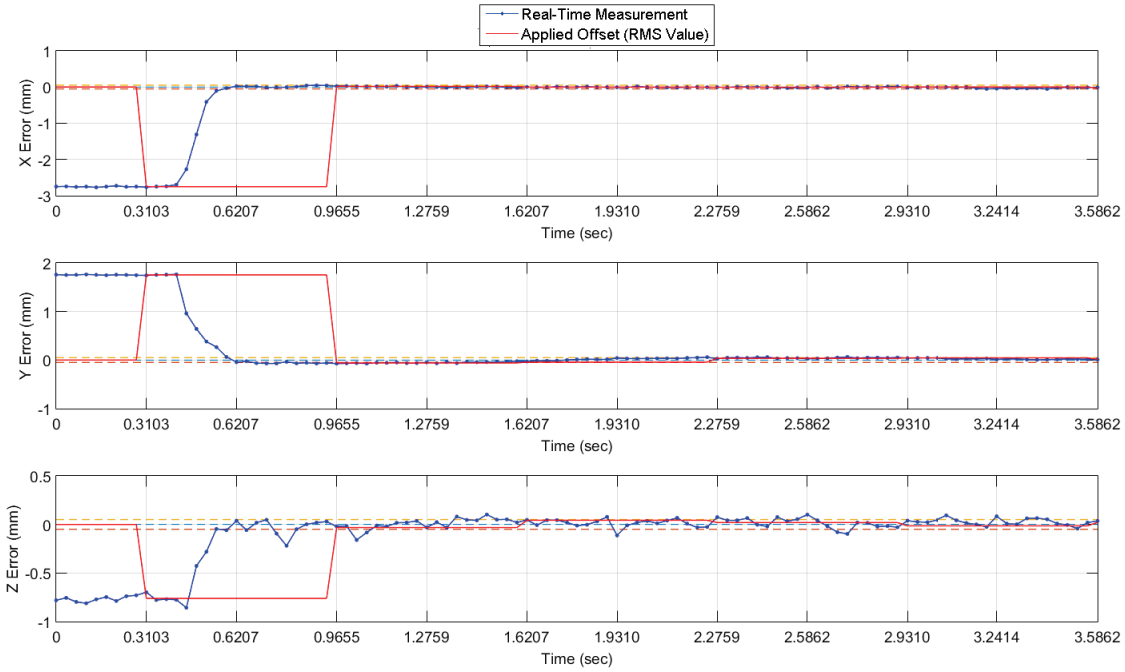


Figure 6.5: Position error for hole 1 (first round)

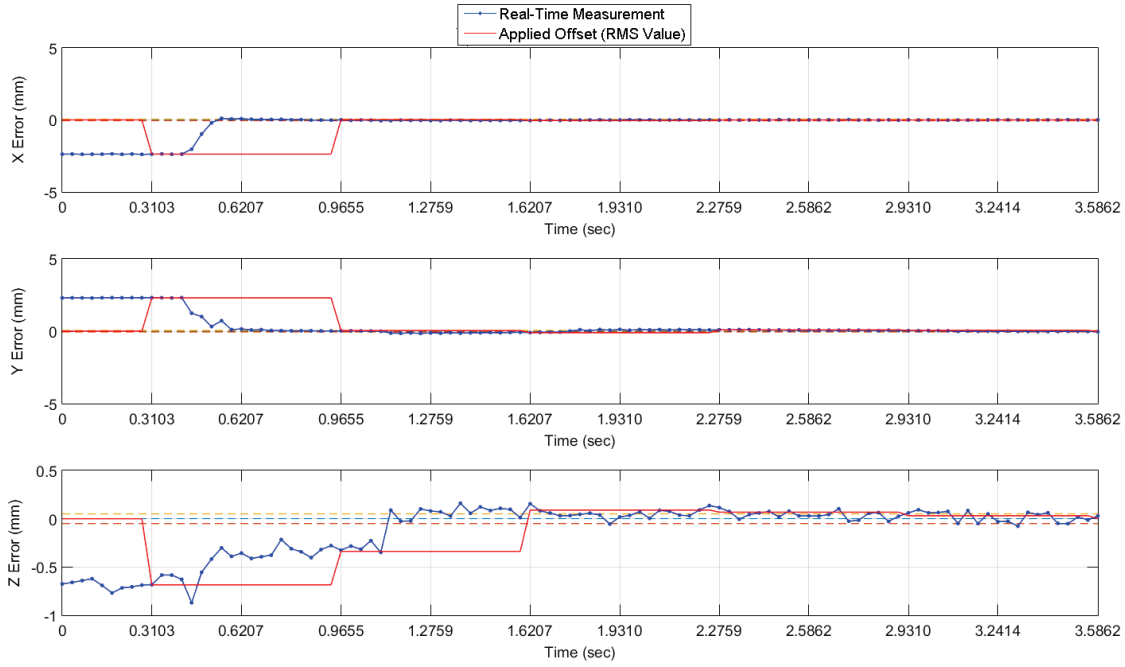


Figure 6.6: Position error for hole 2 (first round)

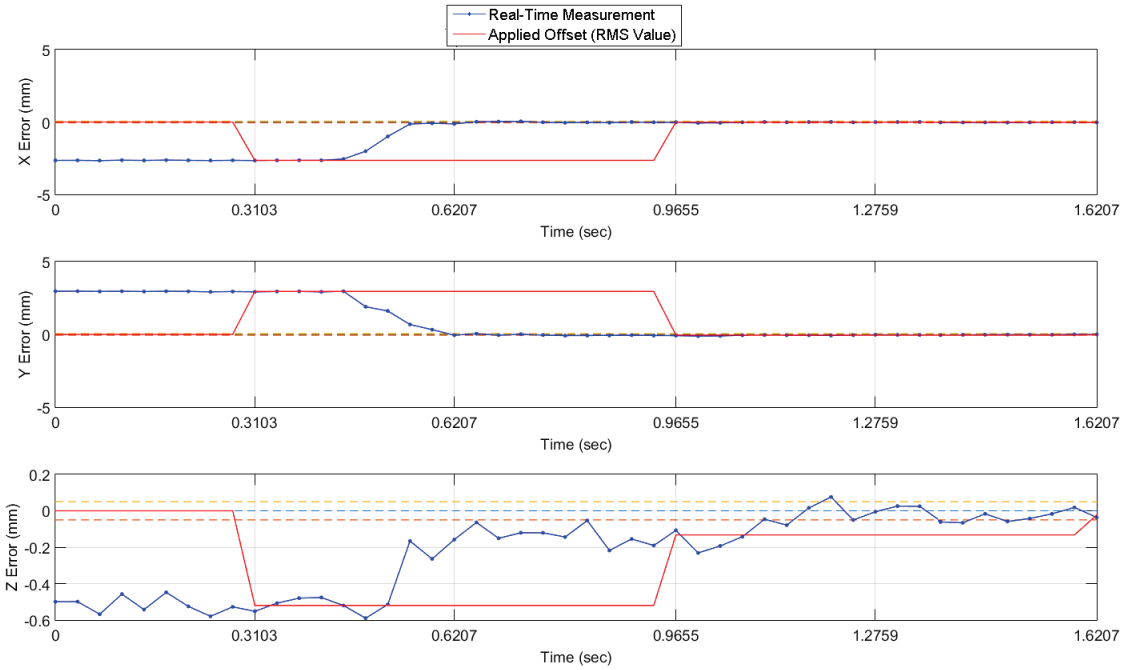


Figure 6.7: Position error for hole 3 (first round)

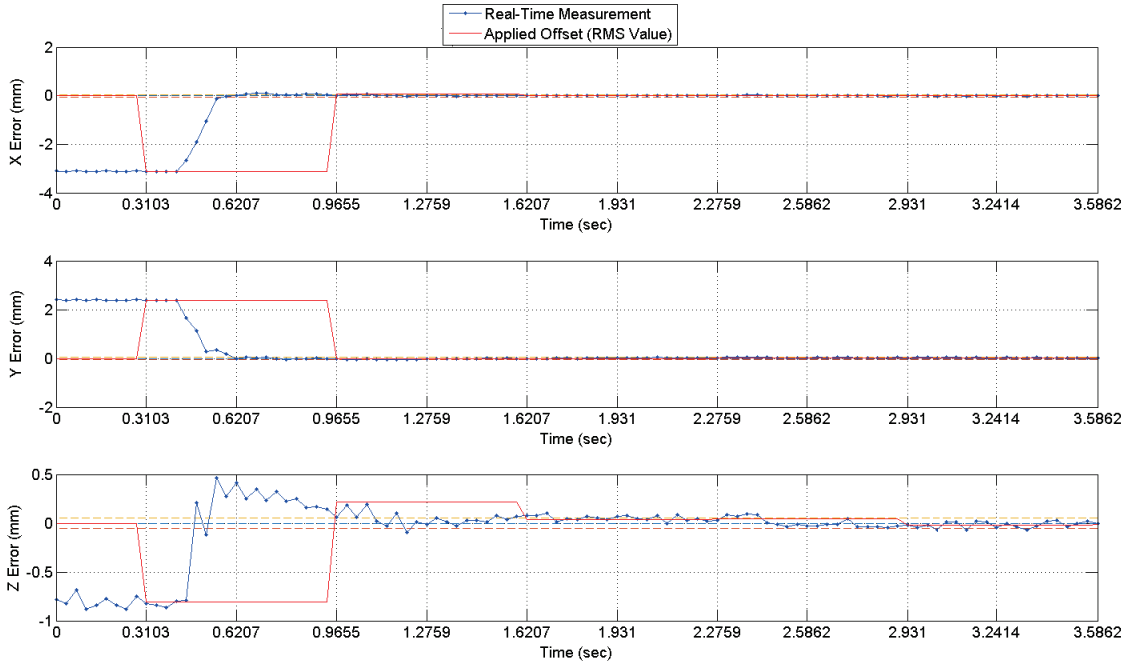


Figure 6.8: Position error for hole 4 (first round)

6.1.1 Automatic Learning Procedure in dynamic position correction

This section illustrates the effect of the automatic learning algorithm on the performance of the dynamic position correction algorithm. Using the proposed automatic learning method, the last corrected pose of the robot is recorded after the correction procedure is finished. The recorded positions are used in the program instead of the original probed values while the task is repeated.

As mentioned in Section 2.2, industrial robots can repeat the same task with a great accuracy [14]. For instance, the repeatability of the calibrated FANUC M20-iA robot is $\pm 0.08\,mm$ [11]. Therefore, if the software can record the last corrected pose of the robot and load it to the program, the robot can move directly to the corrected pose instead of the original probed once. As a result, the initial error on each axis can be reduced for the task.

In this research, the automatic learning task is performed through the developed software by recording the latest corrected pose (when the precision on each axes is satisfied). When the task is completed by the robot once, the learned positions are loaded to the robot TP program by the developed software. Figures 6.9 – 6.12 show the error on each axis in the second round when the program uses the learned poses.

The experiment results show that the response of the system is smoother and faster since the initial errors has been decreased significantly due to the corrections performed in the first round. The response time is reduced from 3.58 seconds to 1.62 seconds (54% improvement)

Learning procedure can be applied to the systems with repeating tasks, specially the industrial robots. Considering this strategy, the operator needs to run the program once and the software can automatically learn and record the last corrected pose for each hole which can be used for the next cycles.

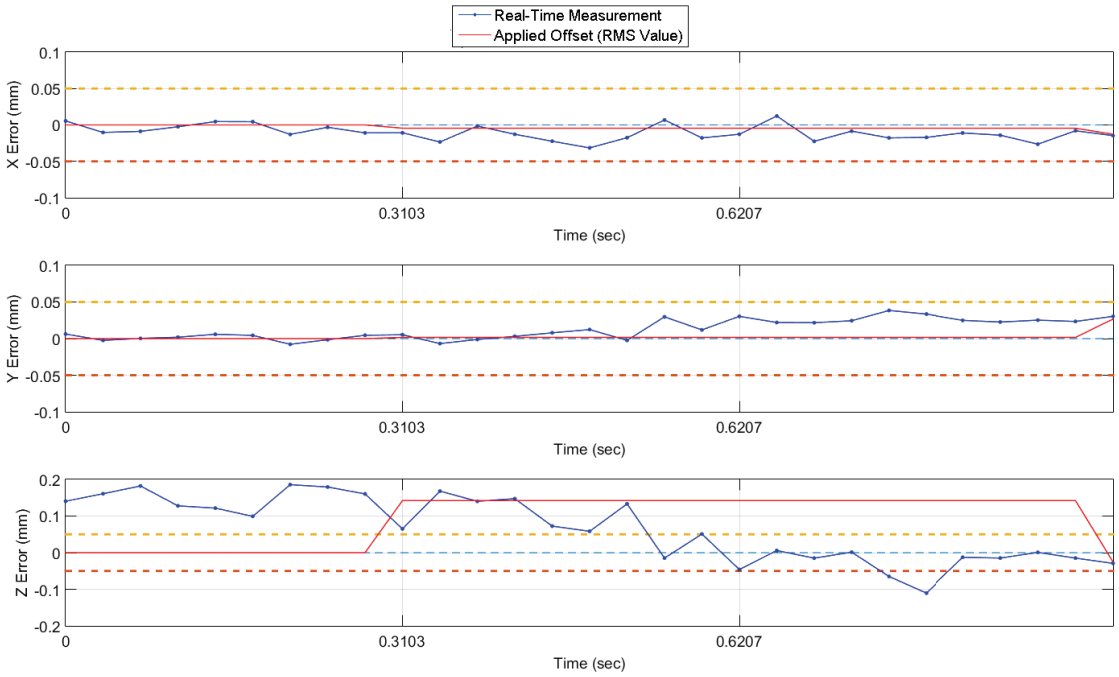


Figure 6.9: Position error for hole 1 (second round)

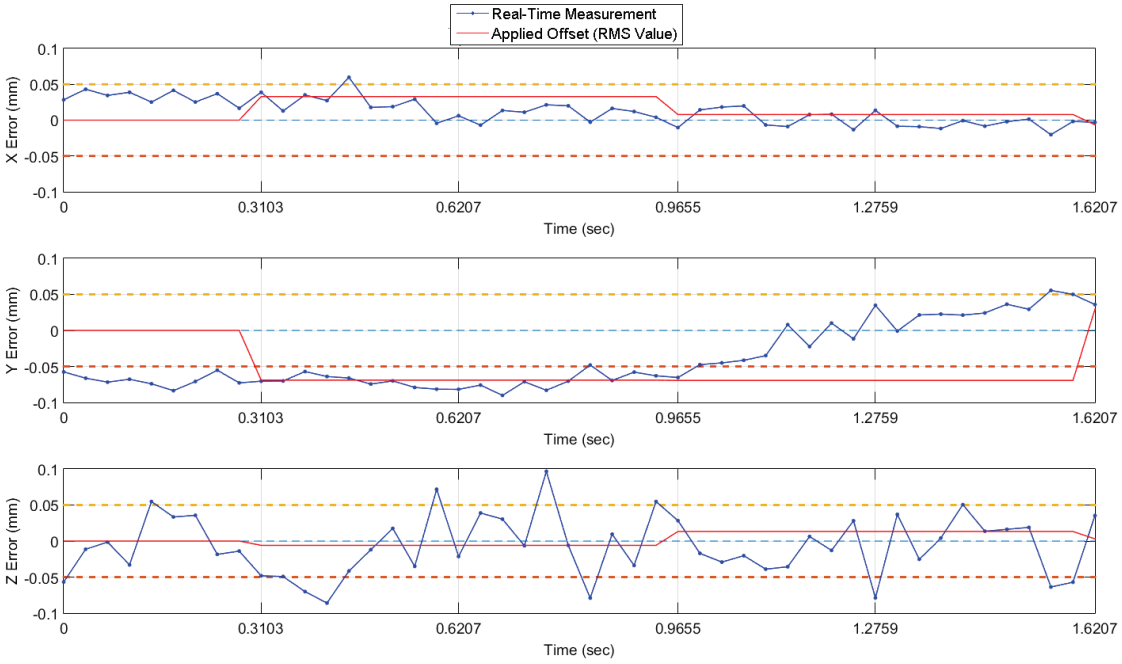


Figure 6.10: Position error for hole 2 (second round)

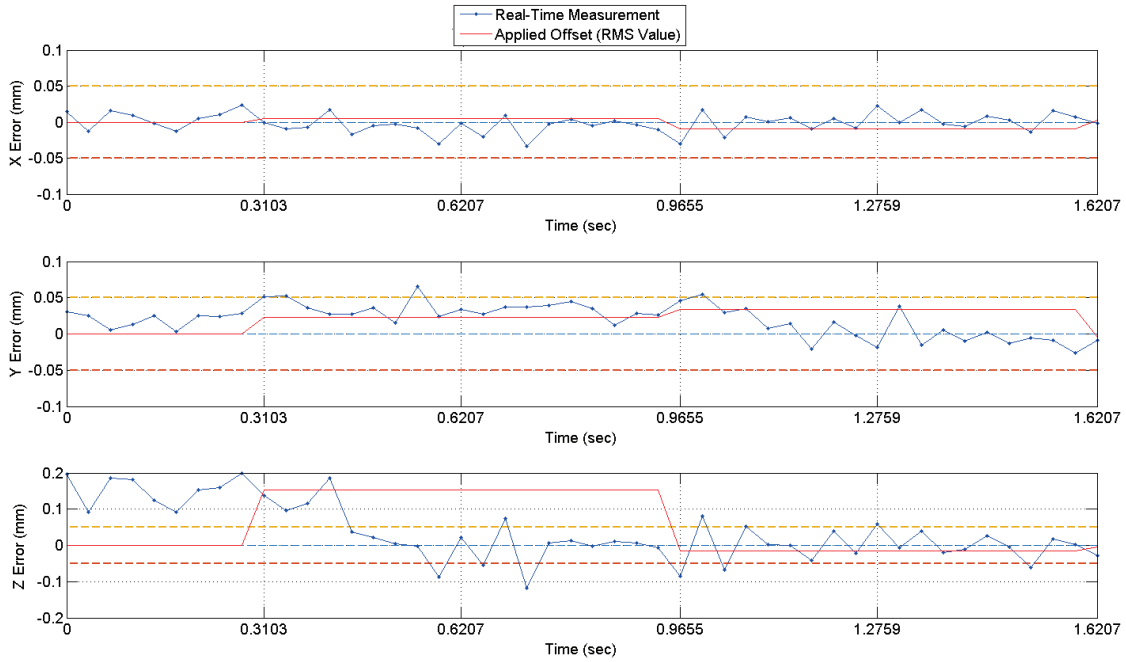


Figure 6.11: Position error for hole 3 (second round)

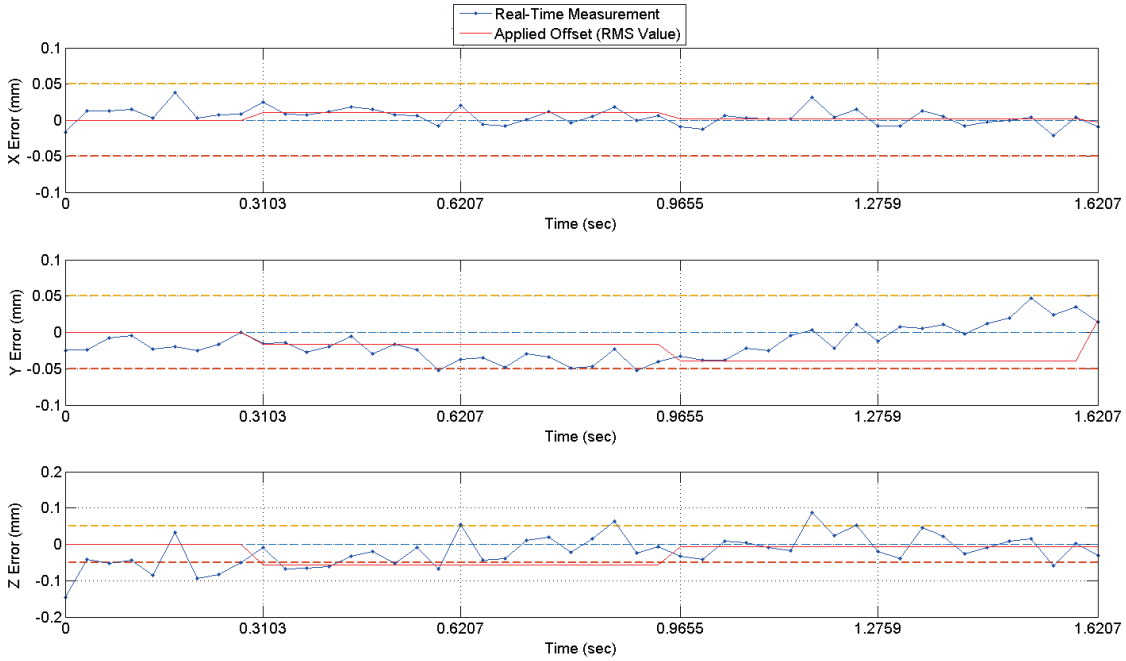


Figure 6.12: Position error for hole 4 (second round)

6.1.2 Limitations

Although this experiment can fulfil the task successfully, it is very sensitive to the measurements and the workspace, as mentioned in Section 6.1. Since the desired values are obtained by probing, the cube needs to be fixed in the workspace. Any changes in the position of the cube or C-Track results in repeating all the procedures and measurements, which is time consuming and infeasible for the industrial applications.

In addition, this experiment shows only the position control, while for the majority of industrial applications, it would also be necessary to correct and control the orientation of the robot. Moreover, there are several factors affecting the HandyProbe measurements, such as the operators skills, vibrations in the environment and smoothness of the probed surfaces. The industry is seeking for more advanced and adaptive methods that can perform the real-time correction task without keeping the target object (cube) at the same position.

6.2 Experiment 2: Dynamic Pose Correction to insert the holes on a moving cube

The limitations from the first experiment motivate us to develop a new method that would resolve all the mentioned problems based on the industry demand. In this experiment the robot is able to track a hole on a moving cube dynamically. In addition to the position, the orientation of the end-effector is controlled, therefore, the algorithm which is used in this experiment is different from the first one.

In order to eliminate the effect of the operator skills and the surface smoothness in measurements, instead of using the HandyProbe, the robot tool model is used to identify the desired pose for inserting the holes. The whole process can be performed using the robot, the C-Track and the developed software, without manual measurements.

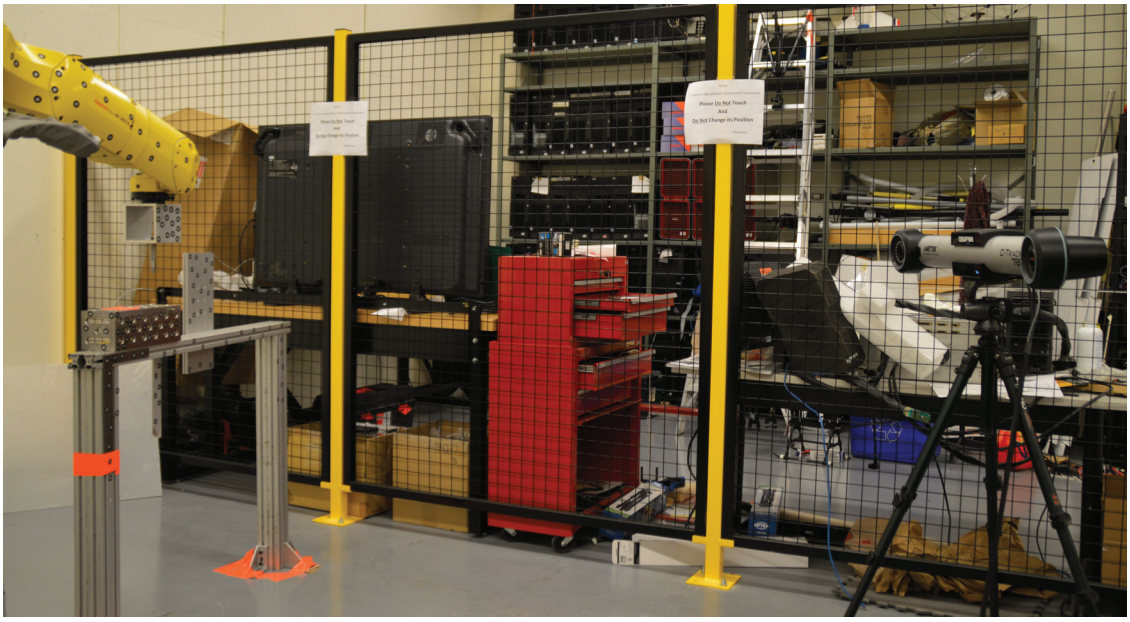


Figure 6.13: Setup for the second experiment

The capability of this algorithm to control the pose of the robot and update the position of the hole according to the cube movement is one of the key advantages of this method. The configuration of the C-Track, robot, cube and aluminium bar for this experiment is shown in Figure 6.13.

As mentioned earlier, all the measurements are taken considering the robot tool model, cube model and the user frame. All the calculations for the desired pose, current pose, errors and the corrections are performed within the developed software based the created models and frames. Figure 6.14 shows the schematic of the frames and the relations used in this method to calculate the offsets.

In Figure 6.14, $\{T\}$, $\{U\}$ and $\{C\}$ represent the tool frame, user frame and cube frame respectively. These frames are represented with respect to the C-Track frame by the homogeneous transformation matrices ${}^{S}_{T}H$, ${}^{S}_{U}H$ and ${}^{S}_{C}H$. As mentioned in Section 5.2.4, the coordinates of the user frame is known with respect to the C-Track frame (${}^{S}_{U}H$). Furthermore, the coordinates of the tool frame and cube frame are known with respect to the C-Track as

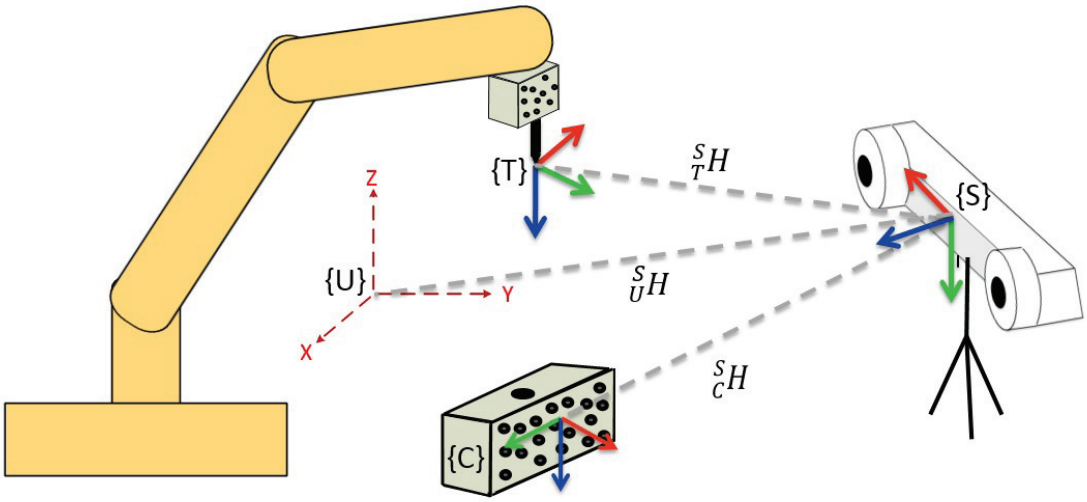


Figure 6.14: Schematic of the relations used for error calculation

long as the cameras can see these models.

The software interface designed for this experiment is shown in Figure 6.15. In this test, in addition to the dynamic referential and tool model, the cube model needs to be created and loaded to the software. The position of each hole is detected with respect to the cube model. The approach point is defined as the coordinates of the TCP when it is exactly on top of the hole center and the Z direction of the tool frame is perpendicular to the cube plane as shown in Figure 6.16. In the proposed method, this point is considered as the desired point in the control system.

The desired point in the software is defined by using the *Teach Approach* button when the robot is jogged by the teach pendant to the center of the hole. In order to make sure the approach point is recorded accurately, the robot is jogged in the Z direction of the tool frame; if the robot insert the hole without touching the cube, the desired point is acceptable and can be recorded in the software.

Since the C-Track measurements are noisy, as explained in Section 3.4, we cannot rely on

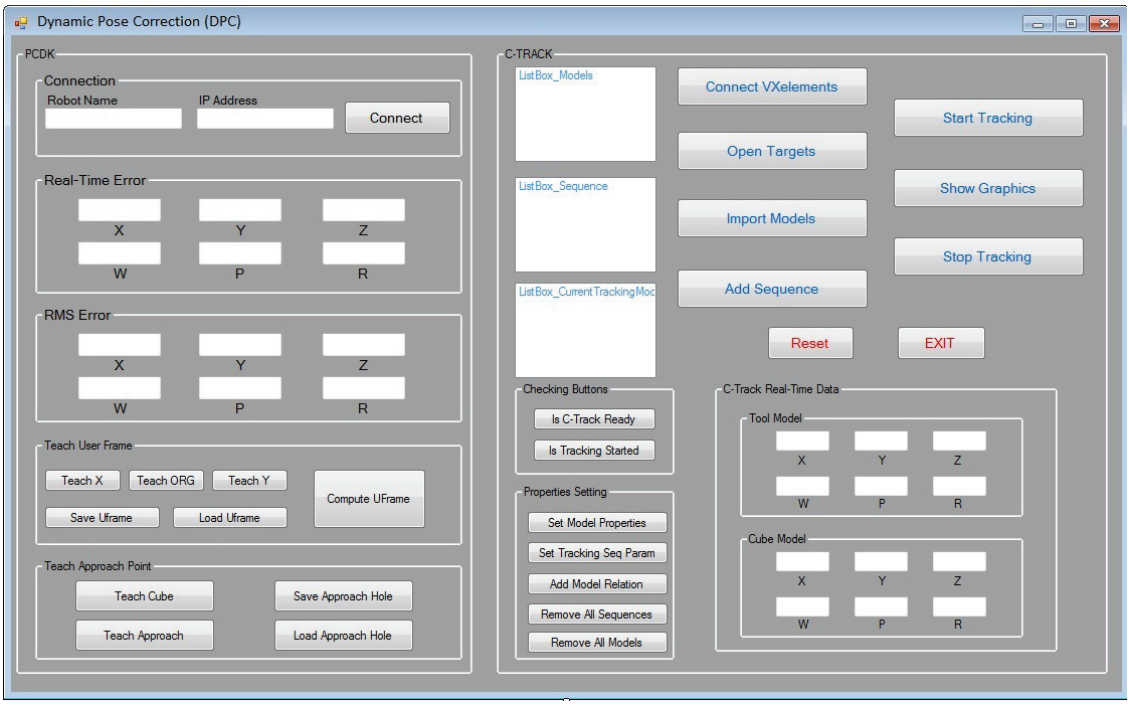


Figure 6.15: Final interface for dynamic pose correction

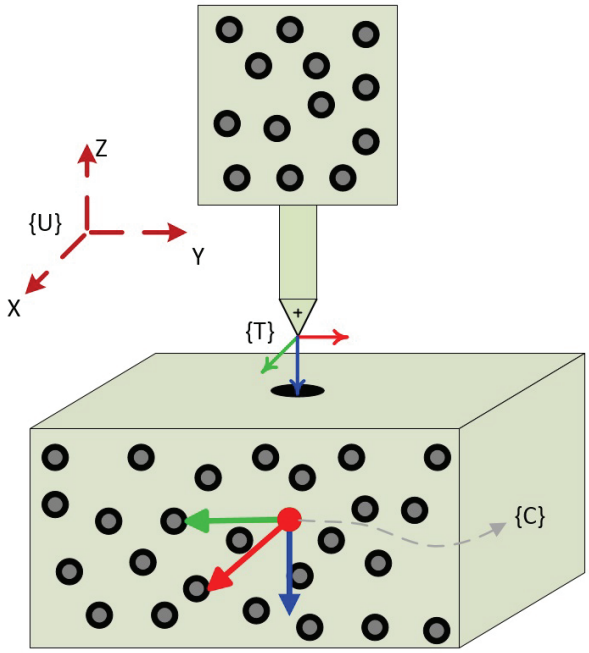


Figure 6.16: Schematic representation of the Approach point

only one measurement. Therefore the software calculates the RMS value of 10 measurements while recording the approach point.

After finding the approach point with respect to the sensor frame, the coordinates of this point with respect to the cube frame, $\{C\}$, can be calculated by:

$$ {}^{C}_{T}H_{A} = {}^{S}_{C}H^{-1} \times {}^{S}_{T}H_{A} \tag{6.2} $$

where, ${}^{C}_{T}H_{A}$, represents the coordinates of the tool with respect to the cube frame when the robot is at the approach point. In order to insert the hole, the robot should reach to the approach point first, therefore, this relation is recorded in the software and is used as the desired pose for the robot.

Using the recorded approach point, the coordinates of the approach point with respect to the sensor frame can be calculated as follows:

$$ {}^{S}_{T}H_{A} = {}^{S}_{C}H \times {}^{C}_{T}H_{A} \tag{6.3} $$

where ${}^{S}_{T}H_{A}$, represents the desired homogeneous transformation matrix of the tool frame with respect to the sensor frame.

$$ {}^{S}_{T}H_{Des} = {}^{S}_{T}H_{A} \tag{6.4} $$

As can be seen in (6.4), ${}^{S}_{T}H_{Des}$, can be updated dynamically according to the position and orientation of the cube model (${}^{S}_{C}H$). Therefore, the robot can track the hole while the cube is moving.

Since the current pose of the robot with respect to the sensor frame is calculated dynamically by the C-Track, and the desired pose is obtained from (6.4), the error can be obtained

as follows:

$$ {}^{S}_{T}H_{E} = {}^{S}_{T}H_{Des} - {}^{S}_{T}H_{Cur} \tag{6.5} $$

As explained in Section 5.3, the corrections are applied to the robot with respect to the user frame. Therefore, the transformation matrix of the error obtained in (6.5), can be transformed into the user frame, using the following equation:

$$ {}^{U}_{T}H_{E} = {}^{S}_{U}H^{-1} \times {}^{S}_{T}H_{E} \tag{6.6} $$

where

$$ {}^{U}_{T}H_{E} = \begin{bmatrix} {}^{U}_{T}R_{3\times3} & {}^{U}_{T}t_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{6.7} $$

In (6.7), ${}^{U}_{T}t$ represents the position error with respect to the user frame.

$$ {}^{U}_{T}t = \begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{bmatrix} \tag{6.8} $$

Since the angle differences can not be directly transformed into the user frame, the orientation error with respect to the user frame is calculated from the equivalent angle axis method as explained in Section 5.3. In order to find the orientation error, we need to find the transformation matrix between the current pose and the desired pose of the tool. This matrix can be computed by:

$$ {}^{Des}_{Cur}H = {}^{S}_{T}H^{-1}_{Des} \times {}^{S}_{T}H_{Cur} \tag{6.9} $$

where ${}^{Des}_{Cur}H$, represents the coordinates of the current pose of the robot tool frame with respect to the desired tool frame. The equivalent angle and equivalent axis between these

two frames can be computed using (5.7) and (5.8) respectively. According to [15], the equivalent axis $\hat{K}$ obtained from these equations is represented in the current tool frame. This vector can be transformed into user frame by:

$$^{U}\hat{K} = {}^{U}_{Cur}R \times {}^{Cur}\hat{K} \tag{6.10}$$

where ${}^{U}_{Cur}R$, is the rotation matrix of the tool frame at current pose with respect to the user frame which is extracted from the following matrix:

$${}^{U}_{Cur}H = {}^{S}_{U}H^{-1} \times {}^{S}_{Cur}H = \begin{bmatrix} {}^{U}_{Cur}R_{3\times3} & {}^{U}_{Cur}t_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{6.11}$$

Using the $^{U}\hat{K}$ and the calculated angles, the new rotation matrix can be computed from (5.9). Comparing this rotation matrix with the general form of the rotation matrix according to Euler angles, (3.33), the W-P-R values can be calculated by the algorithm explained in Section 5.3. The calculated W-P-R are the orientation errors which can be applied to the robot through the developed software.

In this experiment, the robot moves from an initial pose towards the desired coordinates (approach pose) using the DPM option. When the error between the pose of the tool frame and desired pose satisfies the required precision ($\pm 0.05\,mm$ for position and $\pm 0.05\,deg$ for orientation), the robot will insert the hole. When the position and orientation of the cube is changed, the desired pose will be updated by the software real-time according to the new location of the cube. Therefore, the robot can be guided to the updated desired pose and track the hole on the cube automatically.

The dynamic pose correction algorithm is implemented in two different methods. In the first experiment, the wait function is considered in the software according to the maximum

offset limit for each axis. This function allows the robot to finish the applied offsets first and then to take new measurements by the C-Track, since the RMS technique is reliable only when the robot is stationary as explained in Section 3.4.

The wait time in this experiment is set to be $10 \times T_s$. According to the DPM specifications, the maximum offset is $0.2\,mm$ per $8\,ms$. Therefore, during 10 sampling interval ($344.8ms$), the maximum offset that can be applied to the robot can be calculated as follows:

$$d_{max} = \frac{0.2 * 344.8}{8} = 8.6\,mm \tag{6.12}$$

This value is considered as the saturation for the controller output.

Since the offset values are large only at the beginning, this waiting time is not reasonable when the offsets are very small. According to this technique, the robot can not adaptively change the waiting time based on the offset values. The experiment results using this method are shown in Figure 6.17. Since the cube is moving during the experiment, the initial error is different for each segment.

As illustrated in Figure 6.17, the response of the robot is not smooth due to the saturation. The effect of the saturation is obvious at the beginning when the initial error is large.

This issue is solved in the second method by checking the remaining values of the applied offsets. Using this method, the new RMS values are calculated only if the applied offsets are consumed completely by the robot.

The experiment results using the second method are shown in Figure 6.18. As can be seen in this Figure, the movement of the robot is smoother and the response is faster compared to the first method, since the waiting time is decided by the offset value.

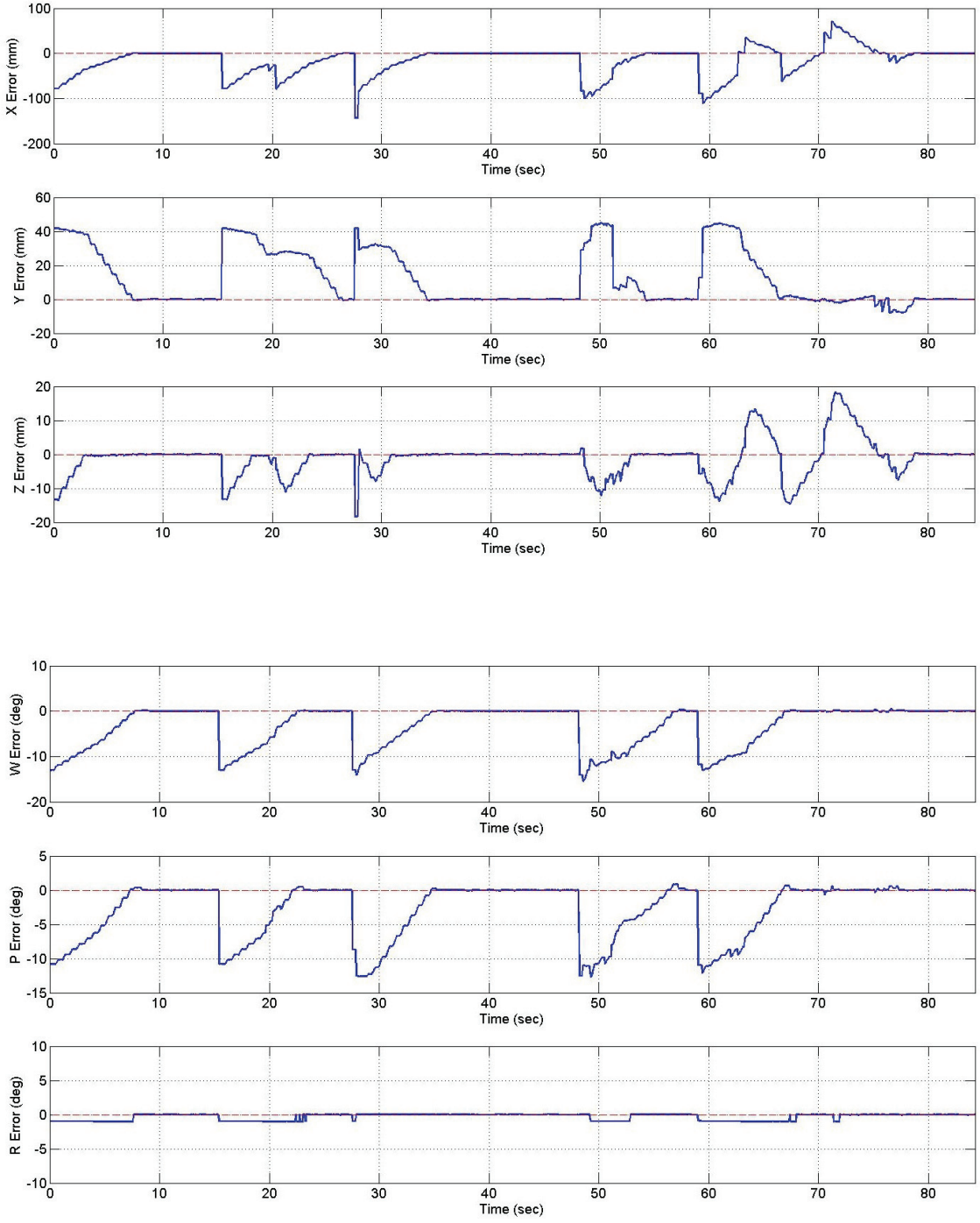


Figure 6.17: Pose correction to insert a hole on a moving cube without offset limitation

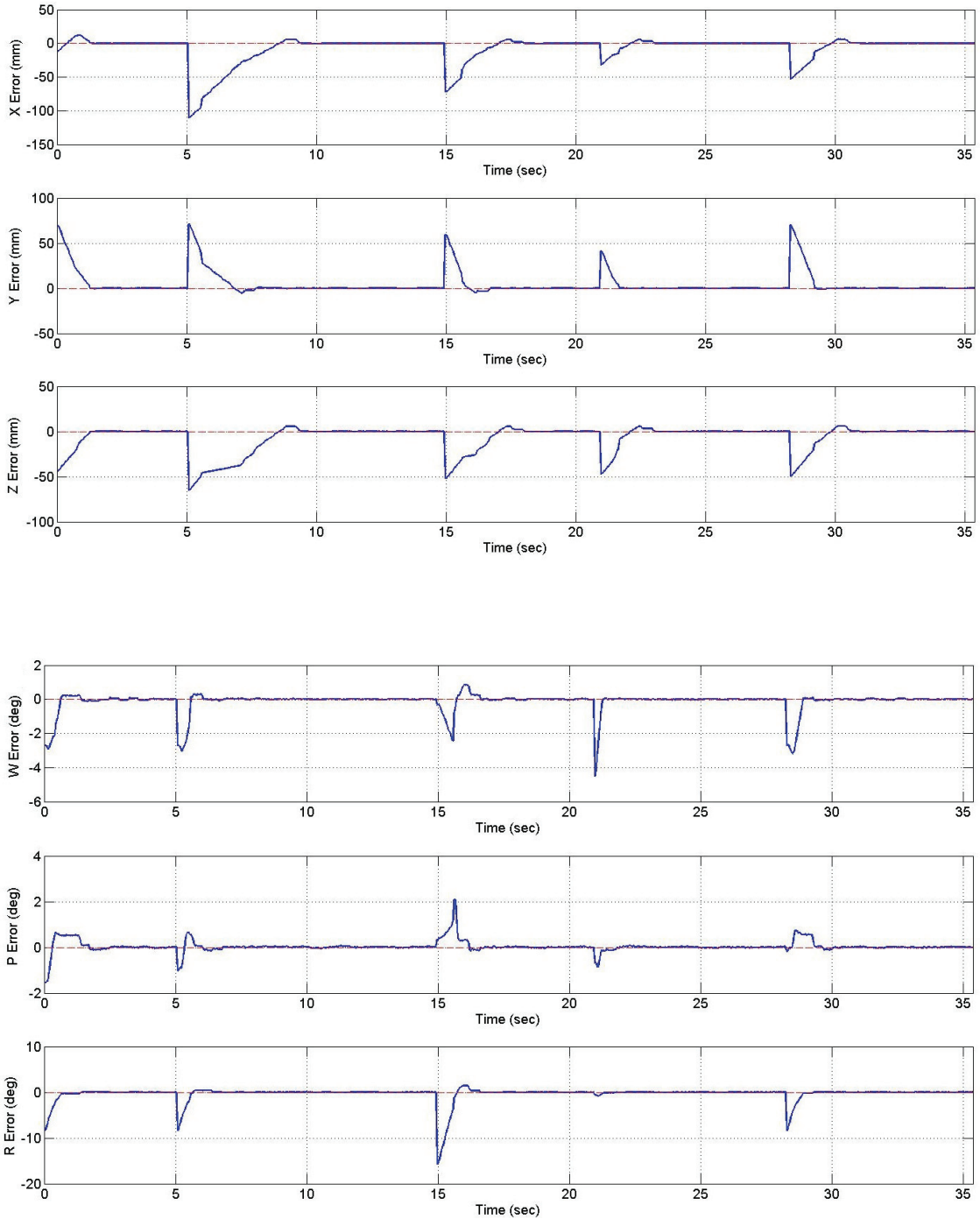


Figure 6.18: Pose correction to insert a hole on a moving cube without offset limitation

Table 6.1: Control Algorithm Performance Assessment

	Without Control Algorithm						With Control Algorithm					
	Position			Orientation			Position			Orientation		
	X	Y	Z	W	P	R	X	Y	Z	W	P	R
Response time	12 seconds						3.6 seconds					
Settling time (2%)	5 sec			4 sec			0.1 sec			1 sec		
Percent Overshoot	60%	50%	20%	40%	17%	80%	0	0	0	0	0	0

6.3 Control Algorithm Performance Evaluation

In this section, the performance of the developed control algorithm used in this project is evaluated. In order to assess the performance, the dynamic pose control experiment for inserting the hole is performed two times. First, the control algorithm is neglected and the response of the system is recorded. Then the control algorithm is applied to the same experiment and the errors for each axes is recorded.

The experiment results show that the task can be completed without the developed algorithm with the same desired precision but the response is not as smooth as using the proposed control strategy. Figures 6.19–6.22 show the errors for each axis.

As can be seen in Figures 6.19 and 6.20, the required precision is satisfied after 12 seconds while using the control algorithm, this time is reduced to less than 3.6 seconds which is 70% improvement in the response of the system. Table 6.1 compares the time response specifications for these experiments.

In addition to the response time, the robot is directly guided to the desired pose using the proposed control algorithm. Without this strategy, the response will have an overshoot for each axis, which may cause serious damage to the robot and environment when used in industry.

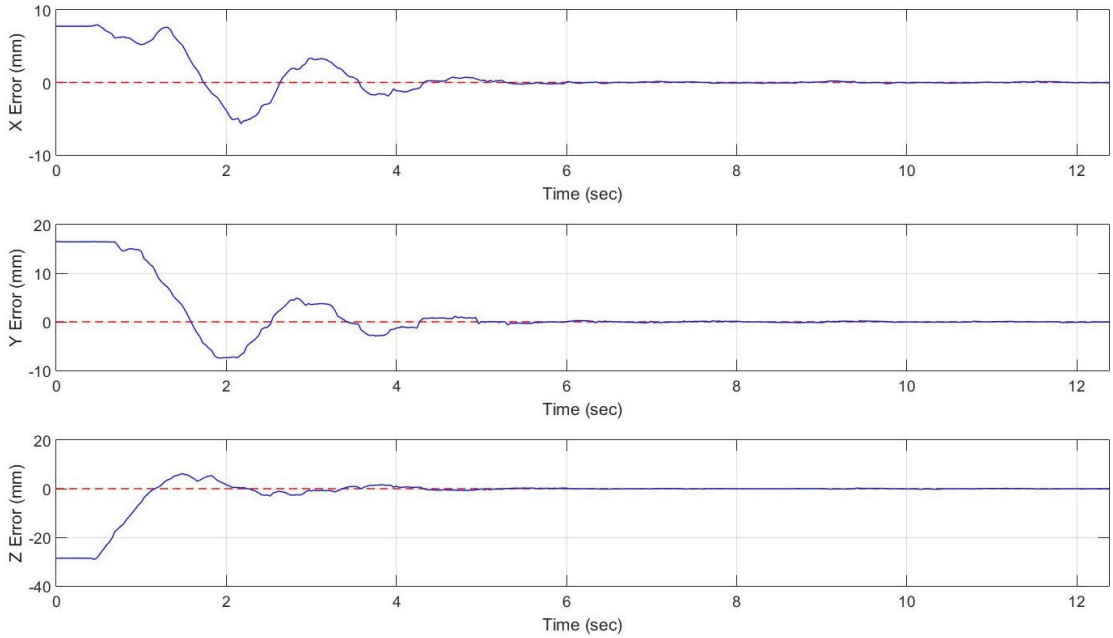


Figure 6.19: Position error for inserting the hole without control algorithm

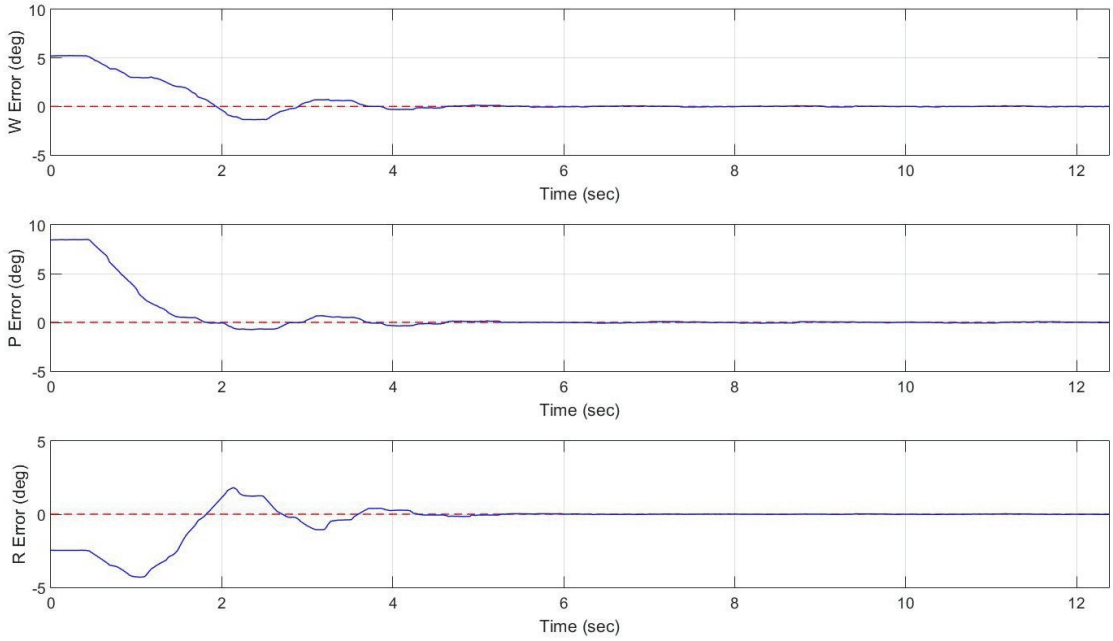


Figure 6.20: Orientation error for inserting the hole without control algorithm

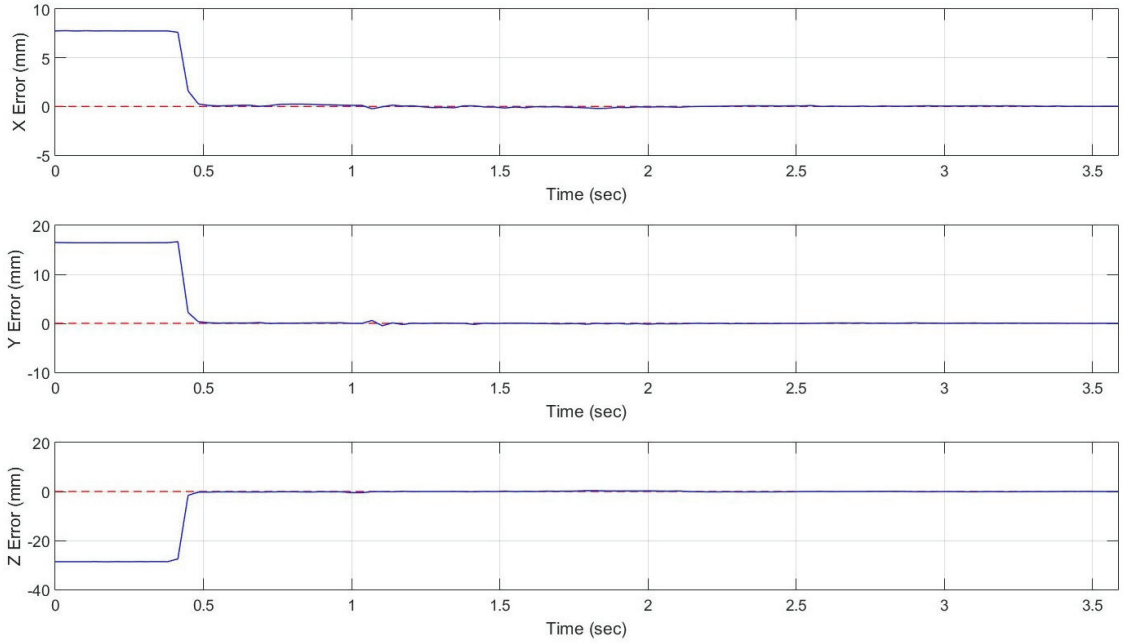


Figure 6.21: Pose correction to insert the hole with control algorithm

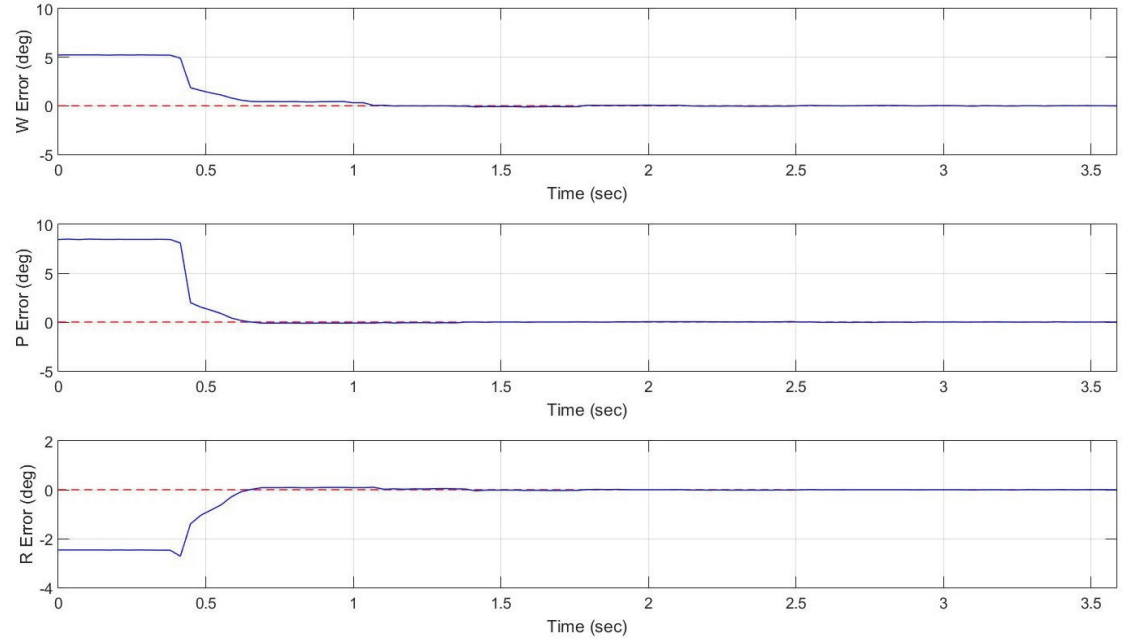


Figure 6.22: Pose correction to insert the hole with control algorithm

6.4 Summary

In this chapter, the experiment results for the dynamic position control and pose control of the robot end-effector are presented. For both experiments, the results show that the required task is fulfilled and satisfied the desired accuracy.

In the first experiment, the coordinates of each hole are measured manually using the HandyProbe. However, during the second experiment, the software performs the procedures automatically. Additionally, both the position and orientation of the robot end-effector are controlled in the second experiment. The proposed algorithm for this experiment can follow the hole dynamically while the cube is moving. The performance of the proposed control strategy is evaluated and the results are provided in the last part of this chapter.

Chapter 7

Conclusion and Future Works

In this chapter, the main conclusions of this thesis are summarized according to the obtained results and then some possible extensions and future works are suggested as well.

7.1 Research Summary

This research presents a novel real time pose control algorithm for an industrial robot (Fanuc M20-iA) using Creaform 3D measurement sensor (C-Track). The main contributions of this thesis are as follows:

- The noise associated with the sensor data is filtered by using an RMS value of 10 measurements. This idea reduced the noise of the data for a static point measurement by 75%.
- Using the proposed algorithm, both the position and orientation of the robot, are controlled while the task is being performed.
- The developed DPC algorithm, can guide the industrial robot to any desired pose in the FOV of the C-Track, accurately and precisely by using the real-time feedback

measurements.

- Using the developed dynamic pose control method, the robot can perform stationary tasks such as drilling, spot welding and riveting with the absolute accuracy of $\pm 0.05\,mm$ for position and $\pm 0.05\,deg$ for the orientation.

- The developed algorithm is independent of the robot kinematics; therefore, the proposed strategy can replace the costly and time-consuming available calibration methods.

- Both pose calculation and pose dynamic correction controller are implemented in a user-friendly software in Visual Basic.

- The developed software allow the operator to work with the interface and complete the tasks easily.

- Using the proposed algorithm for the last experiment, all the calculation and measurements are performed through the software which eliminate the errors caused by the operator skills such as shaking during measurements.

- Although the robot can perform the same task with the required accuracy even without the proposed control algorithm, the response of the system is not satisfactory due to the large overshoot and long convergence time. The results prove that the response time of the system has been improved 70% by using the proposed control strategy.

7.2 Future Works

Although the performance of the robot for the dynamic pose control is tested and verified using two different experiment setups for a static pose, the possibility of correcting different

tasks performed by the robots can still be challenging. According to the noises of the current measurement sensor and limitations of the data acquisition frequency, correcting the path of the robot while moving on a line or circular trajectory can be evaluated.

The control algorithm used in this project is based on the PID controller. The PID gains are identified by trial and error for the current robot. In order to apply this algorithm to different types of industrial robots it is necessary to find an auto-tuning method for the controller gains. Iterative learning control (ILC) can be used as an advanced method for this goal which uses the past errors to predict the future errors.

In addition to PID controller, other types of model-based and non-model-based controllers can be applied. Model predictive control (MPC), sliding mode control and fuzzy controller are some examples of the control algorithms which can be used and the results can be compared with the proposed control algorithm.

The current developed software and algorithm is tested specifically on Fanuc M20-iA robot, but the proposed dynamic pose control strategy can be applicable to different types of industrial robots such as ABB and KUKA.

Bibliography

[1] A. Mohebbi, M. Keshmiri, and W. F. Xie, "An eye-in-hand stereo visual servoing for tracking and catching moving objects," in *Control Conference (CCC), 2014 33rd Chinese*. IEEE, 2014, pp. 8570–8577.

[2] J. F. Engelberger, *Robotics in practice: management and applications of industrial robots*. Springer Science & Business Media, 2012.

[3] R. P. Judd and A. B. Knasinski, "A technique to calibrate industrial robots with experimental verification," *Robotics and Automation, IEEE Transactions on*, vol. 6, no. 1, pp. 20–30, 1990.

[4] G. Duelen and K. Schröer, "Robot calibrationmethod and results," *Robotics and Computer-Integrated Manufacturing*, vol. 8, no. 4, pp. 223–231, 1991.

[5] A. Nubiola and I. A. Bonev, "Absolute calibration of an abb irb 1600 robot using a laser tracker," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 1, pp. 236–245, 2013.

[6] M. Abderrahim, A. Khamis, S. Garrido, and L. Moreno, "Accuracy and calibration issues of industrial manipulators," *Industrial Robotics: Programming, Simulation and Applications*, p. 131, 2007.

[7] Wikipedia, "Industrial robot — wikipedia, the free encyclopedia," 2016. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Industrial_robot&oldid=704417506

[8] M. Design, "The difference between cartesian, six-axis, and scara robots," 2016. [Online]. Available: http://machinedesign.com/motion-control/difference-between-cartesian-six-axis-and-scara-robots

[9] ABB, "Abb robotics," 2016. [Online]. Available: http://new.abb.com/products/robotics

[10] H. A. Robotics, "Pishrobot," 2007. [Online]. Available: http://www.pishrobot.com/en/products/robotic_arms.htm

[11] FANUC, "Fanuc america corporation," 2016. [Online]. Available: http://www.fanucamerica.com/

[12] R. Ranch, "Types of robots." [Online]. Available: http://prime.jsc.nasa.gov/ROV/types.html

[13] Wikipedia, "Scara — wikipedia, the free encyclopedia," 2015, [Online; accessed 16-March-2016]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=SCARA&oldid=673558515

[14] P. Fixel, "Absolute accuracy marketing presentation," *ABB Automation Technologies AB*, vol. 51, 2006.

[15] J. J. Craig, *Introduction to robotics: mechanics and control*. Pearson Prentice Hall Upper Saddle River, 2005, vol. 3.

[16] Wikipedia, "Off-line programming (robotics) — wikipedia, the free encyclopedia," 2016.

[Online]. Available: https://en.wikipedia.org/w/index.php?title=Offline_programming_(robotics)&oldid=709670155

[17] B. Greenway, "Robot accuracy," *Industrial Robot: An International Journal*, vol. 27, no. 4, pp. 257–265, 2000.

[18] M. Summers, "Robot capability test and development of industrial robot positioning system for the aerospace industry," *SAE transactions*, vol. 114, no. 1, pp. 1108–1118, 2005.

[19] A. Elatta, L. P. Gen, F. L. Zhi, Y. Daoyuan, and L. Fei, "An overview of robot calibration," *Information Technology Journal*, vol. 3, no. 1, pp. 74–78, 2004.

[20] A. Nubiola, M. Slamani, A. Joubair, and I. A. Bonev, "Comparison of two calibration methods for a small industrial robot based on an optical cmm and a laser tracker." *Robotica*, vol. 32, no. 3, pp. 447–466, 2014.

[21] R. DeVlieg and T. Szallay, "Applied accurate robotic drilling for aircraft fuselage," *SAE International Journal of Aerospace*, vol. 3, no. 1, pp. 180–186, 2010.

[22] Y. Shirai and H. Inoue, "Guiding a robot by visual feedback in assembling tasks," *Pattern recognition*, vol. 5, no. 2, pp. 99–108, 1973.

[23] G. J. Agin, *Real time control of a robot with a mobile camera*. SRI International, 1979.

[24] P. I. Corke *et al.*, *Visual Control of Robots: high-performance visual servoing*. Research Studies Press Baldock, 1996.

[25] G.-Q. Wei, K. Arbter, and G. Hirzinger, "Real-time visual servoing for laparoscopic surgery. controlling robot motion with color image segmentation," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 16, no. 1, pp. 40–45, 1997.

[26] A. O. Innovation, "Robotic surgery innovation at its finest," 2016. [Online]. Available: http://www.ageofinnovation.org/robotic-surgery-innovation-at-its-finest/

[27] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 4, pp. 82–90, 2006.

[28] D. Kragic, H. I. Christensen *et al.*, "Survey on visual servoing for manipulation," *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, vol. 15, 2002.

[29] E. Malis, F. Chaumette, and S. Boudet, "21/2d visual servoing," *Robotics and Automation, IEEE Transactions on*, vol. 15, no. 2, pp. 238–250, 1999.

[30] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 5, pp. 651–670, 1996.

[31] G. Duelen and K. Schröer, "Robot calibrationmethod and results," *Robotics and Computer-Integrated Manufacturing*, vol. 8, no. 4, pp. 223–231, 1991.

[32] S. Hayati and M. Mirmirani, "Improving the absolute positioning accuracy of robot manipulators," *Journal of Robotic Systems*, vol. 2, no. 4, pp. 397–413, 1985.

[33] J. Chen and L.-M. Chao, "Positioning error analysis for robot manipulators with all rotary joints," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 6, pp. 539–545, 1987.

[34] B. W. Mooring and T. Pack, "Aspects of robot repeatability." *Robotica*, vol. 5, no. 3, pp. 223–230, 1987.

[35] M. R. Driels and W. E. Swayze, "Automated partial pose measurement system for manipulator calibration experiments," *Robotics and Automation, IEEE Transactions on*, vol. 10, no. 4, pp. 430–440, 1994.

[36] Z. Roth, B. Mooring, and B. Ravani, "An overview of robot calibration," *IEEE Journal on Robotics and Automation*, vol. 5, no. 3, pp. 377–385, 1987.

[37] D. Whitney, C. Lozinski, and J. M. Rourke, "Industrial robot forward calibration method and results," *Journal of dynamic systems, measurement, and control*, vol. 108, no. 1, pp. 1–8, 1986.

[38] Dynalog, "Dynacal: Robot calibration system," 2015. [Online]. Available: http://www.dynalog-us.com/dynalogmainsite_030.htm

[39] R. C. DeVlieg, "Robotic manufacturing system with accurate control," Mar. 24 2015, uS Patent 8,989,898.

[40] Creaform, "Creaform," 2002-2016. [Online]. Available: http://www.creaform3d.com/

[41] A. D. Systems, "Moveinspect technology dpa," 2016. [Online]. Available: http://aicon3d.com/products/moveinspect-technology/dpa/at-a-glance.html

[42] J.-F. Larue, D. Brown, and M. Viala, "How optical cmms and 3d scanning will revolutionize the 3d metrology world," in *Integrated Imaging and Vision Techniques for Industrial Inspection*. Springer, 2015, pp. 141–176.

[43] K. B. Atkinson, "Close range photogrammetry and machine vision," 2001.

[44] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011.

[45] C. Obreja, G. Stan, D. Andrioaia, and M. Funaru, "Design of an automatic tool changer system for milling machining centers," in *Applied Mechanics and Materials*, vol. 371. Trans Tech Publ, 2013, pp. 69–73.