



Industrial Robotics Object Calibration System

A Major Qualifying Project

Submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the
Degree of Bachelor of Science

(Date 15/12/2017)

Team Members

Dai, Kezheng	kdai@wpi.edu
Yu, Mingxin	myu2@wpi.edu
Zhang, Jinwei	jzhang8@wpi.edu
Zhu, Yunze	yzhu2@wpi.edu
mqp-cal@wpi.edu	

Project Advisors

Professor Craig B. Putnam
Professor Torbjorn S. Bergstrom
Professor Susan M. Jarvis

This page was intentionally left blank

Abstract

The Industrial Robotics Object Calibration System is designed to use a six-axis industrial robot to calibrate a work object autonomously and efficiently. In addition to the industrial robot, the calibration system consists of one processing module and two measurement tools. The two tools are for calibrating the robots' tool center point (TCP) coordinates and work object coordinates, respectively. If an object is either placed on a processing stand (i.e. an aluminum ingot) or mounted on the robot arm (i.e. a drill bit), its real coordinates can have slight offsets (both linear positions and angular orientations) relative to the coordinates defined in off-line programming (simulation). In that case, the calibration process is necessary to measure and update both tool and the workpiece's coordinates, thus matching the tool's pre-set moving path. This system provides an efficient calibration solution that can achieve TCP calibration in 30 seconds and work object calibration in 10 seconds. The precision of the calibration results is better than 0.2 millimeters.

Table of Contents

Abstract.....	3
Table of Figures.....	7
Table of Tables	10
Chapter I: Introduction.....	11
1.1 Objective.....	11
1.2 Problem Statement	11
1.3 Proposed Solution	13
1.4 Customer Value Proposition	13
Chapter II: Background.....	15
2.1 Industrial Robot.....	15
2.1.1 Overview	15
2.1.2 End Effector.....	16
2.2 Tool Center Point (TCP) Calibration	17
2.2.1 ABB Navigator.....	17
2.2.2 Bull’s Eye calibration of TCP	18
2.2.3 Laser Lab.....	19
2.3 Work Object Calibration	20
2.3.1 ABB Robot Manual Method	20
2.3.2 DynaCal Robot Cell Calibration Solution by Dynalog	20
2.3.3 Emitter Method of Robotic Work Object Calibration.....	21
2.4 Other System Components.....	22
2.4.1 Measurement tool	22
2.4.2 Sensor communication	22
2.4.3 Robot Communication.....	23
2.4.4 Processing Unit.....	24
Chapter III: System Operation.....	26
3.1 Set Up the System	26
3.2 Run the System.....	27
Chapter IV: Technical Documents.....	28
4.1 System Overview	28
4.2 System Design.....	29

4.2.1 Processing Units	29
4.2.1.1 System Requirements	29
4.2.1.2 Processing Unit Selection	29
4.2.2 Sensor	31
4.2.2.1 System Requirements	31
4.2.2.2 Contact and Non-Contact Sensor Analysis.....	32
4.2.2.3 Laser Sensor Analysis.....	34
4.2.3 Communication Design	37
4.2.3.1 TCP/IP communication between Raspberry Pi and ABB controller	39
4.2.4 End Effector Design	44
4.3 Laser Range Sensor Calibration Method Design	46
4.3.1 Measurement Tool Calibration.....	46
4.3.1.1 Overview	46
4.3.1.2 Theoretical Method.....	46
4.3.1.3 Least Square Edge Detection Method.....	52
4.3.1.4 Four-point Algorithm.....	56
4.3.1.5 Source of Error.....	60
4.3.1.6 Program Design	62
4.3.2 Object Measurement and Calibration.....	64
4.3.2.1 Overview.....	64
4.3.2.2 Sensor Detected Point.....	64
4.3.2.3 Theoretical Method.....	65
4.3.2.3 Source of Errors	67
4.3.2.4 Program Design	68
4.3.3 Application: Blisk Calibration.....	69
4.3.3.1 Overview	69
4.3.3.2 Theoretical Method.....	69
4.3.3.3 Least Square Edge Detection Method.....	70
4.3.3.4 Source of Error.....	74
4.4 U-Shape Sensor Calibration Method Design	75
4.4.1 Measurement Tool Calibration.....	75
4.4.1.1 Overview	75
4.4.1.2 Theoretical Method.....	76
4.4.1.3 Measurement Tool Adjustment.....	80
4.4.1.4 Four-point Algorithm.....	82
4.4.1.5 Recovery	82
4.4.1.6 Source of Errors	86
4.4.1.7 Program Design	89

4.4.2 Object Measurement and Calibration	91
4.4.2.1. Overview	91
4.4.2.2 Theoretical Method	92
4.4.2.3 Source of Errors	94
4.4.2.4 Program Design	95
Chapter V: Product Performance Evaluation	97
5.1 Four-point Method Evaluation	97
5.1.1 Overview	97
5.1.2 Error from the calculated TCP	98
5.1.3 Accuracy of the TCP	100
5.1.4 ABB Four-point Method Algorithm Versus Our Algorithm.....	101
5.2 U-Shaped Sensor Calibration Evaluation.....	106
5.2.1 Overview	106
5.2.2 Repeatability of the Calibration Result	106
5.2.3 Precision of the Calibration	108
5.2.4 Duration of Entire Calibration.....	109
5.3 Laser Range Sensor Calibration Evaluation.....	110
5.3.1 Overview	110
5.3.2 Repeatability of the Calibration Result	110
5.3.3 Precision of the Calibration	110
5.3.4 Duration of Entire Calibration.....	110
Chapter VI: Conclusion	112
Appendix I: Nomenclature Glossary.....	113
Appendix II: Project Video Links	115
References	116

Table of Figures

Figure 1. ABB® RobotStudio: An Example of An Offline Programming Interface	12
Figure 2. ABB IRB 1600 Diagram	16
Figure 3. ABB Navigator TCP Calibration Method	17
Figure 4. TCP Calibration of A Spherical Probe Using Bull’s Eye Method	18
Figure 5. Laser Lab Calibration Equipment	19
Figure 6. Defining Work Object Frame	20
Figure 7. Emitter Method of Robotic Work Object Calibration	21
Figure 8. Specification sheet for OPTEX CD33-85N-422 laser range sensor	35
Figure 9. Keyence LV-S71 Laser Sensor Specification Sheet	36
Figure 10. Keyence LV-N11P Laser Sensor Amplifier Specification Sheet.....	37
Figure 11. Communication Design Overview	38
Figure 12. TCP/IP Communication Between Client and Server	39
Figure 13. Windows IPv4 Configuration for Communicating with ABB Controller.....	41
Figure 14. Raspberry Pi Configuration of Static IP Address to Connect with ABB Controller...	41
Figure 15. Python TCP/IP Socket Code for Connecting to ABB Controller As The Client	42
Figure 16. RAPID Code About TCP/IP Socket Used for Server Application.....	43
Figure 17. 3D Model of End Effector	44
Figure 18. Detailed Structure of The End Effector.....	45
Figure 19. The End Effector Reaches One Edge Point and Record The Position	47
Figure 20. The End Effector Reaches Another Two Edge points.....	48
Figure 21. The End Effector Reaches Three More Edge points With Another Height	49
Figure 22. Determine the Vector-Based On Center Points On Two Planes	49
Figure 23. The End Effector Changes the Orientation	50
Figure 24. Tool Positions Can Be Determined In the End	51
Figure 25. The Behavior of the Laser Sensor Scanning the Edge in Three Different Poses	52
Figure 26. Calibrator Design.....	53
Figure 27.The Behavior of Laser Sensor Scanning the Beveled Edge in Three Different Poses.	53
Figure 28. Sensor Reading While Scanning the Calibrator Surface Perpendicular To the Laser Beam	54
Figure 29. Sensor Reading While Scanning the Calibrator Surface That Is Slightly Tilting Towards the Laser Beam	55
Figure 30. Sensor Reading While Scanning the Calibrator Surface That Is Slightly Tilting Away the Laser Beam	55
Figure 31: RB_1400 Welding Robot	56
Figure 32: Example of four ABB variable: <code>robtargets</code> used for 4-point method.....	58

Figure 33. Flowchart of Laser Range Sensor Measurement Tool Calibration	62
Figure 34. Sensor Detected Point Method	65
Figure 35. Theoretical Process Diagram.....	66
Figure 36. The Flowchart of the Three-plane Method Implementation	68
Figure 37. Blisk inner Circle and the Critical Points	70
Figure 38. Representation of Sensor Scanning the Edge of the Blisk	71
Figure 39. The Reading of Sensor while Scanning Edge of Blisk.....	71
Figure 40. Analysis of the Sensor Reading.....	72
Figure 41. Representation of Sensor Scanning Another Edge of the Blisk	73
Figure 42. The Reading of the Sensor while Scanning Another Edge of the Blisk.....	73
Figure 43. The Reading of the Sensor while Scanning Another Edge of the Blisk.....	74
Figure 44. U-shaped Measurement Tool	75
Figure 45. Top View of Objects Moving Alongside Circular Trajectory.....	76
Figure 46. Top View of Captured Eight Critical Points Running One Loop	77
Figure 47. Top View of Four Average Points on Laser Beams.....	77
Figure 48. Flange Plate's Position When Laser Intersects With Tool Point P1	78
Figure 49. Flange Plate's Position When Laser Intersects With Tool Point P2	78
Figure 50. Finding the Tool Center Point	79
Figure 51. Tool Center Point Calculation	80
Figure 52. Laser Beams That Have a Small Offset	80
Figure 53. The Calibrator Follows The Circular Trajectory.....	81
Figure 54. The Calibrator Is Lifted In Z-axis direction	82
Figure 55. Perfect situation In Calibration: Vector 1 & 2 Are Vertical	83
Figure 56. Real Situation: Vector 1 & 2 Are Not Vertical	84
Figure 57. Condition before recovery: TCP might be wrongly recognized	85
Figure 58. Condition after recovery	85
Figure 59. Theoretical Model For Calibrating U-shape Sensor.....	86
Figure 60. Compare Models For Calibrating The U-shaped Sensor When Offsets Occur.....	87
Figure 61. Optimized Model for Calibrating the U-shape Sensor	88
Figure 62. Flowchart of The U-Shape Measurement Tool Calibration	89
Figure 63. Tool That Is Perpendicular to The Laser Beams	93
Figure 64. The Tool Moves Up.....	94
Figure 65. Flowchart of Object Calibration Using U-shaped Sensor	95
Figure 66. The Position Errors from Four-point Method Algorithm.....	98
Figure 67. Flowchart of Error and Accuracy Evaluation.....	101
Figure 68. Normal Distribution Error Added Into Input.....	103
Figure 69. Structure of One Data Set.....	103

Figure 70. Relationship Between TCP Accuracy And Standard Deviation of Input Error in Simulation 104

Figure 71. Relationship Between the Mean Error from the Calculated TCP And Standard Deviation of Input Error in Simulation 105

Figure 72. The Repeatability Distribution of U-Shaped Calibration Result in the Experiment . 107

Figure 73. The Readings of Dial Indicator using U-Shaped Sensor Calibration in the Experiment 109

Table of Tables

Table 1. Table. Basic requirements for range sensor.....	31
Table 2. Basic requirements for laser sensor on U-shaped Measurement Tool.....	32
Table 3. Comparison Table Between Contact and Non-contact Sensors	33
Table 4. Available Communication Port For Each Component in the Calibration System.	38
Table 5. Input Data of Four-point Method.....	102

Chapter I: Introduction

1.1 Objective

The goal of our project is to design and implement a new work object calibration system using a 6-axis industrial robot. The system can provide accurate calibration of wider ranges of workpieces than current solutions.

1.2 Problem Statement

In today's world, manufacturing is one of the main sources pushing modern economic development. An increased demand in the market requires productions to achieve high efficiencies while achieving good qualities. To achieve this goal, using robots to replace human labor seems to be a growing trend because robots can achieve higher efficiencies while costing less than human workers. One good example is the implementation of industrial robotics. Foxconn, an electronics manufacturer providing about 40 percent of the global consumer electronics revenue, has recently claimed that they have replaced 60,000 assembly employees with industrial robot arms (Wakefield, 2016). Industrial robots have been widely introduced in the past few decades. They can do many dull, dirty, or dangerous jobs repeatedly with high efficiency.

Offline programming of industrial robots is widely used in the manufacturing process. The concept of offline programming can be viewed as simulation. Usually, an offline programming software provides a virtual environment with robot's CAD models. Engineers can build a simulated production scenes and program the robot in offline programming interface, shown in Figure 1. Offline programming has two main advantages: The first one is efficiency. The new program can be tested virtually without stopping active manufacturing, thus avoiding loss of system availability to upgrade or maintenance (DELFOI, 2016). The second one is the flexibility of path planning. Offline programming enables program designer to generate more complex trajectory than robot-jogging (online programming).

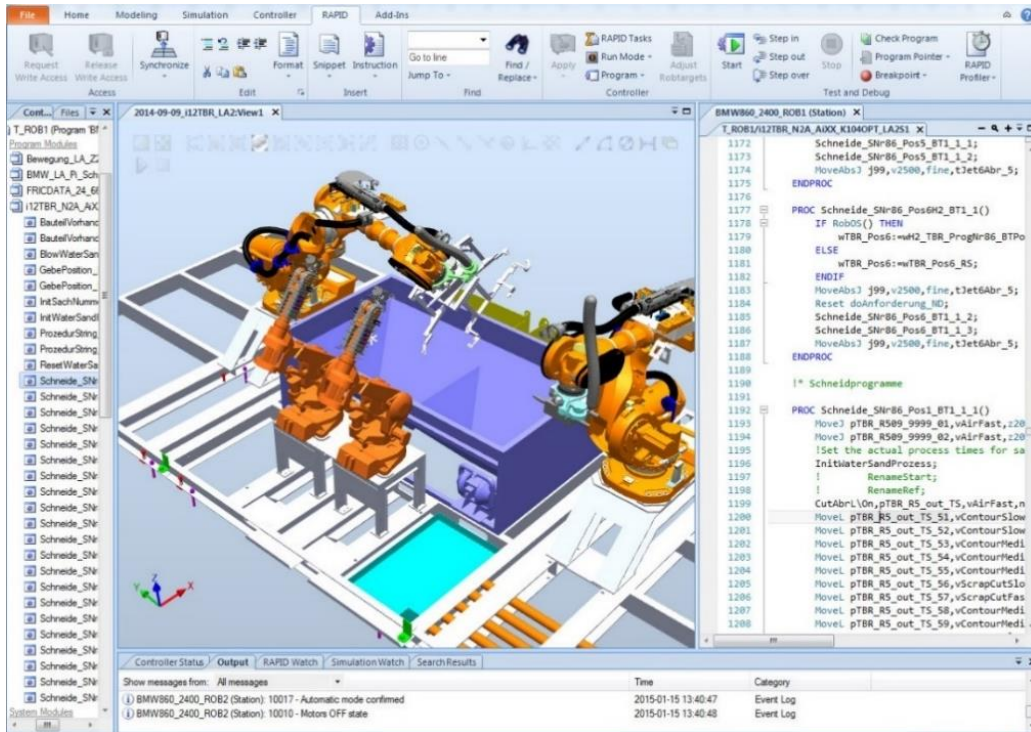


Figure 1. ABB® RobotStudio: An Example of An Offline Programming Interface (Devo Engineering, n.d.)

However, one of the main drawbacks of offline programming is that it cannot respond timely and properly when the outer environment changes. Therefore, it takes time and labor to manually retune or reprogram these robots periodically to keep their performance on mass-production tasks, such as in spot-welding, sealing, arc-welding, assembly and water-jet cutting. An obvious example of the flexibility loss is changing workpiece's position. Every time when one new workpiece is placed onto processing table, small systematic offsets, from either linear or angular position, or both, may occur. If those minor offsets aren't compensated for, they will be amplified, likely causing processing failures. Therefore, each workpiece's position has to be measured and calibrated, then its actual position, in vector form $([x, y, z, \theta_x, \theta_y, \theta_x]^T)$ is updated to the robot controller. This procedure, in the industrial robotics field, is called work object calibration.

Traditionally, work object calibration is implemented by manual measurement, and this method is generally time-consuming and inaccurate. With the development of robot technology, autonomous solutions have recently appeared. There are a few automated work object calibration

methods available on the market. The DynaCal Robot Cell Calibration Method Solution (DynaCal Inc., n.d.) by Dynalog, Inc. and LEONI Advintec Solution (Leoni, n.d.) by LEONI, Inc. are two representative commercial solutions, and both can calibrate workpieces to high accuracy by measuring reference points' positions. However, one of their drawbacks is that those methods are processed by determining the reference points only on an object's flat surfaces. That solution only supports calibrating specific types of the work cells. If some parts of the work object are curved, like turbo blisks, those existed calibration methods won't be effective.

A new work object calibration method that is capable of calibrating workpiece based on detecting a wider set of its characteristics rather than only its flat surfaces It can theoretically calibrate more types of workpieces than current existing methods. If the new method were available, work object calibration would totally become autonomous, which would save a considerable amount of labor doing relative to calibrating manually.

1.3 Proposed Solution

To fulfill the calibration, the arm's end effector can either hold a workpiece to approach a fixed measuring tool or use a measurement tool to approach a fixed workpiece. After measurement, the system uses the calibration algorithm we developed to get the real coordinate system. Finally, the real coordinate system is updated into the off-line programming interface, and the offset is successfully recovered. The precision of the calibration achieved is within 0.2 mm.

1.4 Customer Value Proposition

Traditionally, to implement the work object calibration, the potential customer usually hires experienced technicians to calibrate workpieces manually. The technicians are paid based on their working time or efficiency, and that employment is an enormous expense to the customer. Additionally, the overall efficiency of the manual calibration is not sufficient. The average working time for each calibration process is approximately 20 minutes, and the precision can reach only around 0.4 millimeters (Hao Gu, 2015), which is not acceptable in many industrial fields that require fine processing.

The potential customers can also select one of the automatic calibration products in the market, such as DynaCal or Advintec Solution. Admittedly speaking, those products can reach high efficiencies that vary from 0.02 to 0.5 millimeters, within a short amount of time, from 15 seconds to 20 minutes, depending on the object's size and shape. However, those products can only calibrate objects with flat surfaces. If the customer needs to calibrate objects having mostly curved surfaces, like turbo blisks, those methods cannot be used. Therefore, a new type of calibration system needs to be introduced.

Calibration using our Object Calibration System can achieve very good precision within a short amount of time. The total runtime for each calibration varies from 10 seconds to 30 seconds, and its precision is better than 0.2 millimeters. Since the new calibration method is used to detect certain parts of the object, depending on its characteristics, it can support calibrating workpieces that containing curved surfaces. In addition, the customers can implement the calibration system on a large scale, without additional labor.

If our system were in use, the only consideration from customers would be periodic maintenance. When the system is being used over time, some minor offsets from the measurement tools itself may occur. To keep the best performance in the calibration, the system's measurement tools need to be calibrated. Our system also provides a thorough solution to calibrating the measurement tool. The solution is unified and easy to be manipulated, and a technician can master it through a short-time training. We anticipate that the system maintenance needs to implement only once in every six months, and the tool calibration process lasts five minutes. Overall, our Object Calibration System has a potential to support a wider range of use than the available calibrating solutions. It also has potential to achieve higher efficiency and better precision with less labor and simpler maintenance than traditional manual calibration.

Chapter II: Background

2.1 Industrial Robot

2.1.1 Overview

An Industrial Robot is a robot system using in manufacturing. They are automated, programmable and capable of movement. To achieve agile movements, an industrial robot usually has multiple joints. The number of joints equals the number of links. Most of the industrial robots have six-links because such robots can achieve any poses in the form of $[x, y, z, \theta_x \ \theta_y \ \theta_z]^T$, which contains 6-degrees of freedom, in our three-dimensional world. The flexibility of the robot makes it capable of manipulating many types of objects within the reachable distance. Welding, painting, assembly, picking and placing, packaging and labeling, and palletizing are the typical applications of industrial robots. Such work requires a high endurance, speed, and precision. Industrial robots are capable of performing such tasks well. They are commonly and widely used in all sorts of industrial environments. Once an industrial robot is programmed, it can work automatically and efficiently.

The industrial robot we use for this project is ABB IRB 1600(6kg) industrial robot (ABB Robotics, IRB1600_PR10282EN-I.pdf, 2017), which has 6-kilogram handling capacity, can reach 1.2 meters in radius. The speed of rotation is up to 460 degrees per second and the position repeatability of the robot is 0.02 millimeters.

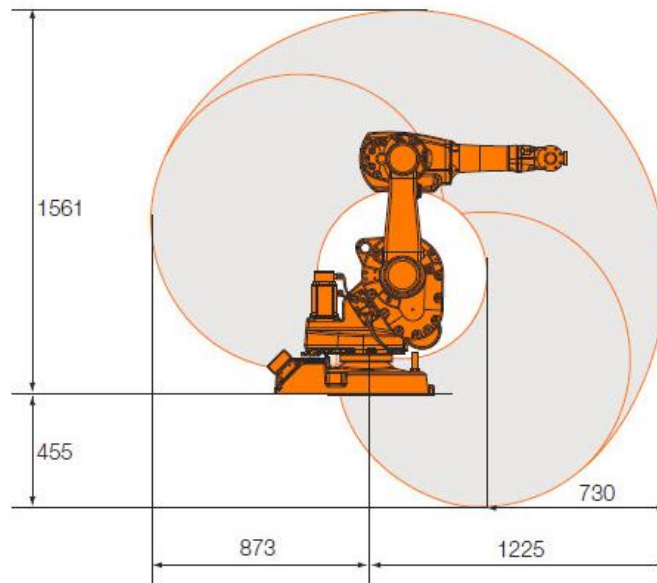


Figure 2. ABB IRB 1600 Diagram (ABB Inc., IRB 1600 Data)

2.1.2 End Effector

The end effector is the device installed on to the end of the industrial robot arm, which is designed to interact with the environment and the robot itself to achieve multiple functions in manufacturing. Typical end effectors, for example, include “welding devices (such as MIG-welding guns, spot-welders, etc.), spray guns and also grinding and deburring devices (such as pneumatic disk or belt grinders, burrs, etc.), and grippers (devices that can grasp an object, usually electromechanical or pneumatic)” (Bonev, 2013). A good end effector is able to hold tools or a workpiece still. In other words, there are many ways to design the end effector, as long as the design can achieve the goal and meet the requirement of the project.

Since the end effectors have various sizes, precisely locating the center point of the end is critically important. The process of locating the center point of a tool is called tool center point (TCP) calibration. In addition to for measuring its Cartesian coordinates manually, industrial robots generally have an internal program to automatically determine the tool center point. The detailed background information of TCP calibration will be discussed in the next section.

2.2 Tool Center Point (TCP) Calibration

2.2.1 ABB Navigator

The ABB Navigator is one calibration method that is automated and accurate to find the TCP, as shown in Figure 3. Instead of letting the user manually point out positions, the robot is equipped with a spherical probe tool and the robot cell is prepared with mounting holes on the fixture for spherical objects. The calibration is performed by letting the robot locate the spherical objects on the installation. The sensor mechanism is tactile, i.e., the touch between objects is determined and causes the robot to stop. The solid sphere is connected to ground, and a voltage is applied to the spherical probe. The physical touch is detected when an electrical circuit is established. The electrical circuit is established when two spherical objects are in contact. The I/O value is changed, and the current coordinate system is stored.

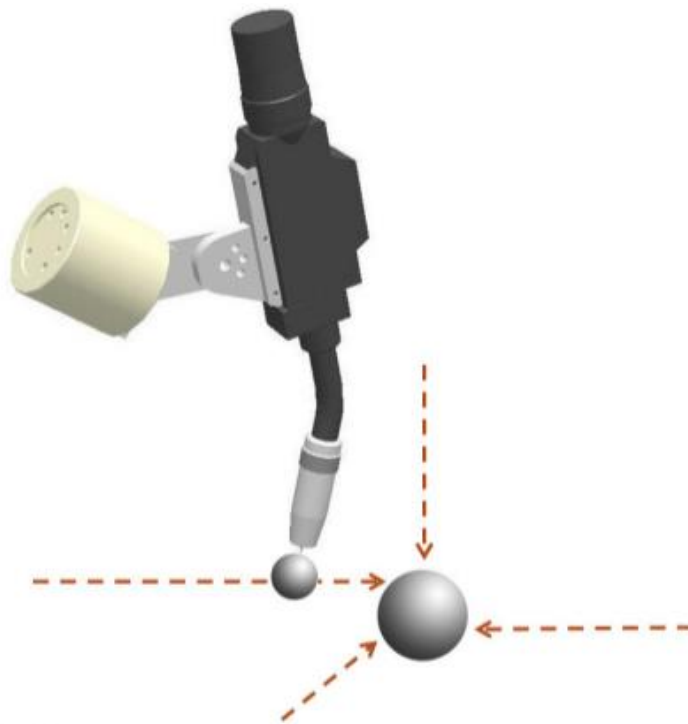


Figure 3. ABB Navigator TCP Calibration Method (Bergstrom, 2011)

2.2.2 Bull's Eye calibration of TCP

The Bull's Eye method can determine the TCP and orientation of a tool, as shown in Figure 4. The TCP is the coordinate system of the end of the robot tooling. A major limitation of this method is that it is only applicable to cylindrically shaped tools.

Bull's Eye uses the laser technology to calibrate. By moving the robot's tool through the laser beam, the physical width of the concentric portions of the tool can be determined given that the nominal geometry of the tool is known. The centerline can be calculated. Based on the determined center line, the tool orientation is set. Finally, the TCP is determined.

The Bull's Eye method can determine the centerline and the end of the tool, but the method cannot find the center point of a probe sphere. However, it will determine the X, Y centerline and the end of the tool in Z-axis. In other words, the TCP can be mathematically determined if the radius of the probe is known.

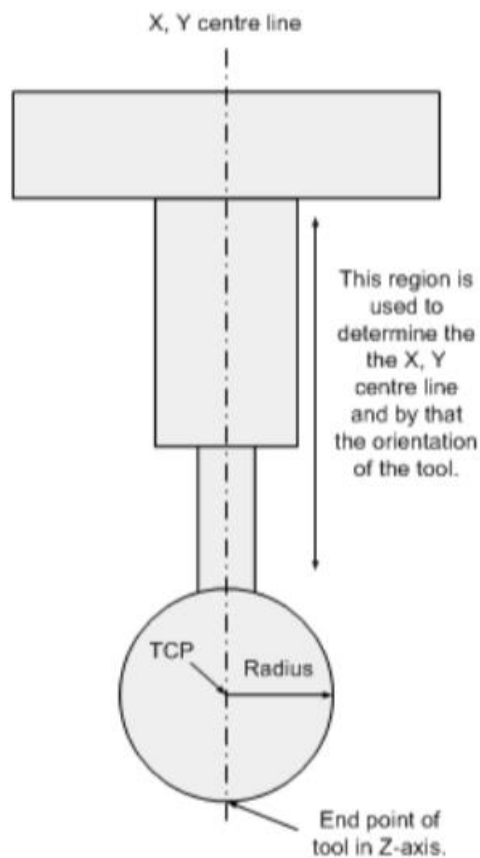


Figure 4. TCP Calibration of A Spherical Probe Using Bull's Eye Method (Bergstrom, 2011)

2.2.3 Laser Lab

Laser Lab is a laser-based calibration technology, as shown in Figure 5. The method consists of a measuring device called “Laser Lab” and a measuring sphere. The Laser Lab measuring device consists of five individual laser sensors. The laser sensors are positioned in a pentagon in the device and are aligned so that the five laser rays will intersect at one common point. By positioning the sphere in the laser device, positions on the surface of the sphere can be determined in three dimensions. This is obtained by measuring the distances from each one of the five sensors. By obtaining surface points, the position of the sphere can be determined. (Bergstrom, 2011)

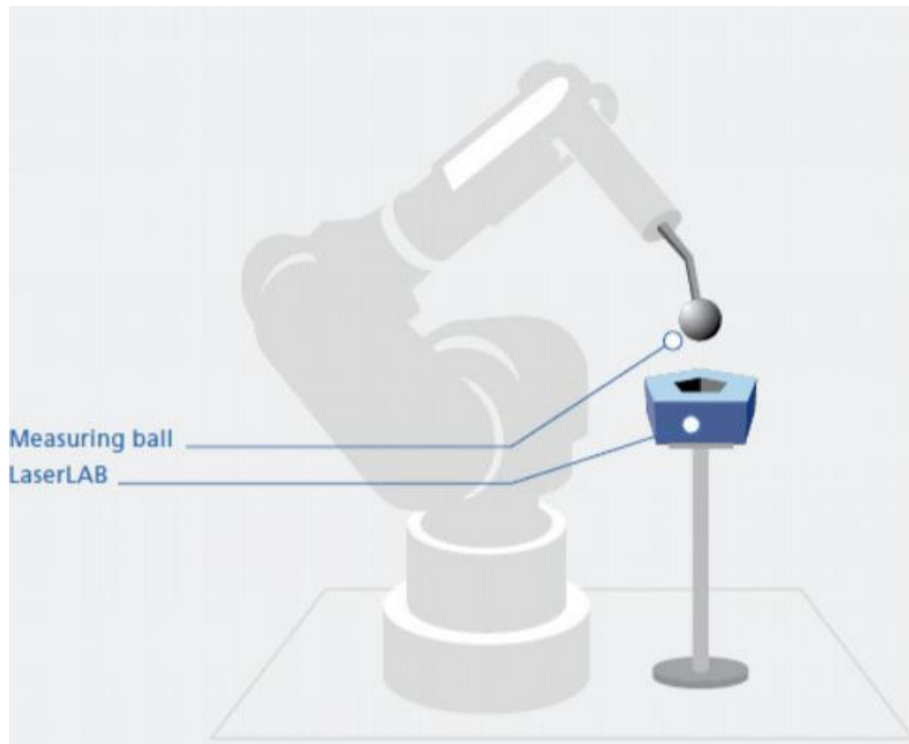


Figure 5. Laser Lab Calibration Equipment (Bergstrom, 2011)

2.3 Work Object Calibration

2.3.1 ABB Robot Manual Method

In the ABB Robot manual, it has a built-in function that can define a work object by measuring positions of three points which two on the x-axis and one on the y-axis, shown in Figure 6. This method is easy to use because it is build-in the ABB robot. However, this method can only calibrate the most basic cubic work objects. It also requires human eye-balling and hands jogging the robot arm. It is slow and has an error of 0.5 millimeter, which is large compared with the results of other methods. (ABB Robotics, Operating Manual RobotStudio, 2007)

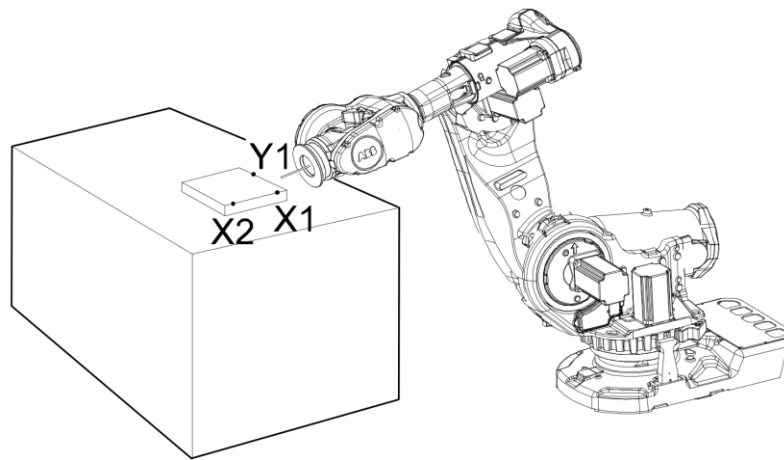


Figure 6. Defining Work Object Frame (ABB Robotics, Operating Manual RobotStudio, 2007)

2.3.2 DynaCal Robot Cell Calibration Solution by Dynalog

The DynaCal Robot Cell Calibration system (DynaCal Inc., n.d.) is the calibration system produced by Dynalog. It can eliminate discrepancies in robot installation between the ideal design and the actual position of the workpiece. This method also uses a laser tracker. The laser sensor is placed on several points of the working object manually. By comparing the theoretical and actual position of the object, a set of corrections to the transformation parameters from the robot axes to the servo controller can be determined. The accuracy of the method is about 0.2 to

0.3 mm. The advantage of this idea is that the operator is independent, which means the operator could move around all the station. The disadvantages of this idea are it requires additional equipment. Additionally, it is relatively expensive and needs significant setup phase.

2.3.3 Emitter Method of Robotic Work Object Calibration

This robotic work object calibration method includes a work object or an emitter, as shown in Figure 7. Initially, the work object is placed in a selected position on a fixture or workpiece on the shop floor. The work object emits a pair of beam-projecting lasers which intersect at a tool contact point and act as a crosshair. The robot tool is manipulated into the tool contact point. The work object emits four plane-projecting lasers which are used to adjust the roll, yaw, and pitch of the robot room relative to the tool contact point. The robotic work object calibration method increases the accuracy of the off-line programming and decreases robot teaching time. Compared to other methods, this method is much quicker to calibrate and higher accuracy. However, the major constraint is the emitter must be mounted to the work object to make it work. (US Patent No. US20120265341, 2012)

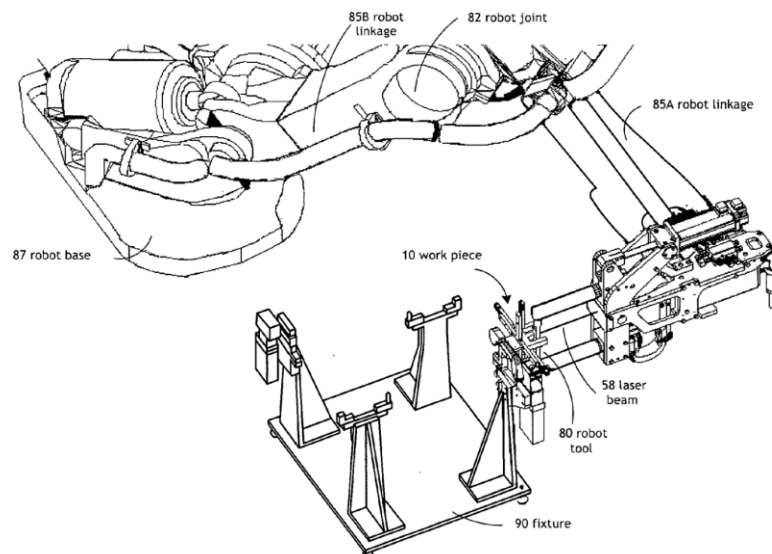


Figure 7. Emitter Method of Robotic Work Object Calibration (US Patent No. US20120265341, 2012)

2.4 Other System Components

2.4.1 Measurement tool

Measurement tools are used mainly in robot manufacturing. We will discuss several most used sensors in the robotic industry and their performance.

Camera, an image sensor, can detect an object and transfer the object into image information. It can detect the different wavelength of the light and transfer that information into analog or digital signals.

The probe is also known as contact displacement sensor. It can measure the distance to a target with high precision and high durability. The contact strikes the target and moves up and down according to fluctuations in the target's surface.

Laser sensor can detect the presence of an object. This is different from laser range sensor. This type of sensor cannot tell the distance between the target and the sensor. It can only tell if the target is present or not.

Laser range sensor is also known as laser displacement sensor. It can measure the distance and position with high speed, accuracy, and precision. When the transmitter shoots the laser at the target, the receiver receives the reflected laser beam.

2.4.2 Sensor communication

There are several possible ports we are going to use in the sensor communication, which include the serial (i.e COM1) port, Ethernet, analog inputs and digital (GPIO) inputs. The communication method is highly depending on the sensor we choose. For example, the communication port of our laser range sensor is the COM1 port, and the output port of our laser sensor is digital inputs.

2.4.3 Robot Communication

Robot communication is very important in our project because the industrial robot needs to communicate with both the sensors and the processing units. In this section, it is going to introduce several communication methods used in the industrial robot: TCP/IP socket, DeviceNet and PC SDK by ABB. It is important to understand them because each of them has their own advantages.

One of the high-level communications is ABB's PC Software Development Kit (PC SDK). It is a software tool, which enables programmers to develop customized operator interfaces for the robot controller. These applications should be programmed by NET programming languages C# or Visual Basic.NET. Such custom applications can be realized as independent PC applications, which communicate with the robot controller over a network (ABB Inc., ABB-Developer Center, 2016). The advantage of using ABB PC SDK is that there are many useful ABB internal functions or libraries that can be called in C# programming. ABB controller does not require any additional programming. For example, we can design a program in PC to display the robot's real-time coordinates over the network. The disadvantage of using ABB PC SDK is that it is only compatible with Windows system, not Linux.

Another high-level communication is DeviceNet. DeviceNet is a Fieldbus standard used in automation technology. DeviceNet uses the Controller Area Network as its underlying communication protocol, with application- particular profiles for various devices (Controller Area Network Sdn Bhd, 2013). ABB controller has many supported DeviceNet modules. Most famous one is DSQC 652 I/O Unit. The advantage of using DeviceNet is that it is simple and stable to connect digital signal. For example, the laser sensor can output high or low signal to ABB controller through DeviceNet. The drawback is obvious that it cannot support the communication of complex data. For example, some floating points like calibration results are hard to transmit to ABB controller through DeviceNet.

In some cases, both the DeviceNet and the ABB's PC Software Development Kit are not available, such as running on the Linux system. Therefore, we can use the communication via a TCP/IP socket by ourselves. We can implement the TCP/IP socket using C#, Python or the other supporting programming languages.

2.4.4 Processing Unit

To choose which types of processing units can be properly used in our calibration system we did some research about the following types.

One choice is to use PC. Personal Computer (PC) is one of the strongest types of processing unit that can be accessed. It has multiple types of I/O that support a wide range of communication protocols. To use one PC, an operating system(OS) is necessary. Its main job is to dynamically assign hardware and software resources to programs. Different OS has various designing logic. The most widely-used OS, Windows, is a typical closed-source OS, and its resource assignment logic can be accessed by the system only; the normal users or administrator don't have access the modify it. Alternatively, another OS, Linux, is open-sourced. The owner of the Linux can be the root user, which have ultimate access to the operating system, including assigning a certain amount of computing resources to specific applications. Generally, on the hardware, PC is quite powerful and compatible. In most cases, the system response to interrupts are more stable on Linux rather than on Windows. (Stallings, 2005).

Another choice, Embedded system, is also a computing system, but less powerful than PC. With the development of chip manufacture, many new types of embedded system are capable of handling more complex calculation, even running an operating system with many applications on the top level. One of the typical examples is Raspberry Pi. It supports running Linux and Windows IoT currently and contains main-stream interfaces: USB 2.0, 3.5 mm audio jack, HDMI, Ethernet, and GPIO etc. With the trend of Internet of Things (IoT), more and more smart devices are implemented with embedded systems like Raspberry Pi.

One last choice is Field-programmable gate array (FPGA). Comparing with PC and Embedded system, FPGAs are less well-known. As its name shows, FPGA achieves a certain type of logic processing by modifying its internal logic gates. Since every step of processing is based on register-transfer level (RTL) design, it usually acts as a coprocessor, running alongside with another embedded system. One of the main advantages of FPGA is efficiencies. According to the white paper introducing FPGA from National Instrument website: "FPGA exceeds the computing power of digital processors by breaking the paradigm of sequential execution and

accomplishing more per clock cycle” (National Instruments Inc., 2012). Therefore, the FPGA is widely used in the industries where need fast processing.

Chapter III: System Operation

In this chapter, we describe the operation of the system in various use cases. Specifically, we discuss how to set up the system and how to run the system.

3.1 Set Up the System

Setting up the calibration system requires five steps.

1. The first step is to understand the purpose of the system, which is designed to perform object coordinates calibration in the field of industrial robotics. Then the system provides calibrating the coordinates of two types of objects: coaxial and symmetric type of objects and regular-shaped type of objects. Therefore, the users should know whether their objects belong to either of those types and which type the objects belongs to. For example, the aluminum ingot is a typical regular-shaped object, and drill bits or a welding torch is the typical coaxial and symmetric objects.

2. The second step is to match their certain types of objects with the corresponding measurement tool. Calibrating coaxial and symmetric objects should use U-shaped measurement tool, and calibrating regular-shaped objects should use the laser range sensor measurement tool.

3. The third step is to determine the object's mounted position and the measurement tool's mounted position. If the object is mounted on the robot's end effector, then its corresponding measurement tool should be mounted in a fixed position beside the robot; if the object is placed in a fixed position beside the robot, then its corresponding measurement tool should be mounted on the robot's end effector. Take the example of aluminum ingot and drill bits again. Usually, the aluminum ingot should be placed on a stand. In order to calibrate its coordinates, the measurement tool, the laser range sensor measurement tool should be mounted on the end effector. A drill bit is usually mounted on the robot's end effector, so its corresponding measurement tool, U-shaped measurement tool, should be mounted on a certain fixed position beside the robot; but if users are going to calibrate the coordinates of a drill bit that is fixed on to a working stand, the U-shaped measurement tool should be mounted on the end effector instead.

4. The fourth step is to adjust the measurement tool. This process helps the measurement tool to achieve high accuracy in later calibration. The adjustment procedures of the laser range sensor measurement tool and U-shaped measurement tool are shown in section 4.3.1 Measurement Tool Calibration and section 4.4.1 Measurement Tool Calibration, respectively.

5. The last step is to finalize the setup. If the measurement tool used is U-shaped measurement tool, then the users should switch the U-shaped calibrator (mentioned in the corresponding measurement tool adjustment section) to the object. If the measurement tool used is laser sensor measurement tool, then the users should jog the robot and select measured points on a sample object. Those points' positions will be recorded in the calibration program. The details about point selections are shown in Section 4.3.2.

After properly going through the five steps, the users have successfully set up the calibration system.

Note that the user needs to set up only once unless the position of fixed measurement tool's position is changed.

3.2 Run the System

The procedure of running the calibration system is simple. The users can just run the corresponding program based on the calibration's properties (object type and object's position mentioned in section 3.1 Set Up the System). The detailed information about object calibration can be referred in the Section 4.3.2 and Section 4.4.2.

Chapter IV: Technical Documents

4.1 System Overview

This section is to briefly introduce the main procedures of the calibration system.

We introduce two calibration methods for two types of workpieces. In order to accurately compute their real coordinates, we have to calibrate the measurement tool first: 1) To determine the accurate relative position between laser range sensor measurement tool (LRSMT) and flange plate. 2) To determine the accurate position between the laser beam's intersection and robot base. The details of achieving those goals are discussed in 4.3.1 Measurement Tool Calibration and 4.4.1 Measurement Tool Calibration, respectively. Those two goals are for calibrating the measurement tool, which is a prerequisite for doing the work object coordinates calibration.

After those subordinate goals are achieved, the two calibration methods can work efficiently and accurately. The first method, called Laser Range Sensor Calibration Method, can calibrate the coordinates of the regular-shaped objects, such as aluminum ingots. The method's details can be found in Section 4.3.2. The second method, called U-shape Sensor Calibration Method, is able to calibrate the coordinates of coaxial and symmetric objects. Those types of objects are commonly seen in many tools, such as welding guns or drill bits. The method's details can be found in section 4.4.2 Object Measurement and Calibration.

4.2 System Design

4.2.1 Processing Units

4.2.1.1 System Requirements

One crucial procedure of calibration algorithm is to capture the instant position using corresponding measurement tool. To achieve the purpose of highly accurate measurement, the system should be responsive and stable. Therefore, we also need to minimize the delay caused by the system communication. Besides making efforts on software optimization later, the hardware selection is also important.

To achieve the requirements of the accuracy and stability, we define the two purposes of the system: 1) Time duration from sensor's detection to processor's computation is no more than 10 milliseconds; 2) The variations of the duration are no more than ± 3 milliseconds. If the system can achieve either one purpose or both, that means the system is qualified to run accurately. Therefore, we did research on the processing unit selection.

4.2.1.2 Processing Unit Selection

To reach the requirements above, we need a processing unit that is able to communicate with sensors and robot controller responsively and to process continuous data. After the research, we found out the processors listed below are possibly suitable: they are easy to access, have a low budget, and contain user-friendly development environment. The detailed evaluations for each processor are also shown below.

a) FPGA Dev Board

The manifest advantage of FPGA is fast because using logic gates is the perfect basis to reach qualified stability and processing speed. However, since the architecture and the development environment are too low-level, it became difficult to build various of communication interfaces based on logic gates. Therefore, FPGA is not friendly to use.

b) PC (Windows OS)

The main advantage of PC processing is the availability of the RobotStudio SDK, officially provided by ABB Inc. It can be programmed using C# in Visual Studio. Therefore, the development environment is user-friendly, and the compatibility is also reliable. But after our

testing, the system response time, however, cannot be constrained within ± 3 milliseconds; its variations are ± 10 milliseconds, relying on the load on the computer. After research, we figured out the potential stability issues are an inherent attribution of Windows OS.

c) Raspberry Pi (Linux)

Raspbian OS provides a user-friendly Python development environment, and the Linux core can be accessed and customized from the resource distribution. After we properly set up and ran test code, the system response time changed from 6ms to 8ms; its relatively fast response and good stability finally reached our requirements.

After doing these comparisons, we found out that the Raspberry Pi is the most suitable processing device among those three. Therefore, we decided to use Raspberry Pi in the end.

4.2.2 Sensor

4.2.2.1 System Requirements

a) Basic requirements for range sensor

Knowing that we would use range sensor for our project, the next thing was to find the proper sensor. We estimated and determined the necessary specifications for the ideal sensor we are looking for.

Table 1. Table. Basic requirements for range sensor

Categories	Requirement	Reasoning
Accuracy	Maximum 0.02 millimeters	Our project goal is set to have 0.2 millimeters of accuracy. The laser range sensor ought to have better of precision.
Response Time	Maximum 10 milliseconds	The range sensor will record while in motion. Minimum response time will give the best results.
Detecting Angle	Minimum 45 degrees	Without knowing the rotational angle of the detecting surface, the sensor needs to be prepared for the worst possible situation.
Measuring Range	Minimum 50 millimeters	Although this project does not need a long measuring range, a minimum measuring range is required.
Price	Limited to 1000 USD	For this Major Qualifying Project, we have a limited budget of 800 dollars. Although we are willing to pay our own money into it, it is best for keeping it under 1000 USD.

b) Basic requirements for laser sensor on U-shaped Measurement Tool

Requirements for the laser sensor are not as strict as selecting the range sensor. The most important thing we care about is the response time of the laser sensor.

Table 2. Basic requirements for laser sensor on U-shaped Measurement Tool

Categories	Requirement	Reasoning
Response Time	Maximum 10 milliseconds	The laser sensor will record while in motion. Minimum response time will give best results.
Measuring Range	Around 100 millimeters	The design of the U-shaped Measurement Tool sets the measuring range.
Price	Limited to 1000 USD	For this Major Qualifying Project, we have a limited budget of 800 dollars. Although we are willing to pay our own money into it, it is best for keeping it under 1000 USD.

4.2.2.2 Contact and Non-Contact Sensor Analysis

Based on our algorithm, we need linear displacement sensor. There are two major types of displacement sensors which are contact displacement sensors and non-contact displacement sensors.

The contact displacement sensors are known as the probe. The probe strikes the target and moves up and down according to fluctuations in the targets' surfaces. The amount of movement is detected using the sensor's internal scale to measure the target's amount of displacement.

The principle of the laser displacement sensor is triangulation. When the target's position fluctuates, the position of the received laser on the laser-receiving element moves. The target's amount of displacement is measured by detecting the amount of movement of the received laser.

Both contact and non-contact sensors are suitable for our project, so we did a thorough research for both types. The table below shows the comparison between the two types of sensors.

Table 3. Comparison Table Between Contact and Non-contact Sensors (Keyence Inc., 2016)

Comparison table		
	Contact Sensor	Non-contact Sensor
Measurement Stability		
Influence of Surface Roughness	Recommended The sensor is not affected by surface roughness.	Not Recommended Measured values fluctuate due to the surface roughness.
Target Resistance	Avoid Soft targets cannot be measured.	Recommended Soft targets and liquids can be measured.
Environmental Resistance	Recommended Stable measurements are possible even if water is on the target.	Not Recommended Measurement errors occur if water is on the target.
Application Responsiveness		
Measurement Speed	Avoid The sampling speed is not as fast as that of the non-contact type.	Recommended The fast sampling speed also makes it possible to measure vibration.
Profile Measurement	Not Recommended The large contact is ill-suited to perform microscopic profile measurements.	Recommended The laser spot diameter is small, which enables microscopic profile measurements.
Measurement Range	Not Recommended The measurement range is comparatively short.	Recommended The measurement range is wide.

Cost		
Product Cost	Recommended The sensor is comparatively inexpensive.	Not Recommended High-accuracy models are more expensive than contact types.
Tooling Changes	Avoid Tooling changes take time.	Recommended The work spent to perform tooling changes can be reduced.
Inspection Cycle Time	Avoid The cycle time is not fast.	Recommended The cycle time can be improved.
Risks		
Impact with the Target	Not Recommended There is the risk of targets striking and damage the sensor.	Recommended There is no risk of targets striking the sensor.
Scratching	Not Recommended There is the risk of targets being scratched.	Recommended There is no risk of targets being scratched.

In our research, the laser sensor is better than the probe. The laser sensor has faster response speed, longer measuring range and no risk of making impacts on the target. The laser sensor with high accuracy is also mostly more expensive than the probe. After a careful consideration, we choose to use laser displacement sensor for our project.

4.2.2.3 Laser Sensor Analysis

a) Choice of the range sensor: OPTEX CD33-85N-422

The sensor we chose for our project is OPTEX CD33-85N-422 laser range sensor. It met all the requirements listed in 4.2.2.1 System Requirements. Figure 8 is a screenshot of the

specification sheet for this sensor. Although the detecting angle is not listed in the specification sheet, it's detecting angle is around 60 degrees based on our tests. This sensor cost 800 USD which was within our budget.

Given better sensor, we would have a better result. We chose this sensor because it met our basic requirements.

Specifications

Diffuse-reflective type Measurement distance based specifications

Model	CD33-30□□□	CD33-50□□□	CD33-85□□□	CD33-120□□□	CD33-250□□□
Center of measurement range	30 mm	50 mm	85 mm	120 mm	250 mm
Measurement range	±4 mm	±10 mm	±20 mm	±60 mm	±150 mm
F.S. (full scale)	8 mm	20 mm	40 mm	120 mm	300 mm
Light source	Red semiconductor laser, wavelength: 655 nm, Maximum output: 1 mW				
Laser class	IEC/JIS: CLASS 2 FDA: CLASS II				
Spot size ¹⁾	0.1 × 0.1 mm	0.5 × 1.0 mm	0.75 × 1.25 mm	1.0 × 1.5 mm	1.75 × 3.5 mm
Linearity	±0.1% F.S.				
Repeat accuracy	2 μm (4 μm when response time is set to FAST)	5 μm (8 μm when response time is set to FAST)	10 μm (15 μm when response time is set to FAST)	30 μm (45 μm when response time is set to FAST)	75 μm (100 μm when response time is set to FAST)
Sampling period	0.5 ms/1 ms/1.5 ms/2 ms				0.75 ms/1 ms/1.5 ms/2 ms
Response time ²⁾ Averaging	Fast	5 ms or less: Averaging 1 time (1 ms) + sensitivity switching time (Max. 4 ms)			7.5 ms or less: Averaging 1 time (1.5 ms) + sensitivity switching time (Max. 6 ms)
	Standard	12.5 ms or less: Averaging 16 times (8.5 ms) + sensitivity switching time (Max. 4 ms)			19 ms or less: Averaging 16 times (13 ms) + sensitivity switching time (Max. 6 ms)
	High-resolution	36.5 ms or less: Averaging 64 times (32.5 ms) + sensitivity switching time (Max. 4 ms)			55 ms or less: Averaging 64 times (49 ms) + sensitivity switching time (Max. 6 ms)
Temperature drift	±0.08%/°C F.S.				
Indicators	Distance indicator	LED bar display on operation surface (25-step)			
	Output indicator	Q1 and Q2 LED lights up during output (orange)			
	Input indicator	MF LED lights up during input (orange)			
MF (multi-function) input	Choose from laser OFF, teaching ³⁾ , sample & hold Response time: 3 ms or less				
Connection type	Cable type: Cable length: 2 m (ø5) Connector type: M12, 8-pin				
Protection circuit	Reverse connection protection, overcurrent protection function				
Degree of protection	IP67				
Ambient temperature/humidity	-10 to +45°C / 35 to 85% RH (no freezing or condensation)				
Ambient illuminance	Sunlight: 10,000 lx Incandescent lamp: 3,000 lx				
Vibration resistance	10 to 55 Hz; double amplitude 1.5 mm; 2 hours in each of the X, Y, and Z directions				
Shock resistance	Approx. 50 G (500 m/s ²), 3 times in each of the X, Y, and Z directions				
Warm-up time	Approx. 15 minutes				
Material	Housing: PBT, Front cover: PMMA, Cable: PVC				
Weight without cable	Approx. 65 g				

Diffuse-reflective type Output based specifications

Type	Analog current output type	Analog voltage output type	RS-422 type
Model	NPN type	CD33-□□NA	CD33-□□N-422
	PNP type	CD33-□□PA	CD33-□□P-422
Supply voltage	12 to 24 VDC, ±10/-5%		12 to 24 VDC, ±10/-5%
Current consumption	Max. 85 mA (including analog output)		Max. 55 mA
Control output	Output channel No.	2ch: Q1, Q2 (default setting of self-diagnosis output for Q2) 1ch: Q2 (default setting of self-diagnosis output)	
	Output method	NPN/PNP open collector output, Max. 100 mA / 30 VDC, residual voltage 1.8 V	
Analog output/serial interface	4 to 20 mA, load impedance: 300 Ω or less	0 to 10 V, output impedance: 100 Ω	RS-422 9.6 k to 256 kbps
Applicable regulations	EMC directive (2004/108/EC) / FDA regulations (21 CFR 1040.10)		
Applicable standards	EN 60947-5-7		EN 60947-5-2

Figure 8. Specification sheet for OPTEX CD33-85N-422 laser range sensor

b) Choice of the laser sensor and its amplifier.

The model we had for the laser sensor was the Keyence LV-S71 laser sensor and its amplifier LV-N11P. It met all the requirements listed in 4.2.2.1 System Requirements. The reason we chose this sensor was mainly that we had this sensor, so we won't have to purchase another. The and are the screenshots of specification sheet for LV-S71 and LV-N11P.

SPECIFICATIONS

Model		LV-S71
Type		Small standard
FDA (CDRH) Part 1040.10		Class 1 Laser Product
IEC 60825-1		
Light source		Visible red semiconductor laser, Wavelength: 655 nm
Detecting distance	MEGA	500 mm 19.69"
	ULTRA	
	SUPER	
	TURBO	
	FINE	
	HSP	
Environmental resistance	Ambient temperature	-10 to +50 °C 14 to 122 °F (No freezing)
Material	Case	Metal part: Stainless steel, Plastic part: Polyarylate
	Lens cover	Transmitter: Norbornene plastic Receiver: Polyarylate
Weight		Approx. 70 g

Figure 9. Keyence LV-S71 Laser Sensor Specification Sheet

SPECIFICATIONS

Model		LV-N11P
Type		2 output
Output		PNP
Cable/connector		Cable
Main/Expansion unit		Main unit
I/O	Control outputs	2 output
	External input	1 input
	Monitor output	None
Response time		80 μs (HIGH SPEED)/250 μs (FINE)/500 μs (TURBO)/1 ms (SUPER)/4 ms (ULTRA)/16 ms (MEGA) ^{†1}
Output selection		LIGHT-ON/DARK-ON (switch-selectable)
Timer function		Timer OFF/OFF-delay timer/ON-delay timer/One-shot timer, Timer duration selectable: 1 ms to 9,999 ms, Maximum error against the setting value: ±10% max.
Control outputs		PNP open collector 30 V, Residual voltage 1.2 V or less (Output current: 10 mA or less) / 2.2 V or less (Output current: 10 to 100 mA) (Stand-alone) 1 output max: 100 mA or less, 2 output total: 100 mA or less (Multiple connections) 1 output max: 20 mA or less
Monitor output		-
External input		Input time 2 ms (ON)/20 ms (OFF) or more ^{†2}
Multiple connections to expansion units		Up to 17 units can be connected in total (two-output type is treated as two units)
Protection circuit		Reverse polarity protection, Over-current protection, Surge absorber
Number of interference prevention units		Connected to other than LV-S31: 0 for HIGH SPEED; 2 for FINE/TURBO/SUPER; 4 for ULTRA/MEGA, Connected to LV-S31: 2 for FINE; 4 for TURBO/SUPER/ULTRA/MEGA ^{†3}
Case size		H 32.6 mm 1.28" × W 9.8 mm 0.39" × L 78.7 mm 3.1"
Rating	Power voltage	24 VDC (operating voltage 10 to 30 VDC (with ripple)), Ripple (P-P) 10 % or less, Class 2 or LPS ^{†4}
	Power consumption	Normal: 950 mW or less (at 30 V, 33 mA at 24 V, 60 mA or less at 12 V) ^{†5} Eco on mode: 815 mW or less (at 30 V, 29 mA at 24 V, 52 mA or less at 12 V) ^{†5} Eco Full mode: 650 mW or less (at 30 V, 24 mA at 24 V, 40 mA or less at 12 V) ^{†7}
Environmental resistance	Ambient temperature	-20 to +55 °C -4 to 131 °F (No freezing) ^{†8}
	Relative humidity	35 to 85 % RH (No condensation)
	Vibration resistance	10 to 55 Hz, Double amplitude 1.5 mm 0.06", 2 hours in each of the X, Y, and Z directions
	Shock resistance	500 m/s ² , 3 times in each of the X, Y, and Z directions
Material	Cable	PVC
	Case	Main unit and cover material: Polycarbonate
Weight		Approx. 75 g
^{†1} 80 μs cannot be selected when the LV-S31/S62/S63 is connected ^{†2} Input time is 25 ms (ON)/25 ms (OFF) when external calibration time is selected. ^{†3} These numbers double when "DOUBLE" is selected. ^{†4} Use with the over current protection device which is rated 30 V or more and not more than 1 A. ^{†5} To connect more than 9 units, the power voltage must be 20 V or more. ^{†6} Increases 30 mW (1 mA) for HIGH SPEED mode. ^{†7} It increases by 15% when connected to the LV-NH100/NH110/NH300. It does not include the power consumption of the load. Power consumption when expansion units are connected is the total power consumption of each amplifier unit. Example: When one main unit (LV-N11N) is connected to 2 expansion units (LV-N12N) and they are used with LV-NH100 heads in HIGH SPEED mode. $(1.15 \times 860 \text{ mW} \times 1) + (1.15 \times 860 \text{ mW} \times 2) = 2967 \text{ mW max.}$ ^{†8} If more than one unit is used together, the ambient temperature varies with the conditions below. Mount the units on the DIN rail with mounting brackets and check that the output current is 20 mA or less for a unit. One or two more units connected: -20 to +55 °C -4 to 131 °F; 3 to 10 more units connected: -20 to +50 °C -4 to 122 °F; 11 to 16 more units connected: -20 to +45 °C -4 to 113 °F. When using 2-outputs, one unit is counted as two units.		

Figure 10. Keyence LV-N11P Laser Sensor Amplifier Specification Sheet.

4.2.3 Communication Design

In our calibration system, one of the most fundamental functionality was to let sensors, the ABB controller, and the processing unit communicate with each other. After researching on each component, we analyzed every object we wanted to communicate and found their supported communication methods as shown in Table 4.

Table 4. Available Communication Port For Each Component in the Calibration System.

	Serial Port	I/O Interrupt (Digital)	Ethernet (TCP/IP)	Analog
Laser Displacement Sensor (only output ¹)	Yes	No	No	No
U-shaped calibration tool (only output)	No	Yes	No	No
Raspberry Pi	Yes	Yes	Yes	Yes
ABB controller	Too expensive ² .	Yes	Yes	No

Base on the table above, we designed our communication logic as shown in Figure 11.

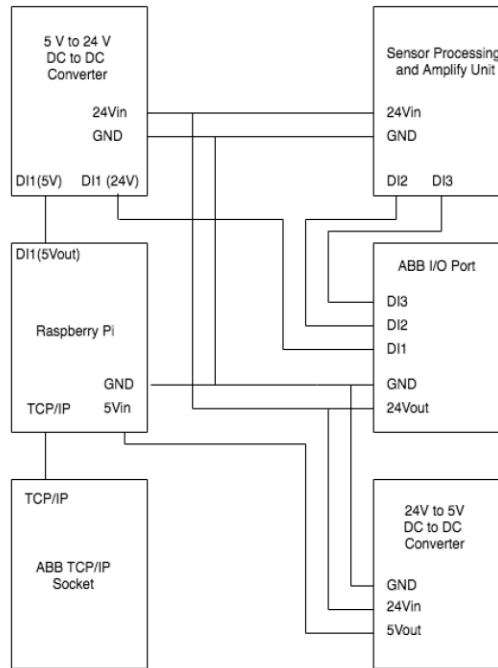


Figure 11. Communication Design Overview

¹ Both sensor only have and need output connection port.

² ABB do support COM1 port, but it needs an adapter that cost 800 dollars in order to work.

4.2.3.1 TCP/IP communication between Raspberry Pi and ABB controller

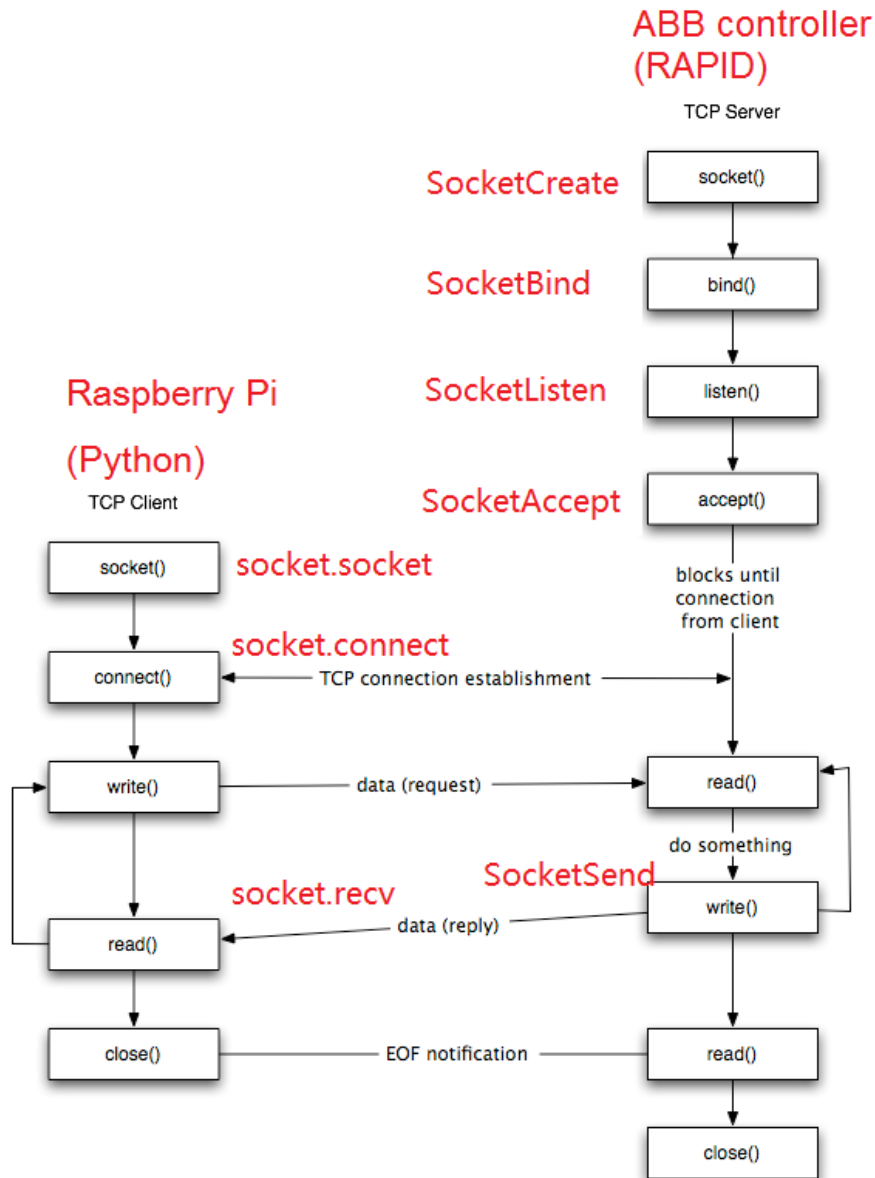


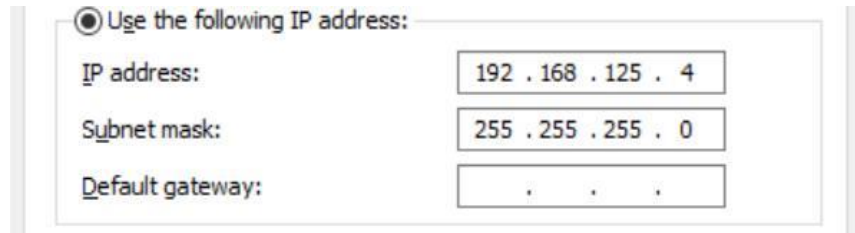
Figure 12. TCP/IP Communication Between Client and Server (Campbell, 2012)

It was very necessary to build the communication between Raspberry Pi and ABB controller. The messages such as robot flange position and pose need to be transmitted from ABB controller to Raspberry Pi. They could be used to calculate the calibration results as inputs. On the other hand, the messages such as calibration results need to be transmitted from the Raspberry Pi to the ABB controller so that a new coordinate system could be updated. These

messages included many floating-point numbers. Since these messages were complex, the communication methods such as the DeviceNet and the ProfiNet was hard to achieve. Also, the RobotStudio SDK environment was hard to run on the Linux system, so we should find the alternative method for the Raspberry Pi to communicate with the ABB controller to capture the current data of the robot at high frequency. The simpler and more efficient way is to use the TCP/IP socket in both the Raspberry Pi and the ABB controller since the ABB RAPID has an internal TCP/IP socket and a TCP/IP socket was well developed with Python in the Raspberry Pi as shown in red labels of Figure 12. Another concern was that the messages of the robot positions and poses should be transmitted when the robot was running the moving commands. In order to do so, we need special program design in the ABB controller to achieve. There were several ways to do so. One way was to use a multi-tasking module in the ABB controller to process communication data, but not all the ABB controller had this module. The second way was to enable a timed interrupt in the ABB controller to process the messages, but the maximum frequency of the timed interrupt was as low as 10Hz. The last way was to use the I/O interrupt in the ABB controller to process the messages. The maximum frequency of the I/O interrupt is as good as 50Hz. We eventually used the I/O interrupt to send the message of robot position and pose to the Raspberry Pi.

We tried both Windows and Linux system to connect to the ABB controller. Both of them could run successfully. The same Python code could run on both systems, but the only difference was the procedure of setting the static IP configuration. Finally, we set up the environment by installing Raspbian OS, then built Python environment on the Raspbian OS. The Python environment enabled us to create a TCP/IP socket, sent a request to the controller and received the position data from robot while receiving the data from the laser displacement sensor at the same time. Moreover, the TCP/IP requires both a server (robot controller) and a client (Raspberry Pi) to have static addresses so that the protocol could proceed. The only valid addresses were any public LAN addresses or the controller service port address 192.168.125.1. Therefore, the IP address of the client should be 192.168.125.X in which X cannot be 1. Therefore, we set up an IP address by modifying the interfaces file from the directory `/etc/network/interfaces`, as shown in Figure 14, and changed the IP address to static. Setting static address on Windows was shown in Figure 13. The port number could be any small numbers (from 0 to 3000) if the ABB controller was set as the server and the port number can only be set

as 5515 if the ABB controller was set to the client. Thus, we set the port number to 1025 in our project.



● Use the following IP address:

IP address: 192 . 168 . 125 . 4

Subnet mask: 255 . 255 . 255 . 0

Default gateway: . . .

Figure 13. Windows IPv4 Configuration for Communicating with ABB Controller

```
# interfaces(5) file used by ifup(8) and ifdown(8)
# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

iface eth0 inet static
address 192.168.125.4
netmask 255.255.255.0

allow-hotplug wlan0
iface wlan0 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

[line 14/23 (60%), col 10/10 (100%), char 358/555 (64%)]

^G Get Help	^O WriteOut	^R Read File	^Y Prev Page	^K Cut Text	^C Cur Pos
^X Exit	^J Justify	^W Where Is	^V Next Page	^U UnCut Text	^T To Spell

Figure 14. Raspberry Pi Configuration of Static IP Address to Connect with ABB Controller

Figure 15 is our code written in the Raspberry Pi to create a client and receive data from the ABB controller. Most importantly, the IP address for the client to connect should only be '192.168.125.1'. This program should be called after the server is established.

```
ip = '192.168.125.1'
port = 1025
client1 = newClient.TCP_client(ip, port)
client1.connect()

class TCP_client:
    def __init__(self, ip, port):
        self.ip = ip
        self.port = port

    def connect(self):
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.s.connect((self.ip, self.port))

    def communication(self):
        BUFFER_SIZE = 1024
        data = self.s.recv(BUFFER_SIZE)
        d = data.split(',')
        return [d[1], d[2], d[3]]

    def close(self):
        self.s.close()
```

Figure 15. Python TCP/IP Socket Code for Connecting to ABB Controller As The Client

Figure 16 is our code written in the RobotStudio to create a server and send the current position and the orientation data to the client. Most importantly, the exception expression of ERR_SOCK_CLOSED should be used after the socket send so that the program error would not appear when the client was unexpectedly closed. Moreover, as mentioned before, the port number in both the RAPID program and the Python program should be same. The data of the robot current position and orientation were changed to strings and were split by a comma.

```

PROC TCP_server() ! the function to build the server
  SocketClose server_socket;
  SocketClose client_socket;
  SocketCreate server_socket;
  SocketBind server_socket, "192.168.125.1", 1025;
  SocketListen server_socket;
  SocketAccept server_socket, client_socket \Time:=WAIT_MAX;

ENDPROC

PROC server_send() ! the function to send robtarget to Raspberry Pi
  VAR robtarget P;
  VAR string x;
  VAR string y;
  VAR string z;
  VAR string rx;
  VAR string ry;
  VAR string rz;

  P := currentRobt();
  SocketSend client_socket \Str:= NumToStr( P.trans.x,3)+"," +NumToStr( P.trans.y,3)+"," +NumToStr( P.trans.z,3)+"," +
    +NumToStr( P.rot.q1,5)+"," +NumToStr( P.rot.q2,5)+"," +NumToStr( P.rot.q3,5)+"," +NumToStr( P.rot.q4,5);
  ERROR
    IF ERRNO = ERR_SOCK_CLOSED THEN
      TCP_server;
      RETRY;
    ENDIF
ENDPROC

```

Figure 16. RAPID Code About TCP/IP Socket Used for Server Application

4.2.4 End Effector Design

This part gives an introduction about the end effector design. In general, our goal of the end effector was that it could stably hold on the tool or object, including the laser range sensor, working object that we wanted to calibrate, and the cylinder shape tools. There were many ways to design the end effector, as long as the design could achieve our goal.

The design of our end effector is shown in

Figure 17, and Figure 18 is the detailed structure. The spherical bolt could be adjusted freely when the pneumatic cylinder was not pushing against the outer bolt holder; it meant that it was able to create some random change in both positions and orientations when we wanted to adjust the cylinder tool. When the pneumatic controller received the signal, it would push the inner bolt holder out, and push the spherical bolt to outer bolt holder. Then the spherical bolt could not move at all due to the frictions between the spherical bolt and the outer bolt holder.

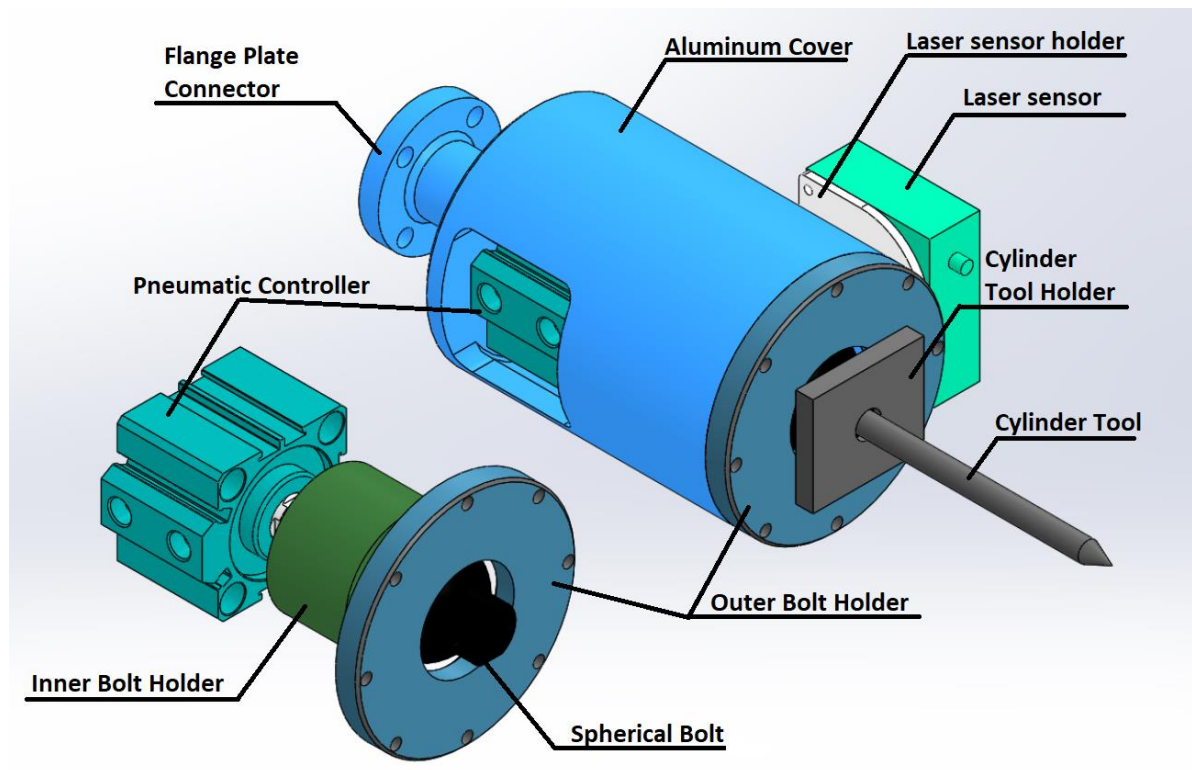


Figure 17. 3D Model of End Effector

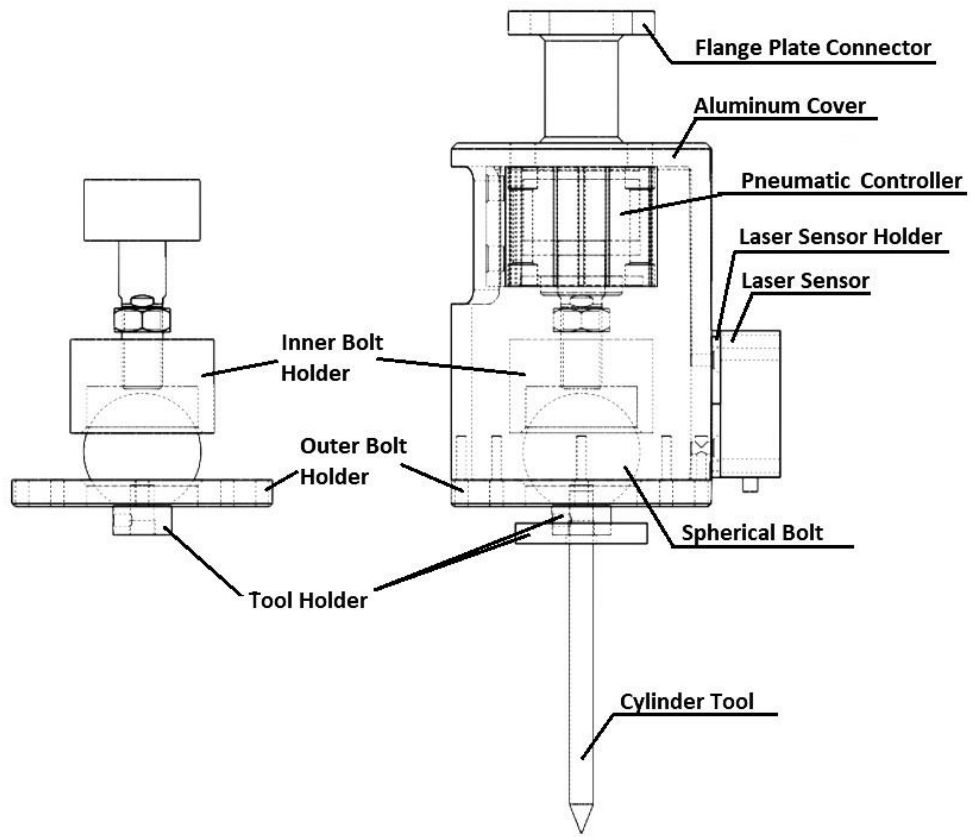


Figure 18. Detailed Structure of The End Effector

4.3 Laser Range Sensor Calibration Method Design

4.3.1 Measurement Tool Calibration

4.3.1.1 Overview

The laser range sensor needs to be calibrated before using it to calibrate the object. The laser position and the direction with respect to the world coordinates are unknown. To have a superior performance of the calibration, a high accuracy of the coordinate system of the laser sensor is needed. As long as we know the sensor coordinate system, while using the laser range sensor to detect a certain point, we could get the position of that point with respect to the world coordinates. Getting the point position from the object for multiple times, we can use our algorithm to calculate the exact position of the object with respect to the world coordinates.

In this section, we are going to cover the edge detection method, Four-point algorithm, program design, and delay evaluation.

4.3.1.2 Theoretical Method

The objective of this calibration method is to determine the TCP of laser sensor's beam, which is to find the relative linear position $[X, Y, Z]$ and the rotational position $[Q_1, Q_2, Q_3, Q_4]$ based on quaternions representation. (Note that the quaternion is a concise way to express the rotational matrix. The transformation of the rotational matrix and a quaternion representation is shown in Equation (1).) (ABB Robotics, RAPID Instructions, Functions and Data types Technical reference manual, 2013) Note that the reason we choose the quaternions representation instead of the typical 3×3 matrix is that the quaternions one is the default representations in the ABB's RAPID program.

$$\begin{bmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ X_3 & Y_3 & Z_3 \end{bmatrix} \Leftrightarrow \begin{bmatrix} Q_1 = \frac{\sqrt{X_1 + Y_2 + Z_3 + 1}}{2} \\ Q_2 = \frac{\sqrt{X_1 - Y_2 - Z_3 + 1}}{2}, \quad \text{sign } Q_2 = \text{sign}(Y_3 - Z_2) \\ Q_3 = \frac{\sqrt{Y_2 - X_1 - Z_3 + 1}}{2}, \quad \text{sign } Q_3 = \text{sign}(Z_1 - X_3) \\ Q_4 = \frac{\sqrt{Z_3 - X_1 - Y_2 + 1}}{2}, \quad \text{sign } Q_4 = \text{sign}(X_2 - Y_1) \end{bmatrix} \quad (1)$$

To find out the TCP of the sensor, we need to use a cylindric-shaped calibrator. By moving the robot above the calibrator's edge and recording their previous positions of the end effector, we can determine the positions of object center and the corresponding rotational positions. Then, the tool coordinates can be determined by positions and orientation of flange plate when the sensor measures the center point.

The following steps are the whole procedures of the program.

Step 1: As shown in Figure 19, the robot with the laser sensor approaches the edge of the cylindric calibrator. On the way moving the laser sensor, the robot is continuously recording the sensor's end effector positions and the distance values from the laser sensor. When the laser is at the one arbitrary point on the edge, labeled as point P₁, the robot marks the instant end effector position (X₁, Y₁, Z₁, Q₁[4]) and the distance value (L₁). The value of L₁ can be customized.

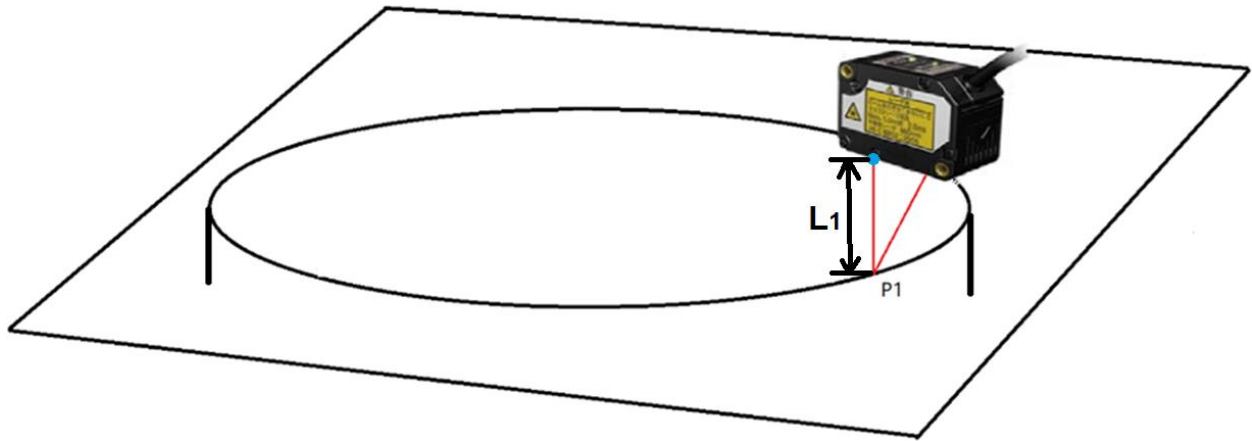


Figure 19. The End Effector Reaches One Edge Point and Record The Position

Step 2: Then the sensor moves to another arbitrarily determined point (P_2) on the edge. The distance value is recorded L_2 . The system tries to find the position that L_2 is equal to L_1 , and the tolerance is set as 0.02 mm. then record L_2 's position ($X_2, Y_2, Z_2, Q_2[4], L_2$). Note that the movement of the sensor is purely linear, and the orientation won't change, and $Q_1[4]$ should equal to $Q_2[4]$

Step 3: Repeat Step 2 to find a point P_3 ($X_3, Y_3, Z_3, Q_3[4], L_3$) that $Q_1 = Q_2 = Q_3$, and $L_1 = L_2 = L_3$. The Step 2 and Step 3 is shown in Figure 20.

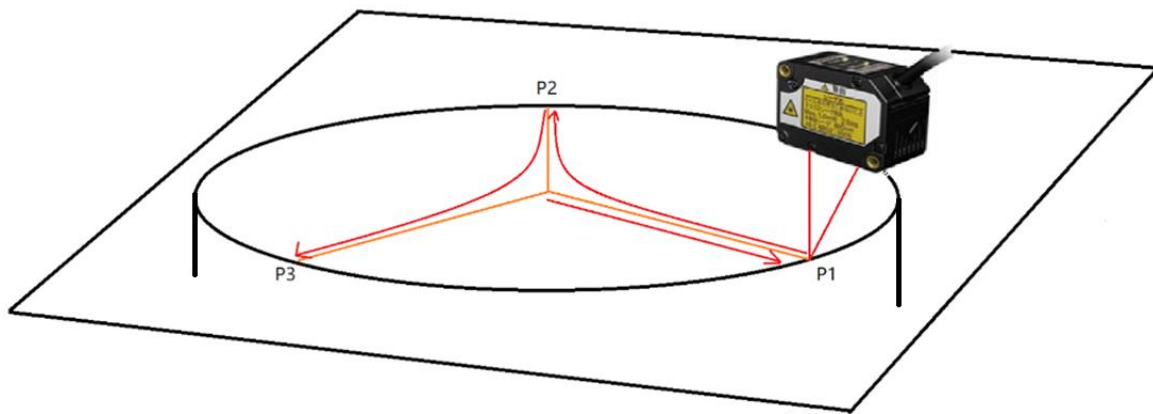


Figure 20. The End Effector Reaches Another Two Edge points

Step 4: As shown in Figure 21, lift the end effector with the sensor for certain height with the same orientation, then repeat Step 1-3 to find the point P_4 ($X_4, Y_4, Z_4, Q_4[4], L_4$), P_5 ($X_5, Y_5, Z_5, Q_5[4], L_5$), P_6 ($X_6, Y_6, Z_6, Q_6[4], L_6$). $Q_4 = Q_5 = Q_6$, and $L_4 = L_5 = L_6$.

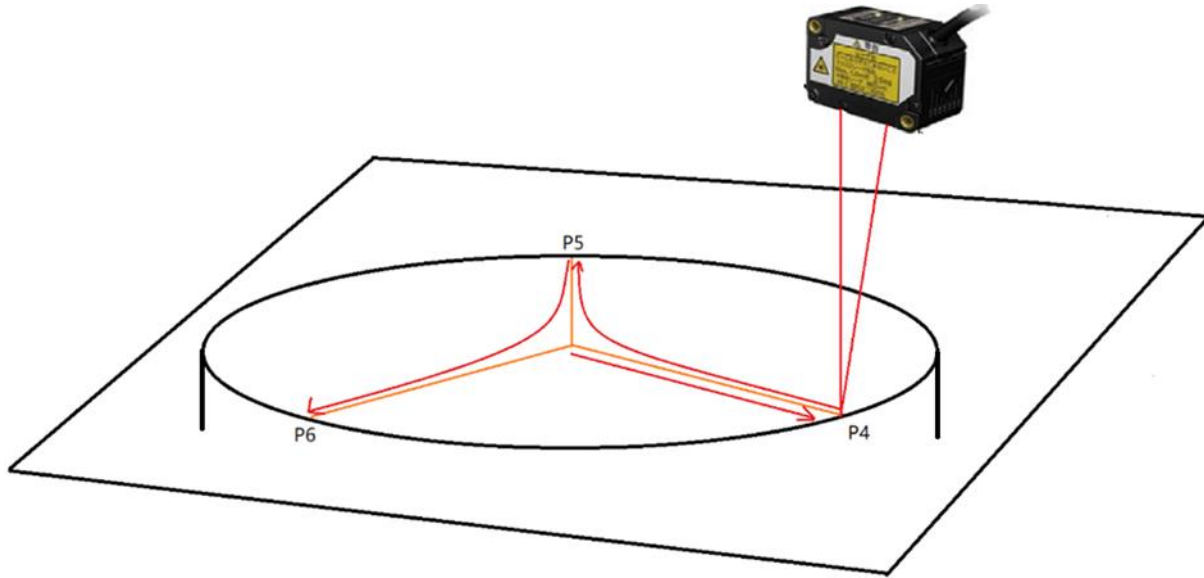


Figure 21. The End Effector Reaches Three More Edge points With Another Height

Step 5: Based on six critical points we measure, we can define two circle's positions. Those two circles form parallel plates, with different distances between the calibration plate and the laser range sensor. Then we can get the vector of two circle's centers, shown in Figure 22. The new tool orientation can be calculated based on the vector of the center line.

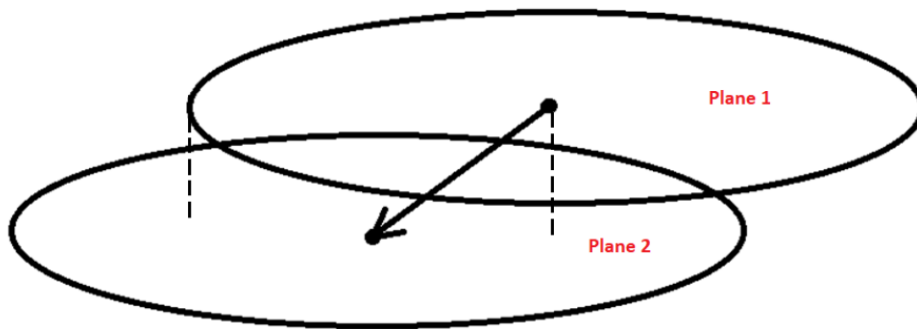


Figure 22. Determine the Vector-Based On Center Points On Two Planes

Note that the direction of the vector is the same as TCP's rotational position $Q[4]$.

Step 6: Switch the orientation of end effector three times, and then repeat Step 1-3 to find points P7-P9, P10-P12, P13-P15, as shown in Figure 23. Notice that the angle between two

orientations should be within 15 degrees. Otherwise, the laser range sensor will not be able to have a reading while scanning the edge.

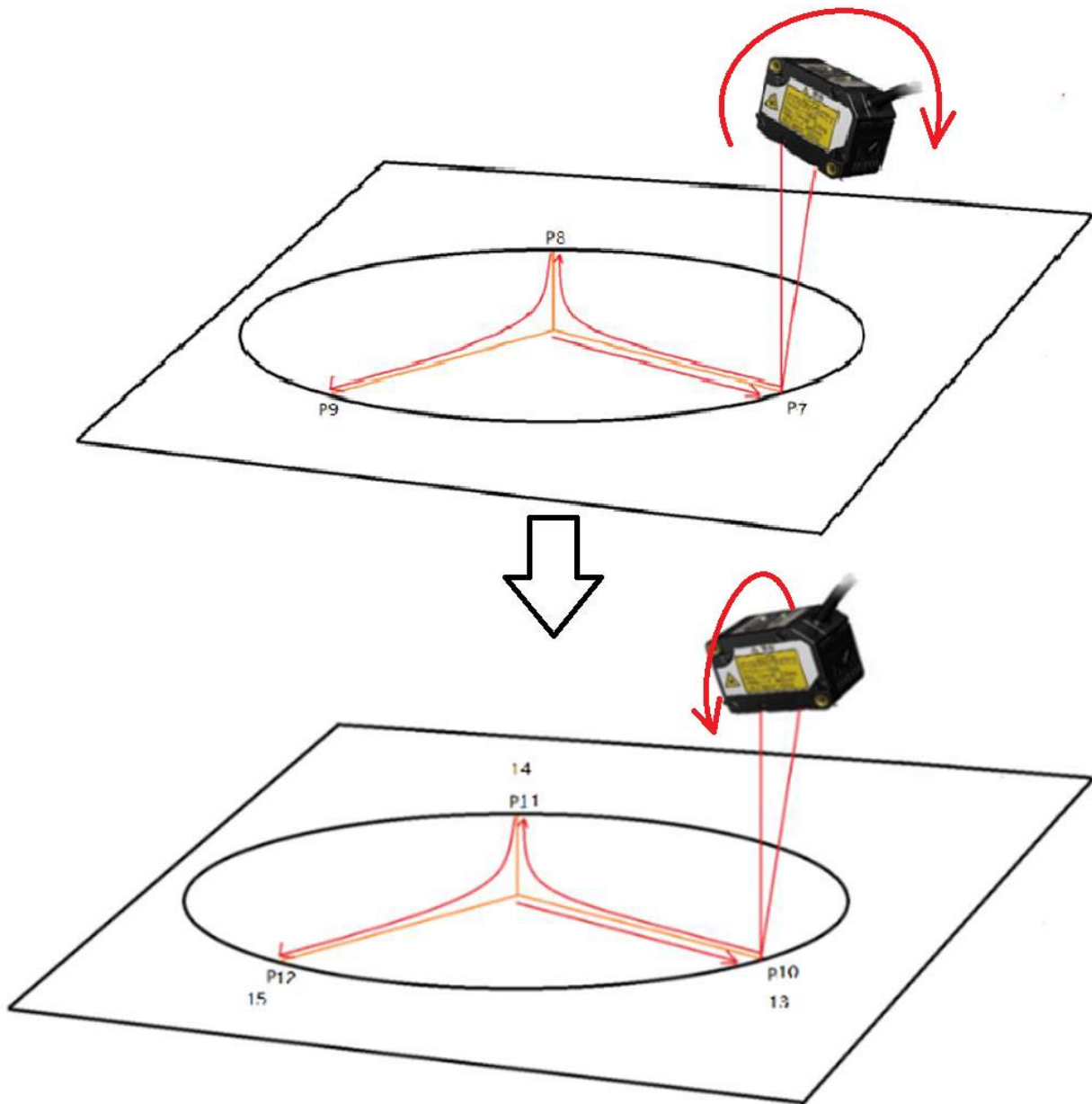


Figure 23. The End Effector Changes the Orientation

Step 7: From Step 6, we can get four positions of the flange plate, when the robot approaches P_c in four different orientations, with respect to Figure 24. One of the orientations is shown in the first image of Figure 24, and the second orientation is shown in the second image.

Those positions of the flange plate are the input of the Four-point method algorithm. The algorithm will output the tool position of the center point (P_C) based on the position of the flange plate. The orientation of the sensor has been calculated in Step 5.

The Calibration of the laser range sensor has finished.

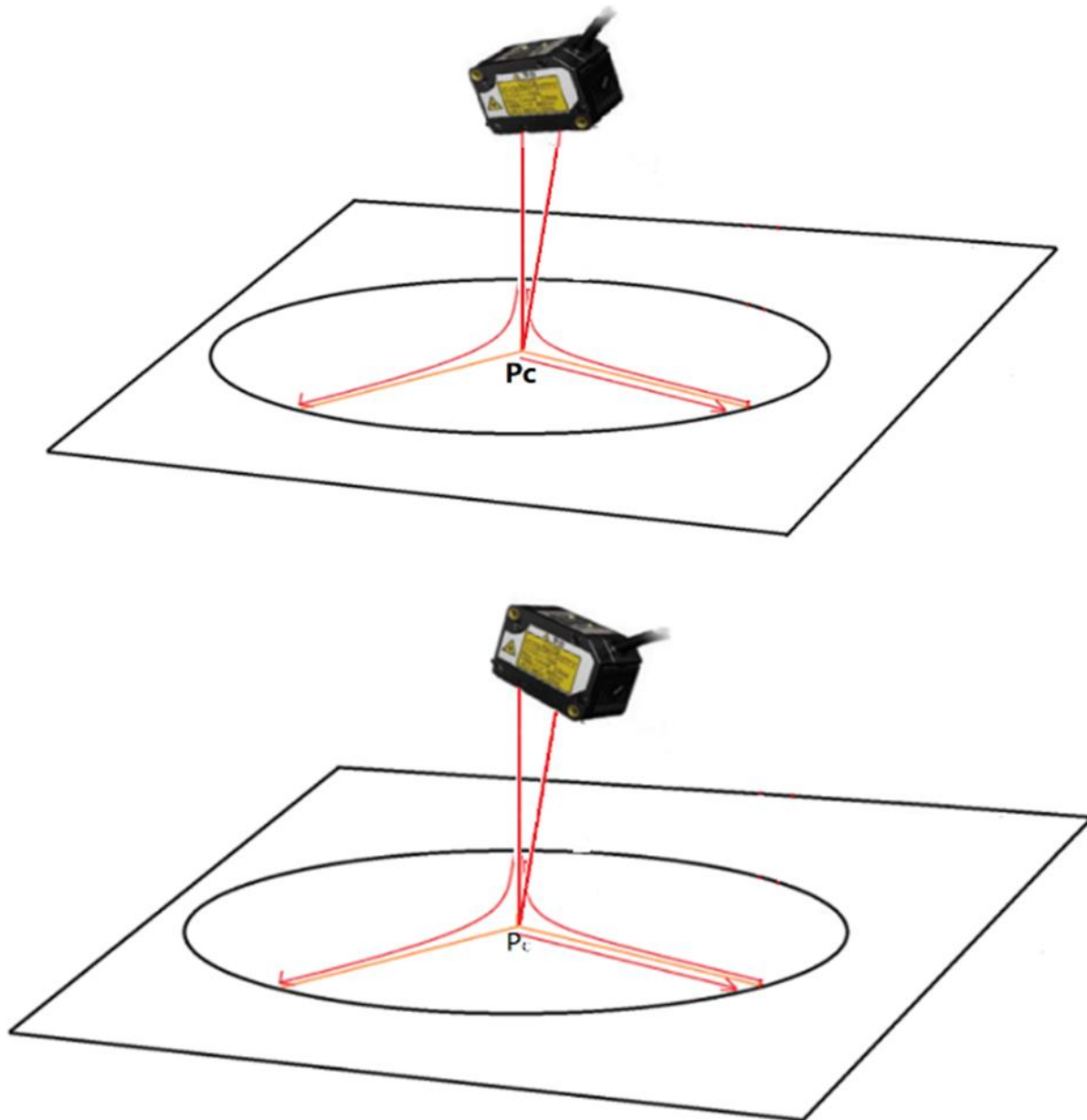


Figure 24. Tool Positions Can Be Determined In the End

4.3.1.3 Least Square Edge Detection Method

There is a limitation for the laser range sensor's detecting angle. In order to let the laser range sensor to work, the receiver needs to receive a sufficient amount of the laser beam data reflected by the detection surface. If the reflected laser beam signal is too weak, the laser range sensor won't have a stable reading. In most cases, it will have noise. An example of how this may occur will now be discussed.

In the process of calibrating the laser range sensor, we need to detect the circular edge of the cylinder shape calibrator. During this process, the laser sensor might accidentally detect something outside of the round surface. One of the most likely situations that might occur is shown in Figure 25. Because the position and the orientation of the calibrator are unknown, the laser might encounter the side surface which, depending on the angle of incidence of the beam, can cause noise that affects our data. For example, when the round surface of the calibrator is perpendicular or slightly tilting away from the laser range sensor, the reading will be accurate. These are the middle and left-handed cases in Figure 18. Contrast those cases with when the round surface of the calibrator is slightly tilting towards the laser beam. This is the right-hand case in Figure 18. Although most of the reflected laser will not be received by the receiver of the laser range sensor, there is going to be a small amount of the laser that will affect the data. This is the noise mentioned earlier.



Figure 25. The Behavior of the Laser Sensor Scanning the Edge in Three Different Poses

To help deal with this problem, a 45-degree beveled edge is incorporated into our calibrator as shown in Figure 26. Now the round surface of the calibrator has been divided into an inner round and an outer circle. The line between the inner round and the outer circle is called the inner edge. The line between the outer circle and the side is called the outer edge.

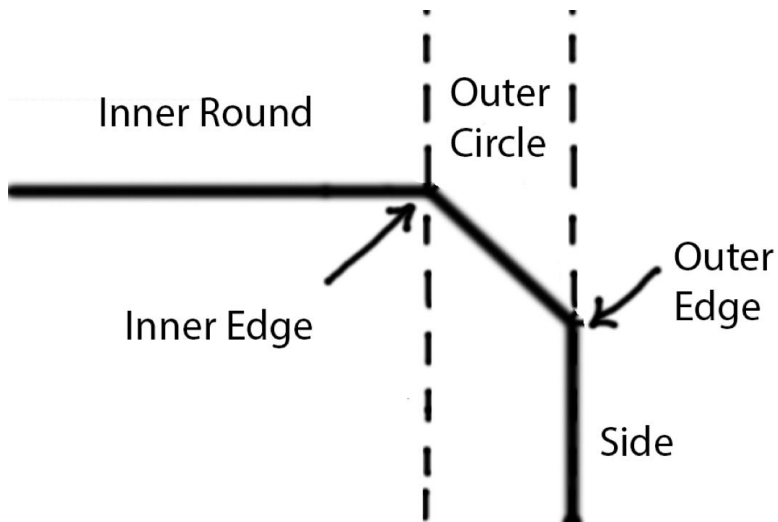


Figure 26. Calibrator Design

Figure 27 shows the situation when the laser range sensor moves scanning the surface of the calibrator. The sensor will still receive noise when scanning the side of the calibrator. However, it can also now detect the inner round and the outer circle.

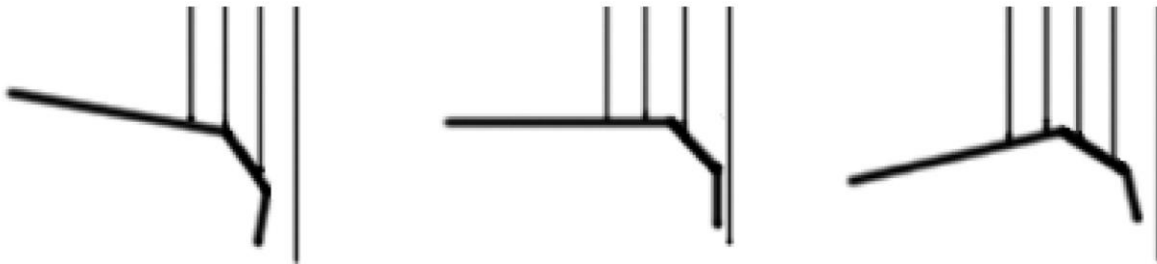


Figure 27. The Behavior of Laser Sensor Scanning the Beveled Edge in Three Different Poses

Figure 28-23 shows the sensor reading while scanning the edge of the calibrator in the three cases above. In the first graph (Figure. 21) where the calibrator surface is positioned perpendicular to the laser beam, the sensor gives perfect readings when detecting the inner round and the outer circle. It has a clear indication after the detection of the outer circle. You can see the clear transformation from the inner round to the outer circle then to the side. There are two turning points which represents the inner and outer edges.

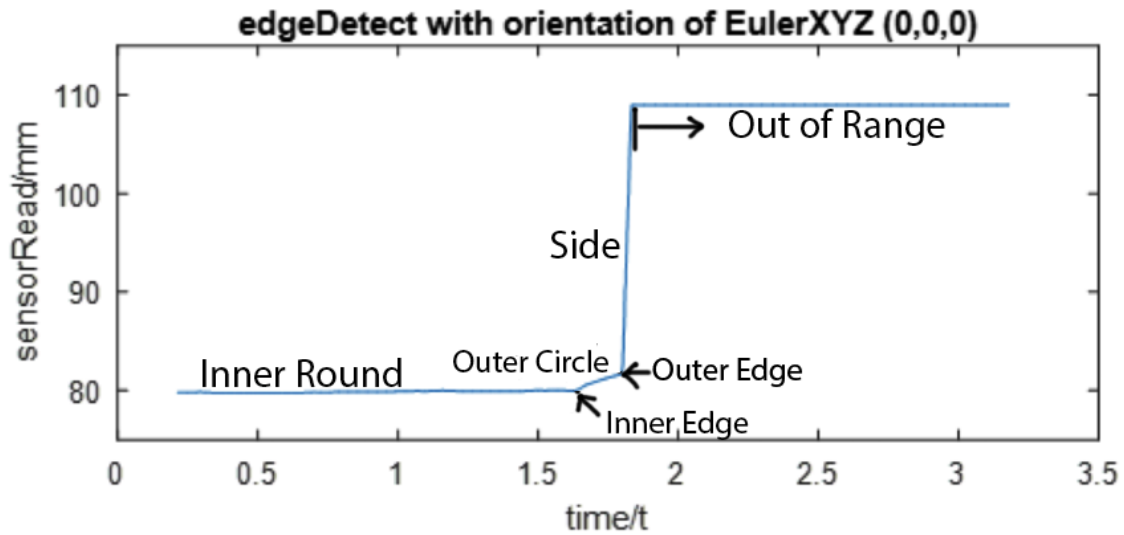


Figure 28. Sensor Reading While Scanning the Calibrator Surface Perpendicular To the Laser Beam

As shown in Figure 29, the calibrator is positioned slightly tilting towards the laser beam. The reading shows clearly when detecting the inner round and the outer edge. But when the laser beam encounters the side of the calibrator, it makes a large amount of noise which gives us unstable readings. This may not happen every time when collecting data, but it certainly will affect our calculation. Although the position of the outer edge might not be accurate, the position of the inner edge is clearly shown.

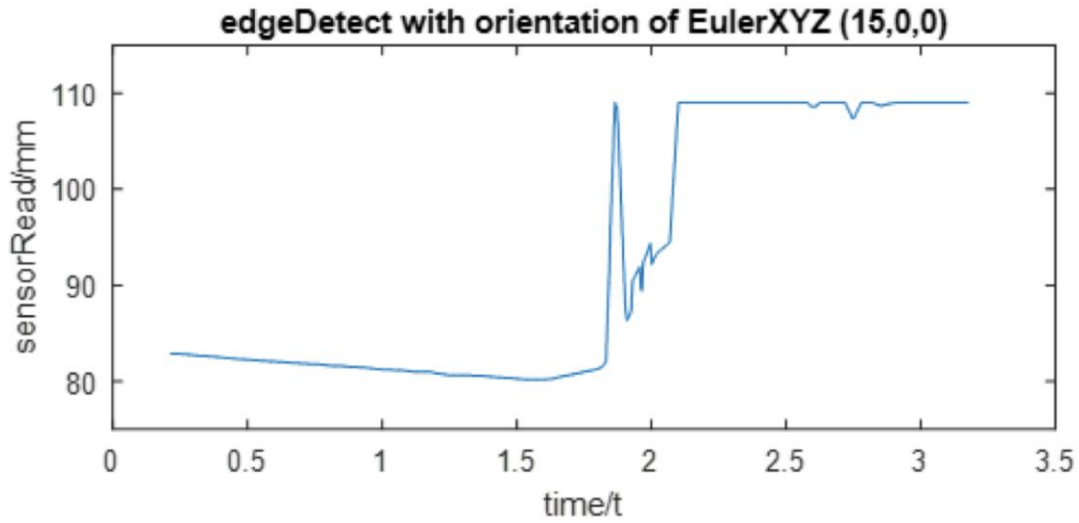


Figure 29. Sensor Reading While Scanning the Calibrator Surface That Is Slightly Tilting Towards the Laser Beam

In Figure 30, the calibrator is positioned slightly tilting away from the laser beam. The sensor gives perfect readings when detecting the inner round and the outer circle. It has a clear-cut after the detection of the outer circle. So, the two turning points of the inner and outer edge are shown.

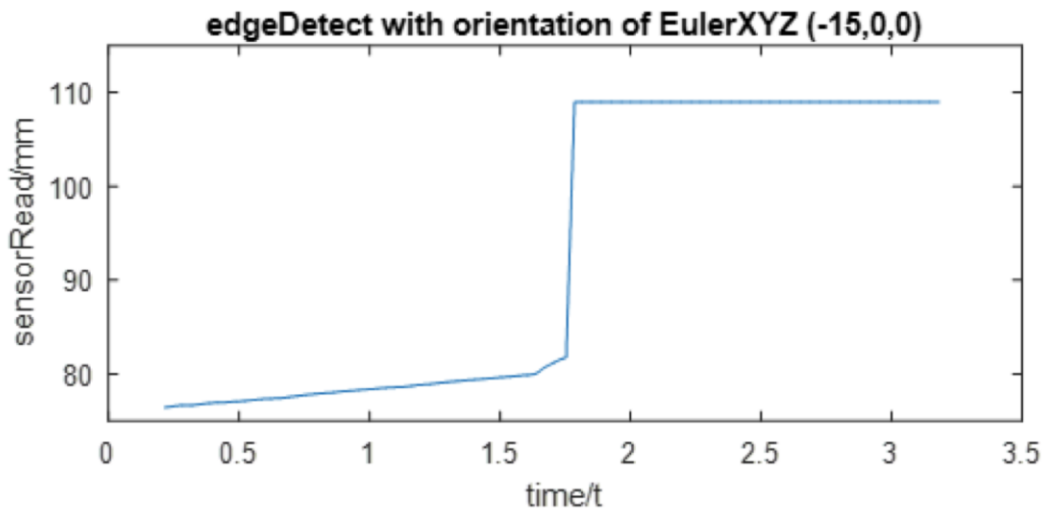


Figure 30. Sensor Reading While Scanning the Calibrator Surface That Is Slightly Tilting Away the Laser Beam

For all cases, the laser range sensor can successfully detect the inner round and the outer circle. Thus, the intersection between the inner round and the outer circle, the inner circle, can be detected which solves the problem.

4.3.1.4 Four-point Algorithm

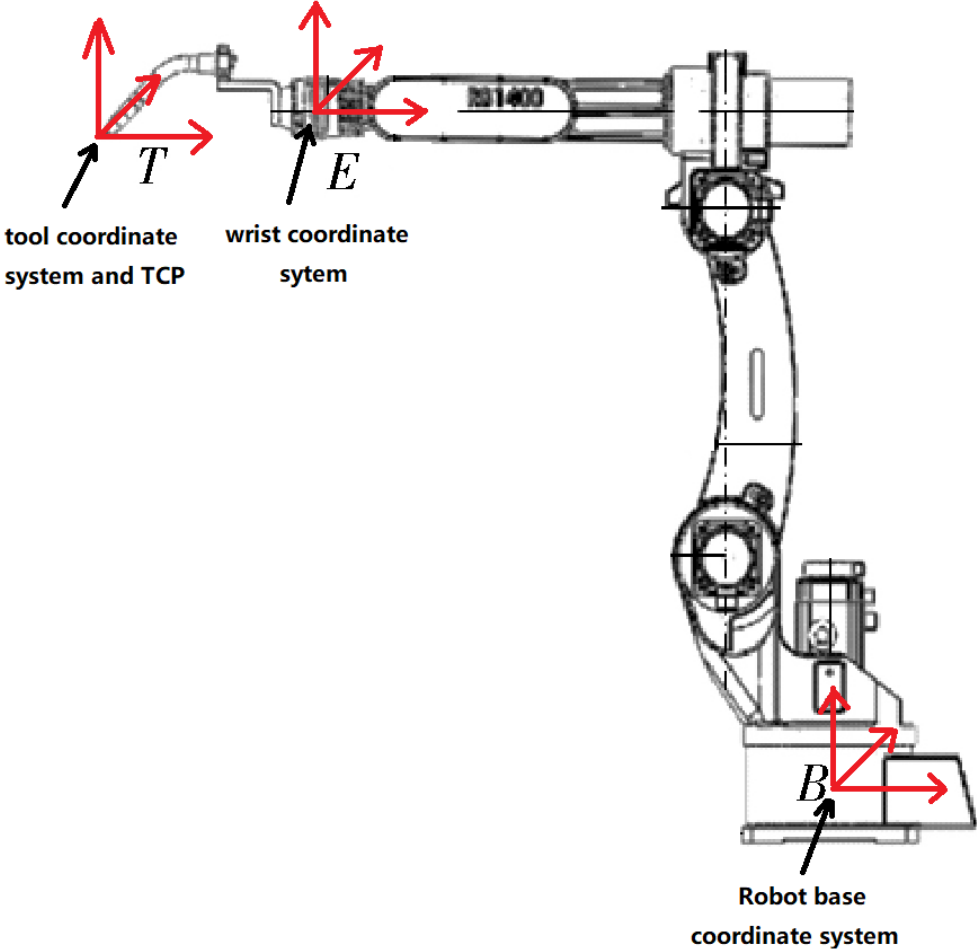


Figure 31: RB_1400 Welding Robot (Hao Gu, 2015)

In this section, an algorithm is introduced to calculate the tool center point (TCP) of the robot. Figure 31 is a side view of a six-axis robot. Part B is the robot base coordinate system, and Part E is the wrist coordinate system. T is the tool coordinate system. The transformation matrix

between coordinates of E and B is defined as ${}^B_E T$. The transformation matrix between coordinates T and E is defined as ${}^E_T T$, and the transformation matrix between coordinates T and B is defined as ${}^B_T T$.

The relationship between the three is:

$${}^B_E T \cdot {}^E_T T = {}^B_T T \quad (2)$$

The expression of ${}^B_E T$ is in Equation (3). ${}^B_E T$ can be obtained by the forward kinematic equations of the robot. It includes the rotation matrix ${}^B_E R$ and the position of the robot flange plate ${}^B P_{E0}$ related to the base. It is easy to get the quaternion (orientation) and the position of the EOAT mounting plate at the end of a robot arm from the ABB controller, and the quaternion can be transformed into a rotation matrix.

$${}^B_E T = \begin{bmatrix} {}^B_E R & {}^B P_{E0} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & x \\ n_y & o_y & a_y & y \\ n_z & o_z & a_z & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$\begin{bmatrix} \cos \theta_1 \cos \theta_2 & -\sin \theta_1 \cos \theta_2 & \sin \theta_2 & x \\ \cos \theta_1 \sin \theta_2 \sin \theta_3 + \sin \theta_1 \cos \theta_3 & \cos \theta_1 \cos \theta_3 - \sin \theta_1 \sin \theta_2 \sin \theta_3 & -\cos \theta_2 \sin \theta_3 & y \\ \sin \theta_1 \sin \theta_3 - \cos \theta_1 \sin \theta_2 \cos \theta_3 & \sin \theta_1 \sin \theta_2 \cos \theta_3 + \cos \theta_1 \sin \theta_3 & \cos \theta_2 \cos \theta_3 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We can use Equation (3) to expand Equation (1) and the result is written as Equation (4). ${}^B_E R$ is the rotation matrix of the endpoint about the base B. ${}^B P_{Ei}$ is the position vector of the endpoint as well as flange plant about the base B. Both of them are known. ${}^E_T T$ is the result that should be obtained finally, which is the tool center point(TCP), expressed as a vector relative to E. ${}^E_T R$ is the tool rotation matrix. Unknown elements ${}^B_T R_{Ei}$ and ${}^B P_T$ are rotation matrix and position of a tip of the tooling (the tip of the welding tool in Figure 24) relative to the base.

$$\begin{bmatrix} {}^B_E \mathbf{R} & {}^B \mathbf{p}_{Ei} \\ \mathbf{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^E_T \mathbf{R} & {}^E \mathbf{p}_T \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} {}^B_T \mathbf{R}_i & {}^B \mathbf{p}_T \\ \mathbf{0} & 1 \end{bmatrix} \quad (4)$$

Since both ${}^E_T \mathbf{R}$ and ${}^B_T \mathbf{R}_{Ei}$ are unknown and not necessary to calculate, simplify the Equation (4), and the result is shown in Equation (5).

Equation (5) is the TCP general equation, and it stands for the general equation that derives the vector of the TCP. ${}^B \mathbf{p}_{Ex}$, ${}^B \mathbf{p}_{Ey}$, ${}^B \mathbf{p}_{Ez}$ and a 3x3 matrix on the left side is positional and rotational matrices of the endpoint mentioned above. Those matrices are used as the input of the Four-point method algorithm. ${}^E \mathbf{p}_{Tx}$, ${}^E \mathbf{p}_{Ty}$, ${}^E \mathbf{p}_{Tz}$ are displacement of x-, y- and z-axis of the TCP. These values are constants, and the result of calculation, ${}^B \mathbf{p}_{Tx}$, ${}^B \mathbf{p}_{Ty}$, ${}^B \mathbf{p}_{Tz}$ are x-, y- and z-axis of the fixed tip point.

$$\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \cdot \begin{bmatrix} {}^E p_{Tx} \\ {}^E p_{Ty} \\ {}^E p_{Tz} \end{bmatrix} + \begin{bmatrix} {}^B p_{Ex} \\ {}^B p_{Ey} \\ {}^B p_{Ez} \end{bmatrix} = \begin{bmatrix} {}^B p_{Tx} \\ {}^B p_{Ty} \\ {}^B p_{Tz} \end{bmatrix} \quad (5)$$

One robot can have various poses while its tool center point faces a certain fixed tip point, and each pose corresponds to a specific type of the TCP general equation. If the robot has N poses pointing at one certain fixed tip point, it can be expressed as N in the TCP general equation. Figure 32 is the RAPID program dataset, and four variables p1 – p4 correspond to four different pose matrices resulting that the tool of the robot approaches to one fixed tip point. These variables will transform to the form of the TCP general equation in the later calculation.

```
p1:=[[844.334476656,-560.669130325,720.723438328],[0.444981,0.549618,0.444972,-0.549465]];
p2:=[[841.026316431,-560.698869294,720.547168069],[0.657892,0.589725,0.390215,-0.259103]];
p3:=[[842.482130557,-560.692865956,721.891574202],[0.619719,0.765393,0.134874,-0.109227]];
p4:=[[842.433239145,-560.666769572,718.647521832],[0.770455,0.623973,-0.0822401,0.101459]];
```

Figure 32: Example of four ABB variable: `robtargets` used for 4-point method

Each pose can be expressed as a form of the TCP general equation with specific constants. When the TCP general equation of pose P_2 is subtracted from the TCP general equation of pose P_1 , we get Equation (6).

$${}^B_E\mathbf{R}_1 - {}^B_E\mathbf{R}_2 \cdot {}^E\mathbf{p}_T = [{}^B\mathbf{p}_{E2} - {}^B\mathbf{p}_{E1}] \quad (6)$$

Expand Equation (6), we get Equation (7):

$${}^B_E\mathbf{R}_1 - {}^B_E\mathbf{R}_2 \cdot \begin{bmatrix} {}^E p_{Tx} \\ {}^E p_{Ty} \\ {}^E p_{Tz} \end{bmatrix} = \begin{bmatrix} {}^B p_{Ex2} - {}^B p_{Ex1} \\ {}^B p_{Ey2} - {}^B p_{Ey1} \\ {}^B p_{Ez2} - {}^B p_{Ez1} \end{bmatrix} \quad (7)$$

If the robot approaches a fixed tip point with four poses, every two individual poses can get an Equation (7). According to binomial coefficients $C\binom{4}{2} = 6$, there will be six instances of Equation (7). Then, combine them into Equation (8) (Hao Gu, 2015).

$$\begin{bmatrix} {}^B_E\mathbf{R}_1 - {}^B_E\mathbf{R}_2 \\ {}^B_E\mathbf{R}_1 - {}^B_E\mathbf{R}_3 \\ {}^B_E\mathbf{R}_1 - {}^B_E\mathbf{R}_4 \\ {}^B_E\mathbf{R}_2 - {}^B_E\mathbf{R}_3 \\ {}^B_E\mathbf{R}_2 - {}^B_E\mathbf{R}_4 \\ {}^B_E\mathbf{R}_3 - {}^B_E\mathbf{R}_4 \end{bmatrix} \cdot \begin{bmatrix} {}^E p_{Tx} \\ {}^E p_{Ty} \\ {}^E p_{Tz} \end{bmatrix} = \begin{bmatrix} {}^B\mathbf{R}_{Ex2} - {}^B\mathbf{R}_{Ex1} \\ {}^B\mathbf{R}_{Ey2} - {}^B\mathbf{R}_{Ey1} \\ {}^B\mathbf{R}_{Ez2} - {}^B\mathbf{R}_{Ez1} \\ {}^B\mathbf{R}_{Ex3} - {}^B\mathbf{R}_{Ex1} \\ {}^B\mathbf{R}_{Ey3} - {}^B\mathbf{R}_{Ex1} \\ {}^B\mathbf{R}_{Ez3} - {}^B\mathbf{R}_{Ex1} \\ \vdots \\ {}^B\mathbf{R}_{Ez4} - {}^B\mathbf{R}_{Ez3} \end{bmatrix} \quad (8)$$

In Equation (8), only ${}^E p_{Tx}$, ${}^E p_{Ty}$, ${}^E p_{Tz}$ are unknown numbers, so this equation is inconsistent. It cannot be directly solved by the nonhomogeneous linear equation method. The coefficient matrix is not invertible, since it is not a square matrix. We used the least squares method of the matrix form to generalize it as an invertible equation. In the end, the TCP position vector can be calculated using the Gaussian elimination method, shown in Equation (9).

$$\begin{bmatrix} E_{P_{Tx}} \\ E_{P_{Ty}} \\ E_{P_{Tz}} \end{bmatrix} = \begin{bmatrix} {}^E R_1 - {}^E R_2 \\ {}^E R_1 - {}^E R_3 \\ {}^E R_1 - {}^E R_4 \\ {}^E R_2 - {}^E R_3 \\ {}^E R_2 - {}^E R_4 \\ {}^E R_3 - {}^E R_4 \end{bmatrix} \setminus \begin{bmatrix} {}^B R_{Ex2} - {}^B R_{Ex1} \\ {}^B R_{Ey2} - {}^B R_{Ey1} \\ {}^B R_{Ez2} - {}^B R_{Ez1} \\ {}^B R_{Ex3} - {}^B R_{Ex1} \\ {}^B R_{Ey3} - {}^B R_{Ex1} \\ {}^B R_{Ez3} - {}^B R_{Ex1} \\ \vdots \\ {}^B R_{Ez4} - {}^B R_{Ez3} \end{bmatrix} \quad (9)$$

Note that, when we implemented the algorithm, we used the Python library called NumPy, which includes the necessary matrix operations for the calculations above.

4.3.1.5 Source of Error

In the calibration of laser range sensor, several factors can cause errors in our result. The main factors include the 1) response time of the sensor and 2) communication delay. The response time listed on the specification sheet, OPTEX CD33-85N-422 has the latency of 8 microseconds. In the calibration of the laser range sensor, we have the Raspberry Pi to receive the sensor reading from through COM1 port and the ABB flange plate position from ABB controller through the TCP/IP.

The communication delay between the Raspberry Pi and the ABB controller can be measured. Let the Raspberry Pi send a signal to the ABB controller asking for the position of the flange plane while recording the time difference between sending data and receiving data. The time difference is the double of the communication delay. As we measured, the communication delay between the Raspberry Pi and the ABB controller is 7 milliseconds.

The communication delay between the laser range sensor and the Raspberry Pi can't be measured in the same way, since the laser range sensor cannot receive information sent by the Raspberry Pi, but we can still estimate it. The communication delay will be influenced by the length of the cable and the length of the package data. The length of the cable is less than 3 meters, and the data size is one-bit digital. The communication between the laser range sensor and the Raspberry Pi should within 9 milliseconds.

In our application, the robot arm will move at a speed of 60 millimeters per second. The amount of offset caused by the laser response latency and the communication delay largely depends on the speed of the robot movement. When an object moves across the U-shaped tool, minor offsets occur, and they are generated towards the direction of motion.

Our solution towards the offset caused by the latency issue is to lower the speed of the robot arm. When the laser range sensor finds the edge of the Cylinder Shape Calibrator, it scans again to find the critical point with the speed of 10 millimeters per second. With the help of the Edge detection method discussed in Section 4.3.1.3, we can minimize our offset.

4.3.1.6 Program Design

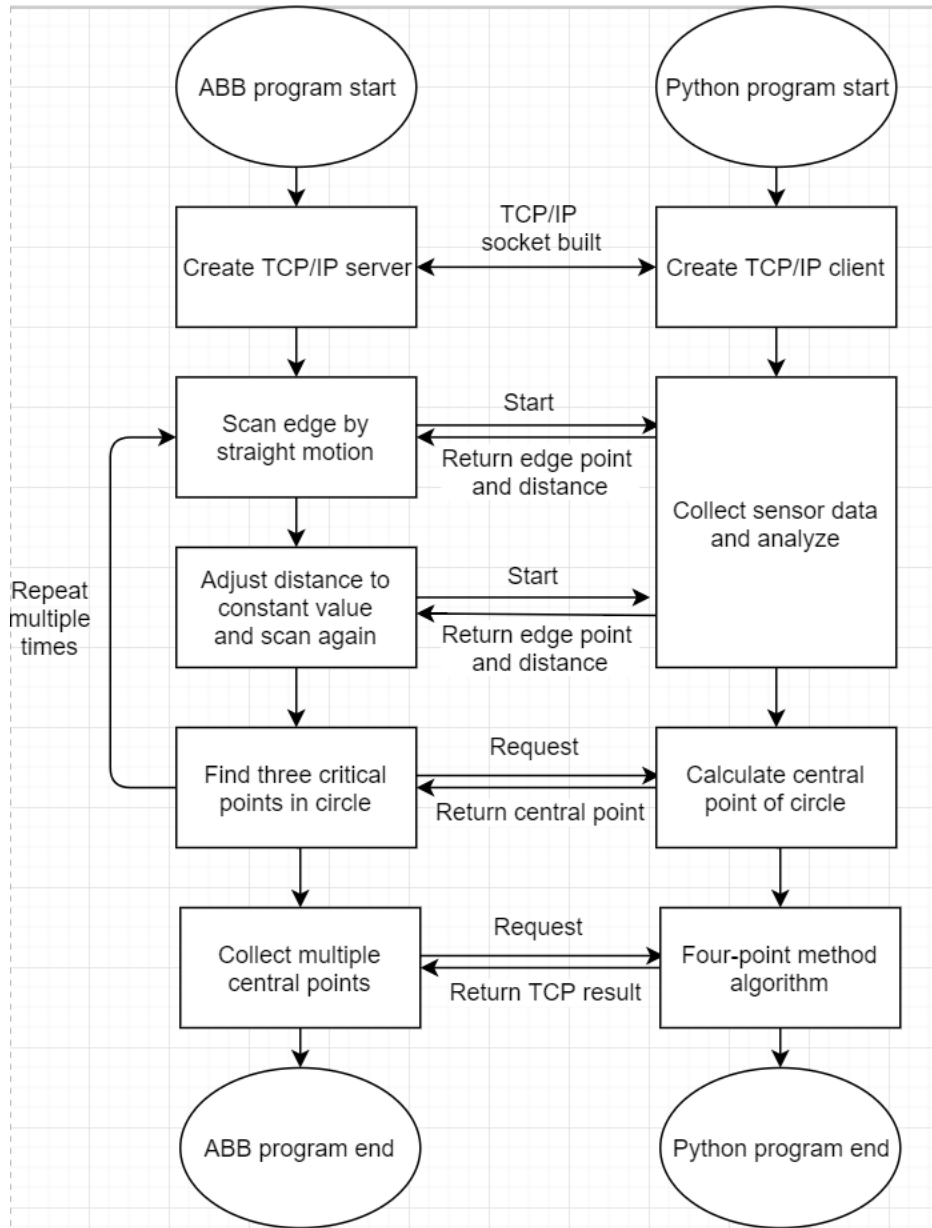


Figure 33. Flowchart of Laser Range Sensor Measurement Tool Calibration

The program design is divided into two parts: ABB program and Python program. The ABB program runs on the ABB controller. The main purpose of the ABB program is path planning because only the ABB program can control the robot automatically. The Python program runs in the Raspberry Pi with Linux environment. The main purpose of using Python to

program our algorithm is because Python has useful libraries to support arithmetical operations, such as matrix operations. These operations are hard to be integrated into the ABB program. Another advantage of Python program is that many histories and datasets from the algorithm calculation can be stored in the Raspberry Pi. They can be analyzed to optimize the algorithm. For example, in 4.3.1.2 Theoretical Method, when the sensor scans the edge of a plate, thousands of readings are recorded in the Raspberry Pi. These readings can become the test case for the least square edge detection algorithm. We can optimize the parameters in the algorithm using these reading as input. It is like the data training in machine learning.

These two programs are connected by the TCP/IP socket as mentioned in 4.2.3 Communication. Figure 33 is the flowchart of both programs. In laser range sensor measurement calibration, the ABB controller is the server and the Raspberry Pi is the client. When the programs are communicating, the Python program runs its finite state machine. The main state of the Python program is to collect the sensor data and the robot position and to call the least square edge detection algorithm when the sensor scans the edge of the plate. Since thousands of the points need to be processed, the Raspberry Pi spends about 0.3-second finishing the calculation. The other states of the Python program are calculating the center points of the circle and the Four-point algorithm. After each state is over, the Python program goes back to a switch statement, waiting for a next request from the ABB program. On the other hand, the ABB program follows a specific procedure according to the theoretical method. The general procedures of the ABB program start with scanning an edge by a straight motion. Adjust the distance between the sensor and the plate to a constant value. Scan again to find three critical points in a circle, and to collect multiple center points. Finally, the calibrated results are updated to the ABB internal program data from the Four-point method algorithm.

4.3.2 Object Measurement and Calibration

4.3.2.1 Overview

The method we implemented in this section is called 3-plane method calibration. This method mainly focused on calibrating the working objects that have at least three independent faces or planes (not parallel faces). After finishing calibrating the laser range sensor, we can use the sensor to determine the position of a point on the object. Within the range of the sensor reading, we could get the position of that point with a very high accuracy. Then we can apply our algorithm to get the real-world position of the object.

We could either have the object held by the end effector to calibrate or have the laser range sensor held by the end effector to calibrate. Both situations are depended on the condition of the work object. For example, the object cannot be picked up due its size or weight.

4.3.2.2 Sensor Detected Point

After we calibrated the laser range sensor in section 4.3.1 Measurement Tool Calibration, we need to know the position of the detected point with respect to the world coordinate system. We have calculated the position of the laser range sensor and the direction vector of the laser. The reading of the sensor is the magnitude of the direction vector. We use the Equation (10) to calculate the face vector based on the collected data from the laser range sensor. In this equation, the rotation matrix of the flange plate represented as Part A. The sensor point position represented as Part B, Coordinate system E in Figure 34. The position of the flange plate is Part C, Coordinate system E in Figure 34. Therefore, we can calculate the world position of the sensor point, represented as Part D.

$$\begin{matrix}
 \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} & \cdot & \begin{bmatrix} {}^E p_{Tx} \\ {}^E p_{Ty} \\ {}^E p_{Tz} \end{bmatrix} & + & \begin{bmatrix} {}^B p_{Ex} \\ {}^B p_{Ey} \\ {}^B p_{Ez} \end{bmatrix} & = & \begin{bmatrix} {}^B p_{Tx} \\ {}^B p_{Ty} \\ {}^B p_{Tz} \end{bmatrix} \\
 \text{A} & & \text{B} & & \text{C} & & \text{D}
 \end{matrix} \quad (10)$$

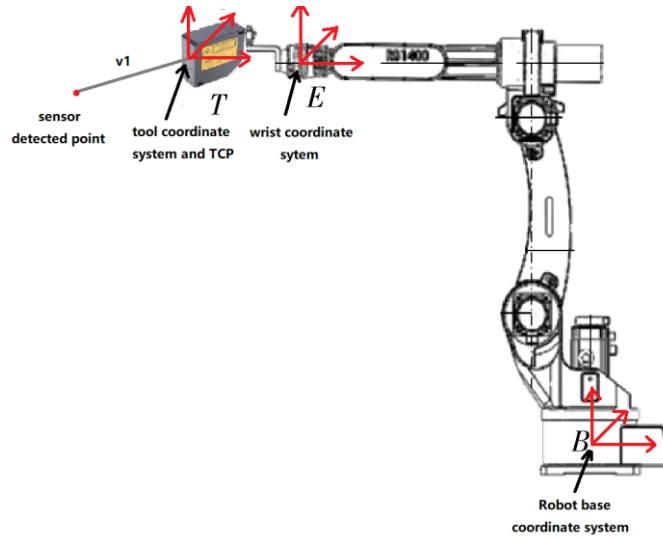


Figure 34. Sensor Detected Point Method

Then, we use Equation (11). p_1 is the direction vector of the laser sensor, shown v_1 in Figure Y, found in section 4.3.1. R_0 is the rotation matrix of the flange plate. Both of them are determined, we can calculate the world position of the sensor-detected point (p_0) based on the reading of the sensor (magnitude).

$$\vec{p}_0 = R_0^{-1} \cdot \vec{p}_1 \quad (11)$$

4.3.2.3 Theoretical Method

The procedure of achieving the calibration is shown in Figure 35. Before we run the program to calibrate the working object, the model of our working object has already imported to the RobotStudio. The first step, we need to select three points on every three independent faces in RobotStudio. The position of those points, with respect to the world coordinates, can be told by the RobotStudio. Then we can easily find out the Theoretical Normal Vector of each face. After knowing the vector of three faces, we can calculate the Theoretical Cross Points of three faces. Those faces are not necessarily physically crossed.

Then, we need to create a coordinate system for the model in the RobotStudio. RobotStudio will also show the Theoretical Vectors and the Origin of the coordinate system, with respect to the world coordinates. Then, we can calculate the angle between each normal vector of three faces and the coordinate system. Also, we can calculate the relationship between the Theoretical Cross Points and the Origin of the coordinate system.

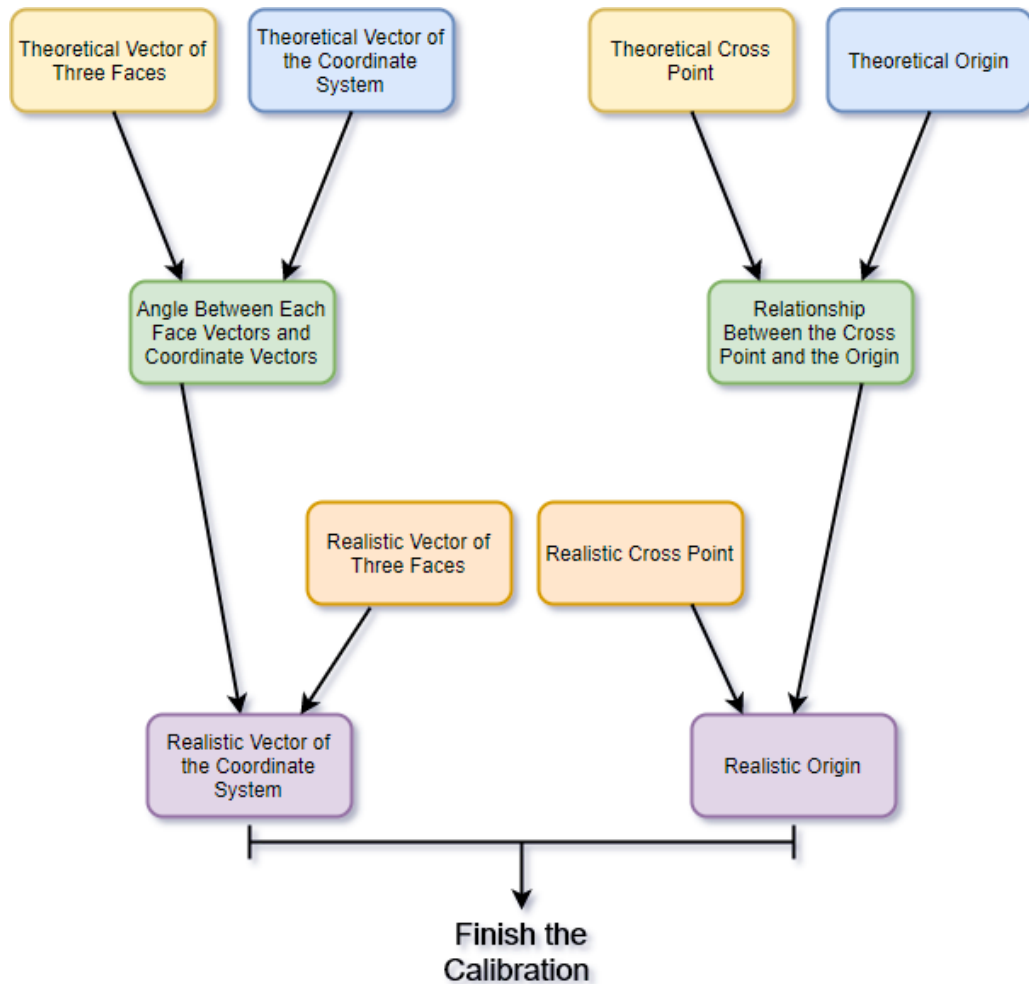


Figure 35. Theoretical Process Diagram

Second, we will let the laser range sensor go to the points, totally nine points in three different faces, which we have picked in Step 1. We have a Python-programmed trajectory generator which will automatically generate the robot trajectory based on the position of those

points, with respect to the world coordinates. Using the method of 4.3.2.2, we could find the realistic position of them, with respect to the world coordinates. Using the same method in Step 1, we could calculate the Realistic Normal Vector of three faces and the Realistic Cross Point of three faces.

Knowing the angle between each face vector and the coordinate system, and the Realistic Normal Vector of three faces, we could calculate the Realistic Vector of the coordinate system.

Knowing the relationship between the Theoretical Cross Points and the Origin of the coordinate system, and the Realistic Cross Point of three faces, we could calculate the Realistic Origin of the coordinate system.

In short, we can find the relationship between the theoretical faces and the coordinate system first. Based on that and the vectors of the realistic faces using the data we got from the laser range sensor, we could find out the realistic coordinate system.

There are possible offsets of the object during manufacture, but most of them would not influence the result of calibration.

4.3.2.3 Source of Errors

In the calibration of the work object using Three-plane Method, the sources of the error come from 1) the error of the workpiece 2) repeat accuracy of the laser range sensor 3) the repeat accuracy of the robot arm 4) the error in the calibration result of laser range sensor. Because the sensor is sampling data in a stationary pose for every point on the object's plain, there is no error regardless of delay and other motion-related problems.

The error of the workpiece is not something we can measure or evaluate precisely. We can only confidently say the error of the workpiece is within 0.02 mm for every edge and distance between surfaces. Although the surface roughness is another subject we are interested in, we don't have the equipment to measure or to evaluate. The repeat accuracy of the laser range sensor is 0.01 mm as listed on the specification sheet. The repeat accuracy of the robot arm is 0.02 mm. We didn't optimize any of the error listed above, and we don't think these errors will significantly affect our result.

The error in the calibration result of the laser range sensor is within 0.15mm. The optimized solution is discussed in section 4.3.1 Measurement Tool Calibration.

4.3.2.4 Program Design

Different from other calibration methods, the Three-plane method needs to process the theoretical model of an object. The calibration trajectory of the robot will be changed with a changed object that needs to be calibrated. It means that the ABB program must be flexible. However, we want to keep the Raspberry Pi program constant. Therefore, we created an ABB program generator using Python as shown in Figure 36. The paths in the ABB program are created by this generator based on the input of the theoretical data, such as the positions of nine selected points and the theoretical coordinate system. The input data is stored in the ABB program. It is sent to the Raspberry Pi for the calculation after the program is communicating. After the algorithm calculation, the Raspberry Pi send the calibrated result back to the ABB program.

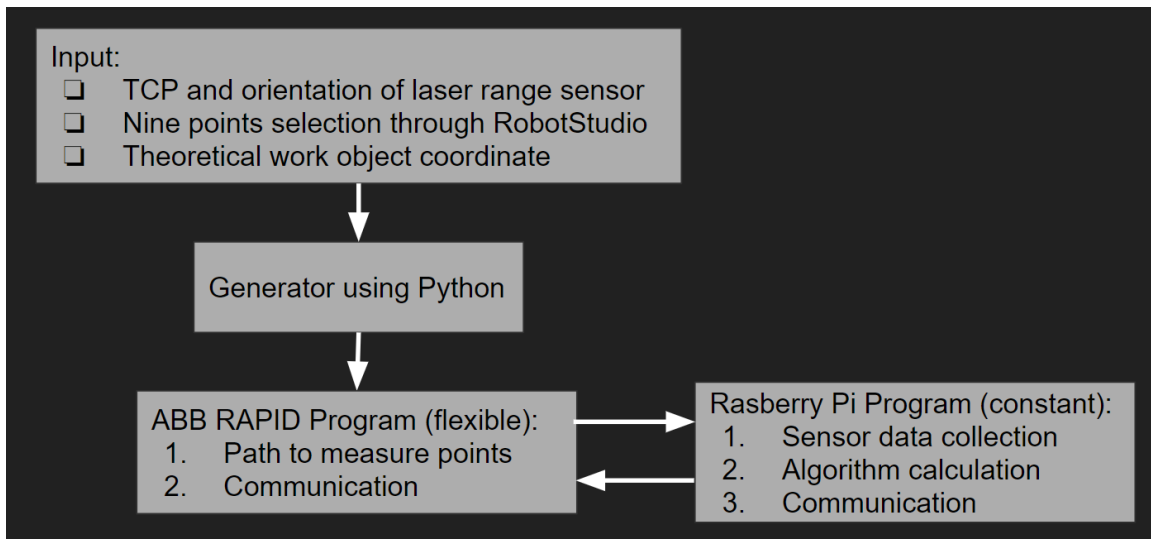


Figure 36. The Flowchart of the Three-plane Method Implementation

4.3.3 Application: Blisk Calibration

4.3.3.1 Overview

Other than the Three Plane Method, we can also use edges to detect a coordinate system of an object. For example, the blisk shaped object shown in Figure 37, is not suitable to use planes to calibrate, but we can define the position of the inner circle by applying the edge detection method to define the whole workpiece. Methods regarding of the overall solution, and edge detection and possible error will be discussed in the following section.

4.3.3.2 Theoretical Method

The calibrated laser range sensor is going to be used for this task. The goal is to calibrate the blisk, which means we want to know the position of blisk's central point and the normal vector.

The blisk has a centroid, which is also the centroid of the inner circular edge, the red circle as shown in Figure 37. Therefore, we could get the centroid of the blisk by finding the inner circle of the edge. The laser range sensor is going to be used in this task. The laser range sensor is installed on the robot end effector.

First, the laser range sensor will follow a trajectory to detect the edge, from the center to the outside. The detected edge point is called the critical point on the inner circle as shown in Figure 37. Using the method that we discussed in section 4.3.2.3 Theoretical Method, we will find out the position of the three critical points.

Using the same method that we discussed in section

4.3.1.3 Least Square Edge Detection Method, the central point and the normal vector can be calculated. However, in this case, we are going through the entire process for one time instead of three. Also, we do not need to change the height and the direction of the laser.

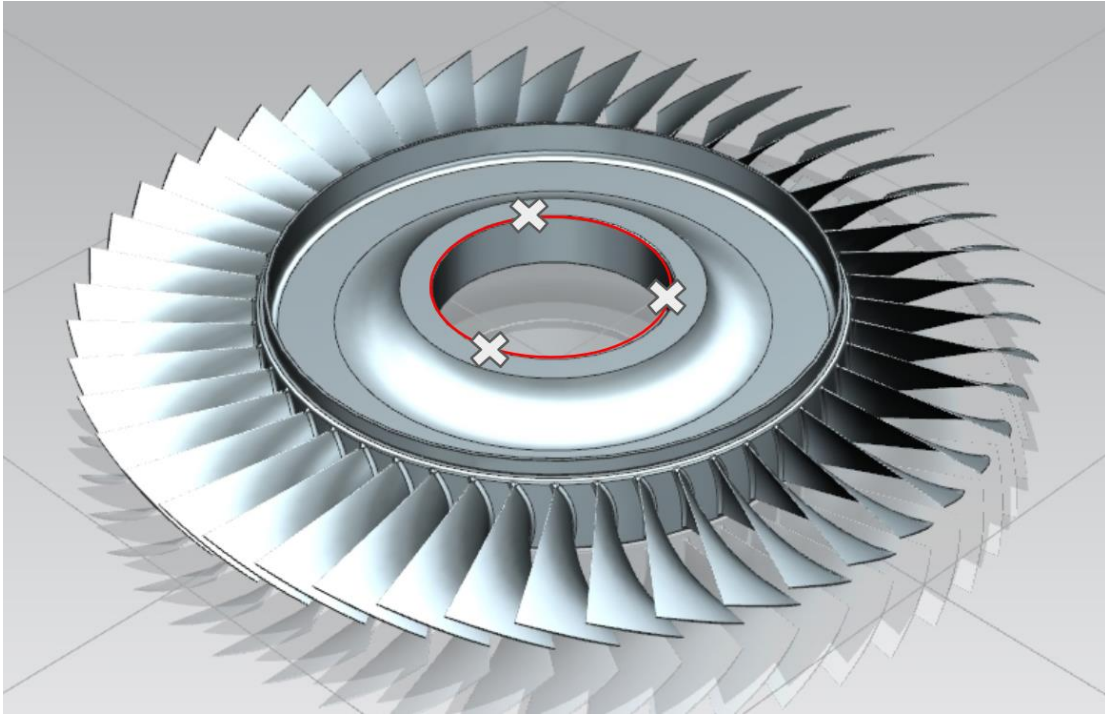


Figure 37. Blisk inner Circle and the Critical Points (Spasić, 2012)

4.3.3.3 Least Square Edge Detection Method

a) General method

To determine the edge of the blisk object, the laser range sensor will need to scan from the center to the surface of the blisk as shown in Figure 38. The sensor will first cross the table surface. After its laser beam encounters the edge of the blisk, it will scan across the blisk surface.

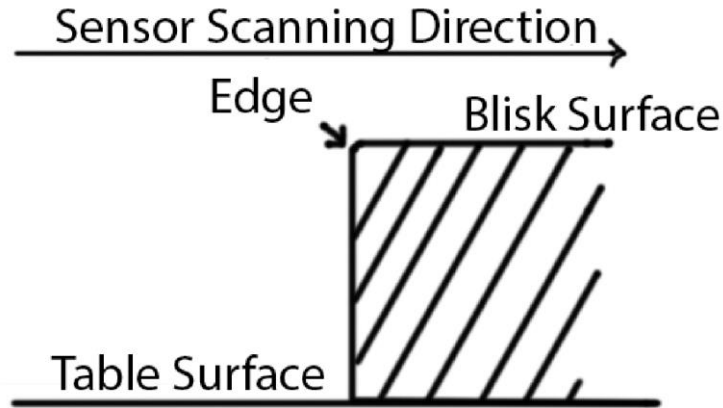


Figure 38. Representation of Sensor Scanning the Edge of the Blisk

After collecting data, we got three graphs that can determine the point of the edge. An example reading is shown in Figure 39. As the sensor moves from the center of the blisk towards the edge, the sensor reading starts from around 95 millimeters which should be the information about the table. In this case, the sensor’s detection range is large enough to detect the table. For sensors that have a limited detection range, they might not have any valid reading and simply showing the information indicated “out of range”. When the sensor laser beam encounters the edge, the sensor reading drops to around 65 millimeters.

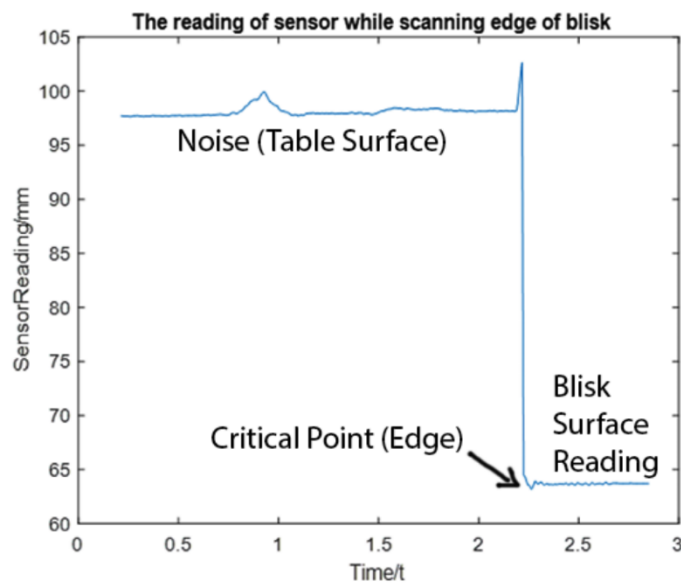


Figure 39. The Reading of Sensor while Scanning Edge of Blisk

In the algorithm, we can eliminate the noise of the table surface by setting a threshold. In this case, we set the threshold to be 85 millimeters which means any data beyond that are ignored. Then we can find the slope of the vertical and horizontal line to get the cross point using the least square method as shown in Figure 40. Thus, we can define the edge of the blisk workpiece.

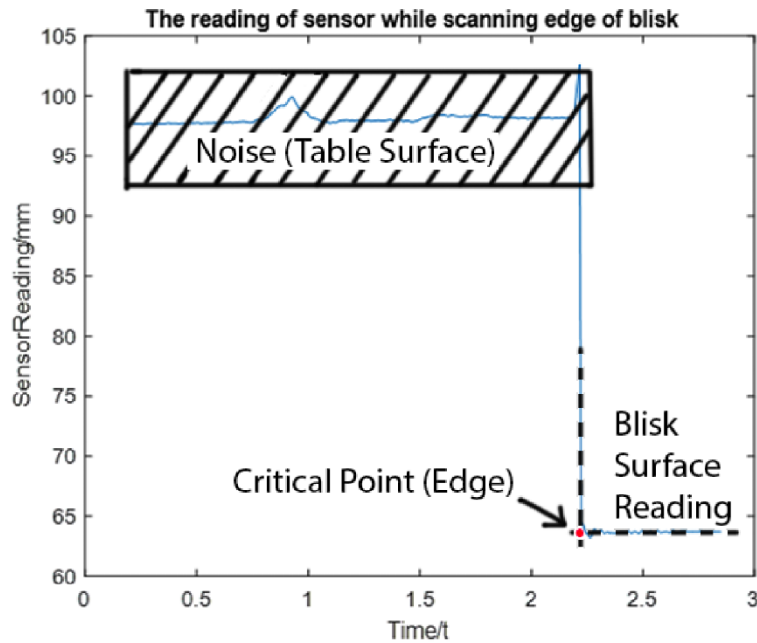


Figure 40. Analysis of the Sensor Reading

b) Noise Cancellation of the Blisk Surface

In a certain situation, where the sensor reading is influenced by the blisk surface, the algorithm will pick up the most effective data to compute. For example, our blisk workpiece has a tilted surface that can influence our scanning result as shown in Figure 41. The surface slope is larger than the maximum detection angle of our laser range sensor.

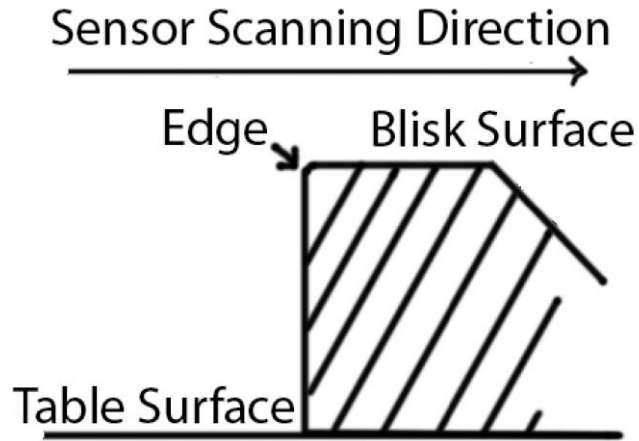


Figure 41. Representation of Sensor Scanning Another Edge of the Blisk

The laser range sensor got an unstable result after it passed the critical point which is the edge of the blisk as shown in Figure 42. Thus we cannot directly apply the Least Square Method to compute the critical point of the graph. The solution is to find the first period of data that is stable enough to indicate the flat surface. Any data that is off the slope will not be considered when finding the critical as shown in Figure 43.

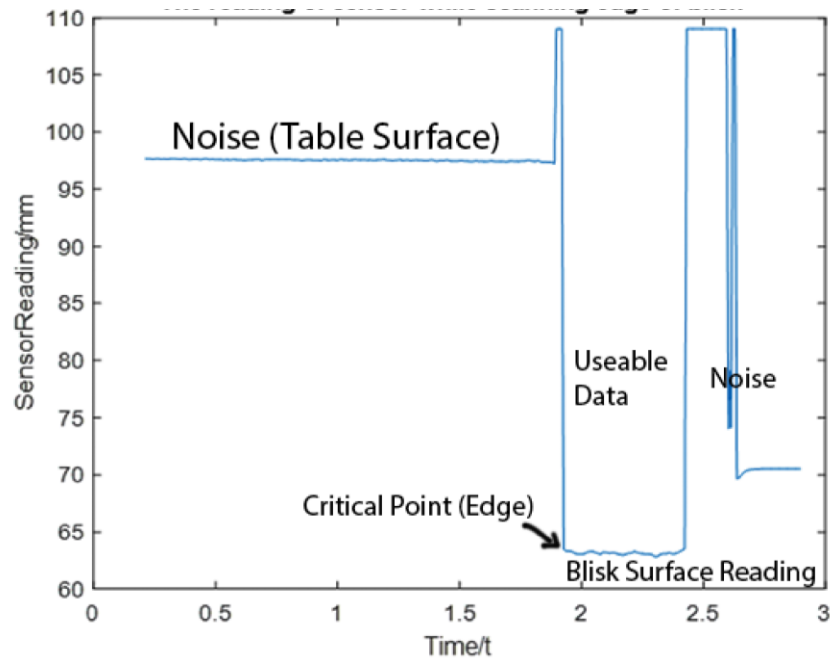


Figure 42. The Reading of the Sensor while Scanning Another Edge of the Blisk

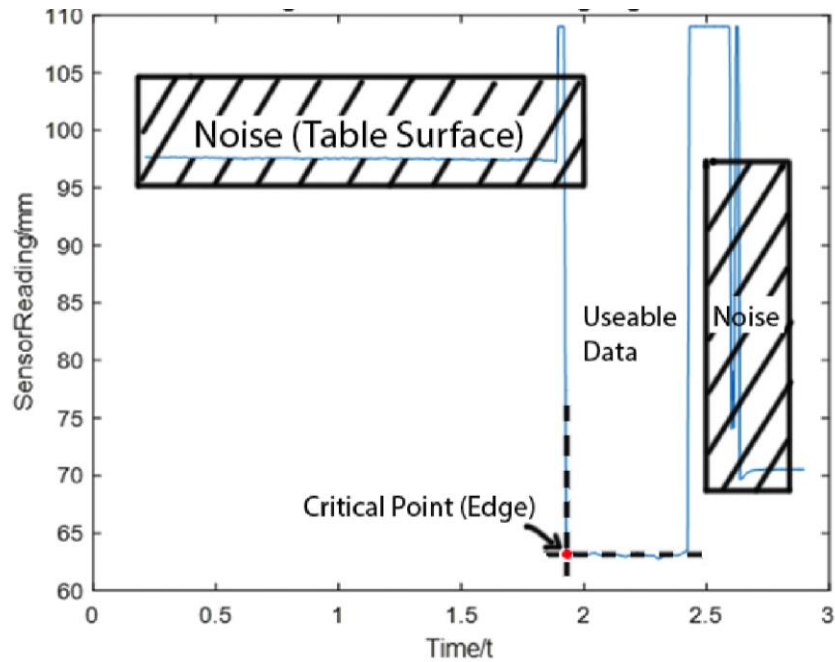


Figure 43. The Reading of the Sensor while Scanning Another Edge of the Blistk

4.3.3.4 Source of Error

In the calibration of the blisk using the edge detection method, the source of error is mainly caused by 1) the sensor delay, 2) communication delay and 3) the error in the calibration result of the laser range sensor. The error of the workpiece is considered but insignificant for our result so we won't be discussing it.

The sensor's delay and communication delay are already discussed in section 4.3.1.5 Source of Error. The error in the calibration result of the laser range sensor is within 0.15mm. The optimized solution is discussed in section 4.3.1 Measurement Tool Calibration.

4.4 U-Shape Sensor Calibration Method Design

4.4.1 Measurement Tool Calibration

4.4.1.1 Overview

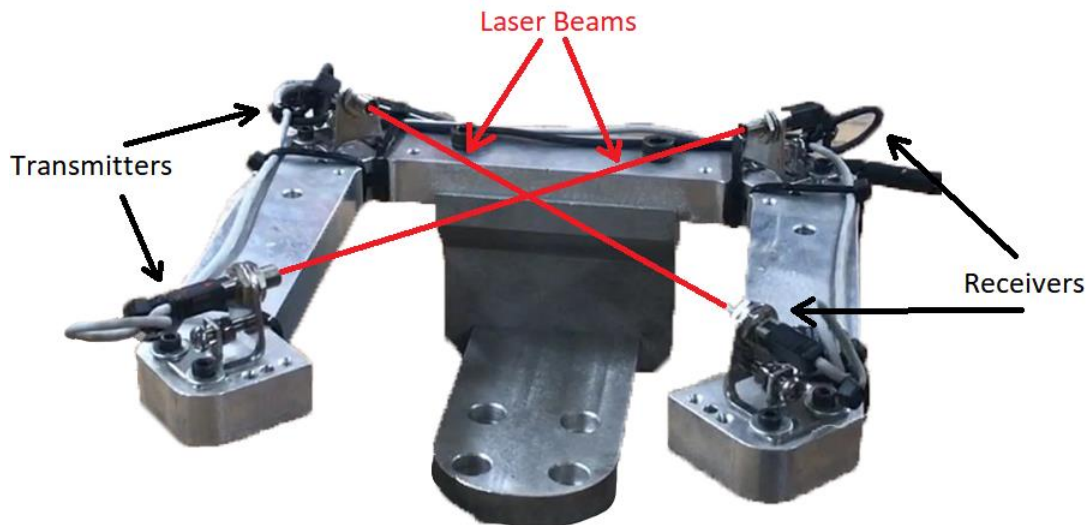


Figure 44. U-shaped Measurement Tool

Before calibrating the working objects, the U-shaped calibration tool sensor-stand set needs to be calibrated. The relative position of the calibration tool is unknown. To have a superior performance of the calibration, a high accuracy of the calibration tool is needed. The U-shape calibration tool set includes a U-shape aluminum stand and two pairs of LV-S71 laser sensor. In each part, a laser transmitter and a receiver are mounted on each corner, facing diagonally.

To guarantee the precise measurements, we need to calibrate the measurement tool first. A calibration measurement tool and a cylindrically shaped rod are needed for this calibration. During the calibration, the rod mounted as an end effector will follow a new circular path of multiple positions inside the U-shape measurement tool. The laser sensor would detect if the object blocked the laser beam. Then we apply the algorithm to get the accurate coordinate system.

4.4.1.2 Theoretical Method

Before calibration begins, we need to set up the U-shaped calibration tool first. The two laser beams should be coplanar, so we need to slightly adjust the laser transmitter and the receiver's position (For more details about the adjustment method, see 4.4.1.3 Measurement Tool Adjustment). After this procedure, the calibration tool adjustment is successfully set up.

In the calibration, the calibrated object, which is coaxial and symmetrical, is mounted on the end effector. Then the end effector with the calibrated object approaches the U-shaped tool, moves the object in a circular trajectory inside of it, shown in Figure 45. While following the trajectory, the object blocks and unblocks the two laser beams for four times. In one loop, the system can receive in total eight digital signal interruptions, shown in Figure 46. The system records the instant position of the flange plate at each critical moment. Then it calculates the average value of two positions, which are the flange plate positions when the tool starts blocking and unblocking one beam, as shown in Figure 47. The average value, in theory, is the flange plate positions when the tool's axis intersects with one of the laser beams.

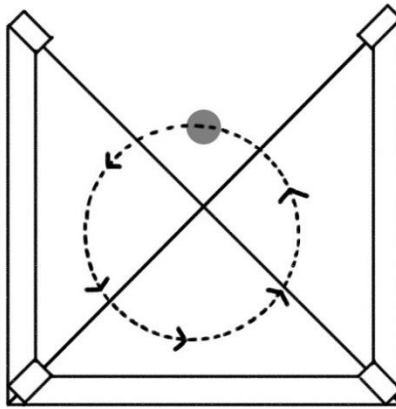


Figure 45. Top View of Objects Moving Alongside Circular Trajectory

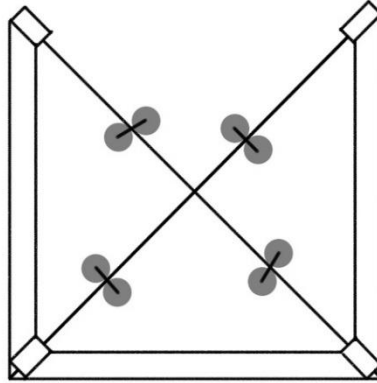


Figure 46. Top View of Captured Eight Critical Points Running One Loop

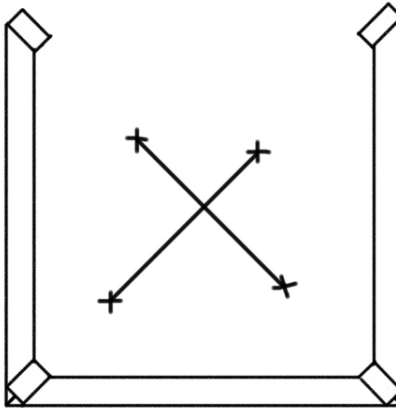


Figure 47. Top View of Four Average Points on Laser Beams

Now we can calculate the instant linear position of the flange plate, $P_{f1} [x_1, y_1, z_1]$, when one arbitrary point on the tool axis, P_1 is at the laser's intersections, shown in Figure 48.

The next step is to lift the end effector 20 millimeters vertically, record the displacement, and then move the object alongside another circular trajectory in the U-shape tool. The procedures above are repeated, and the system records another eight critical positions.

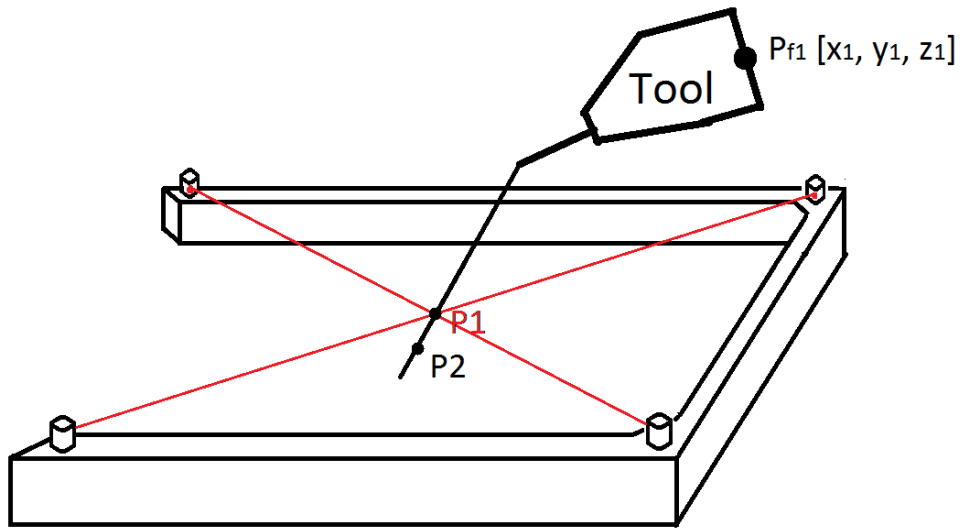


Figure 48. Flange Plate's Position When Laser Intersects With Tool Point P1

After that, repeating the calculation above, we can again get the linear position of the flange plate $P_{f2} [x_2, y_2, z_2]$, when the tool point P_2 is at the laser's intersections, shown in Figure 49.

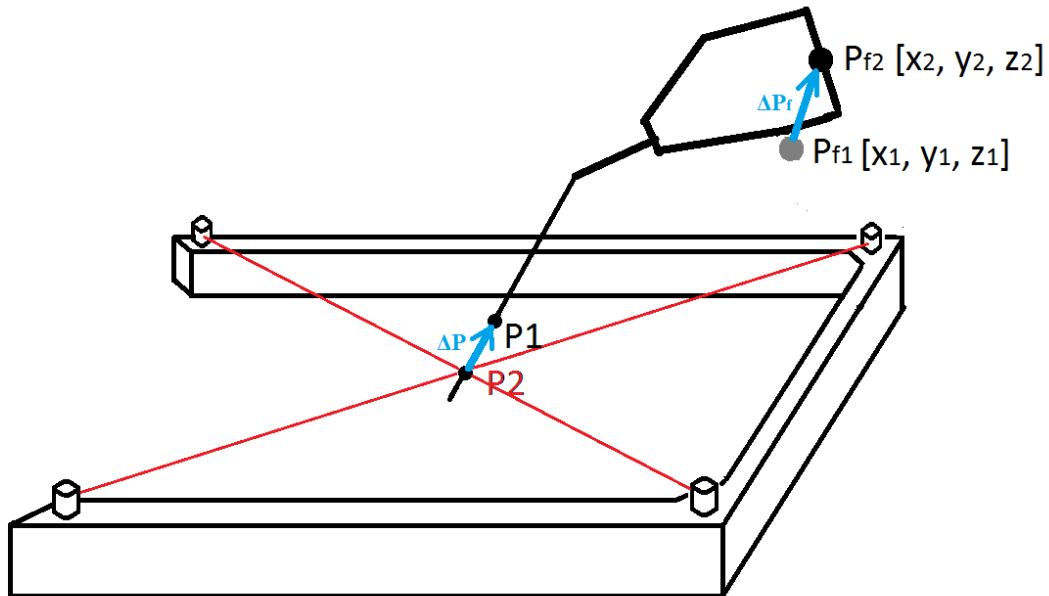


Figure 49. Flange Plate's Position When Laser Intersects With Tool Point P2

Since the action of lifting the end effector changes the tool's linear position instead of the angular orientation, the path remain the same from P_1 to P_2 , shown in vector $\Delta\mathbf{P}$, and the path from P_{f1} to P_{f2} , shown in vector $\Delta\mathbf{P}_f$. Then the orientation of the tool, in x- and y-axis can be calculated. Note that this method cannot measure the tool's coordinate system in the z-axis, and that's why the calibrator requires being coaxial and symmetric.

The next step is to find the linear positions of the tool center point, which is the end point of the tool (shown in P_3 in Figure 50). This process requires the tool to change its orientations (in $\Delta\theta_x$, $\Delta\theta_y$, and $\Delta\theta_z$ axis), and follow the circular trajectory as shown in Figure 45. Lift the end effector with the direction of the tool's orientation, which is known as $[(\theta_x + \Delta\theta_x), (\theta_y + \Delta\theta_y), (\theta_z + \Delta\theta_z)]$, until the tool center point approaches the laser intersection, shown in Figure 50. Then record the linear position and the angular orientation of the flange plate.

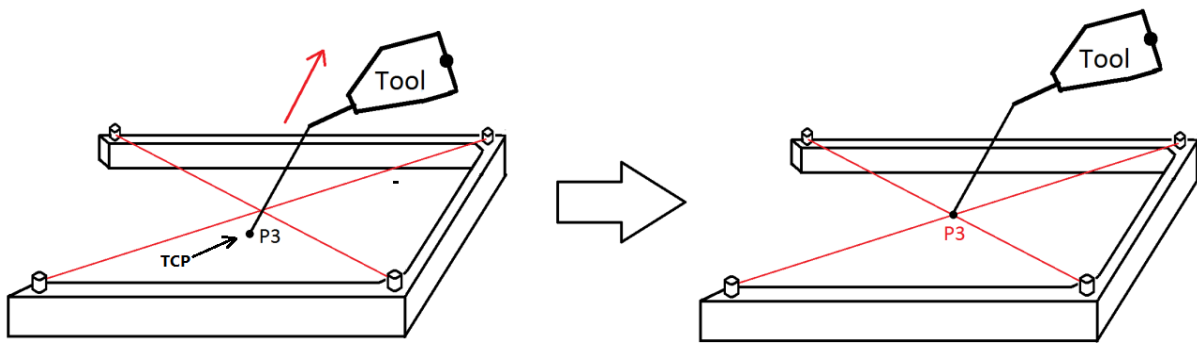


Figure 50. Finding the Tool Center Point

The last step is to repeat the procedure four times, then use the Four-point method (shown in section 4.3.1.4 Four-point Algorithm) to get the linear position of the TCP, shown in Figure 51. Note that the coordinates of the laser beams' intersection can be calculated at the same time. The outcome data is used to calibrate the coordinates of the work object (for more details, see 4.4.2 Object Measurement and Calibration).

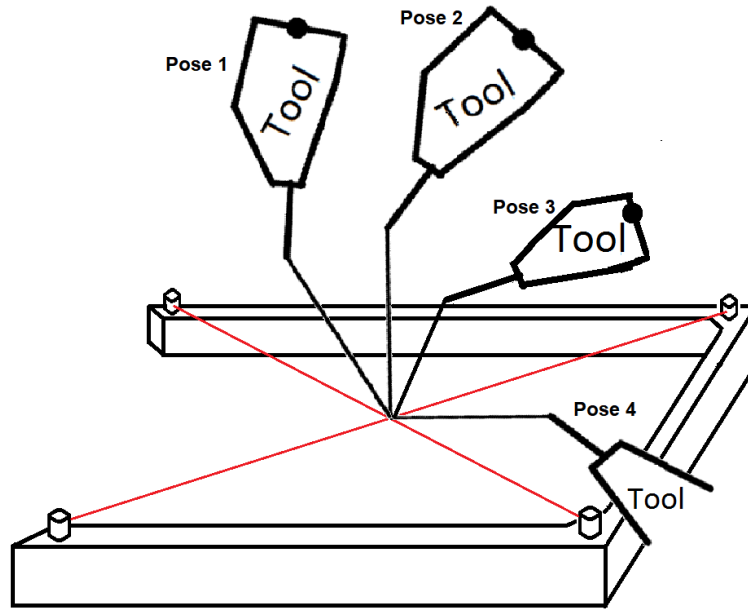


Figure 51. Tool Center Point Calculation

4.4.1.3 Measurement Tool Adjustment

To get accurate measurements, two pairs of the laser units should be adjusted until two diagonal laser beams (labeled in red and green in Figure 52 (a)) are co-planar. Therefore, we need to minimize the offset (marked in black in Figure 52 (b)).

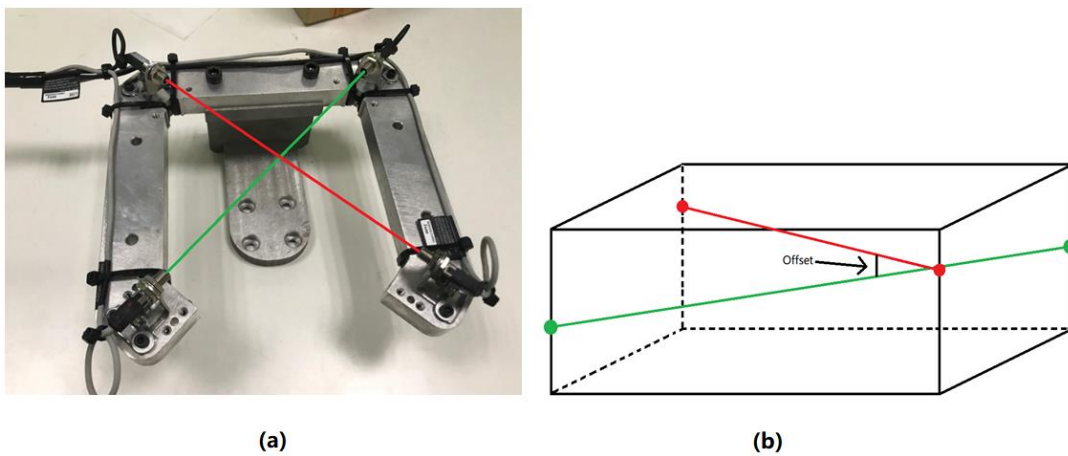


Figure 52. Laser Beams That Have a Small Offset

A small program is implemented to measure the offset. The procedures are simple: the first step is shown in Figure 53. The robot arm with the U-shaped measurement tool calibrator approaches the inside of the U-shaped measurement tool. Then it follows a circular trajectory and calculates each intersection's position in x- and y-axis, labeled as $[x_n, y_n]$. Totally there are four intersections. Technically, this step is the same as the first step in 4.4.1.2 Theoretical Method.

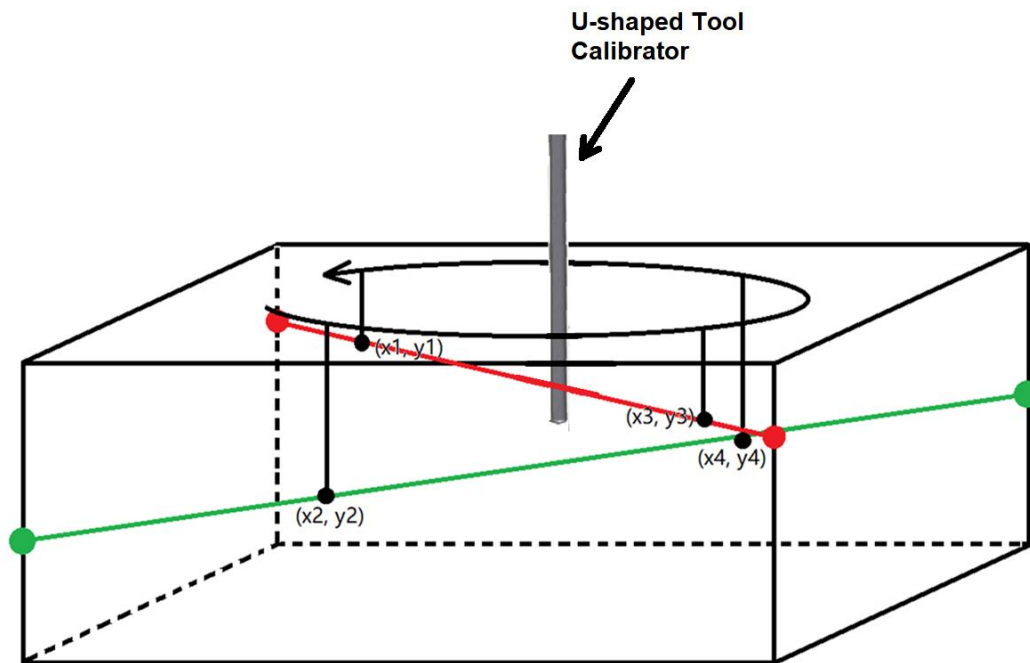


Figure 53. The Calibrator Follows The Circular Trajectory

The next step is shown in Figure 54. Calculate z-axis of each intersection by approaching each intersection point, then vertically lift the U-shaped tool calibrator until it unblocks the laser. At the moment the laser is unblocked, record the z-axis value.

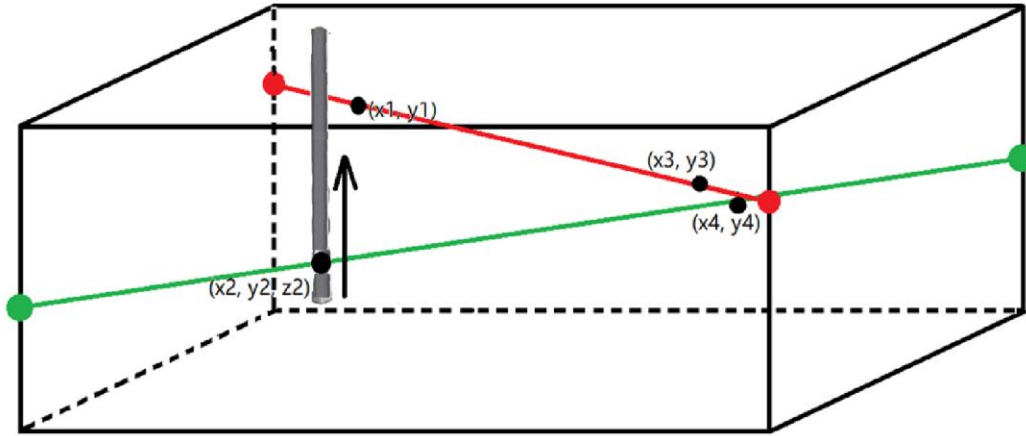


Figure 54. The Calibrator Is Lifted In Z-axis direction

After knowing the four point's positions, the program will calculate the vector equation for each laser beam. Then the shortest distance between two vectors is the offset.

Based on the calculated offset, we can slightly adjust the position by tuning the nuts and then repeat the procedures above until we get minimized offset: 40 micrometers. Since the nuts tuning is made manually, 40 micrometers are the closest error range that we can approach.

This method is mundane and time-consuming, but the adjustment should only be considered as part of the set-up step, instead of the calibration step. Therefore, the drawback could be negligible.

4.4.1.4 Four-point Algorithm

The Four-point algorithm is also implemented as U-shaped measurement tool calibration, which has been stated in Section 4.3.1.4 Four-point Algorithm.

4.4.1.5 Recovery

In this section, a recovery method is used when the robot using the U-shaped sensor calibrator approaches to the fixed tip point with various poses. Figure 55 shows a perfect situation during the calibration. There are three vectors. Vector 1 is the center axis of the tool.

Vector 2 represents one of the laser beams, and Vector 3 represents another laser beam vector that is perpendicular to the graph. In the perfect situation, Vector 2 and Vector 3 are in the same plane, which means that they have a cross section point. At a critical position, the tool blocks two laser beams at the same time.

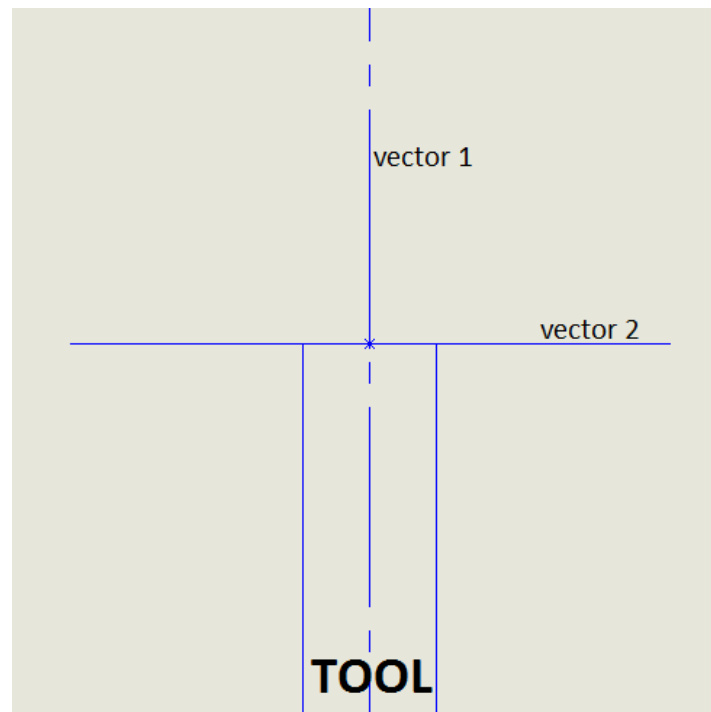


Figure 55. Perfect situation In Calibration: Vector 1 & 2 Are Vertical

But in the real-world application, the tool's position cannot be the perfect normal of a plane constituted by two laser beams. The tool's orientation has to change due to the calibration method requirements when it approaches the point with different poses. In Figure 56, we come up with a way to have a recovered displacement (labeled in red), at the center point. The tool is placed with an Angle 1 to the horizontal laser beam. There are 3 vectors as we discussed before. We can get the expressions for Vector 1 and Vector 2 as we discussed in 4.4.1.2 Theoretical Method and 4.4.1.3 Measurement Tool Adjustment respectively. We could get Vector 2 by using the method mentioned in Measurement Tool Adjustment (see 4.4.1.3 Measurement Tool

Adjustment). Knowing the radius of cylinder calibration tool, we can find the magnitude of the recovered displacement using Equation (12).

$$Recovery = \cos(\text{Angle } 1) \times \text{Radius} \quad (12)$$

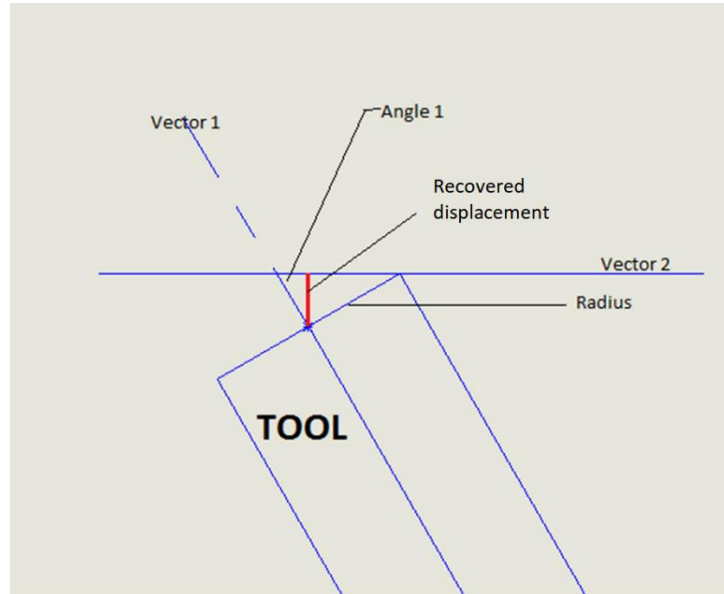


Figure 56. Real Situation: Vector 1 & 2 Are Not Vertical

Before determined the recovered displacement, the calibrated center point of each pose is not at the same point (seeing Figure 57). Points A, B, and C represent the different calibrated center points. After we have calculated the recovered displacement, points A, B, and C would at the same positions, shown in Figure 58. This would improve the accuracy of our calibration.

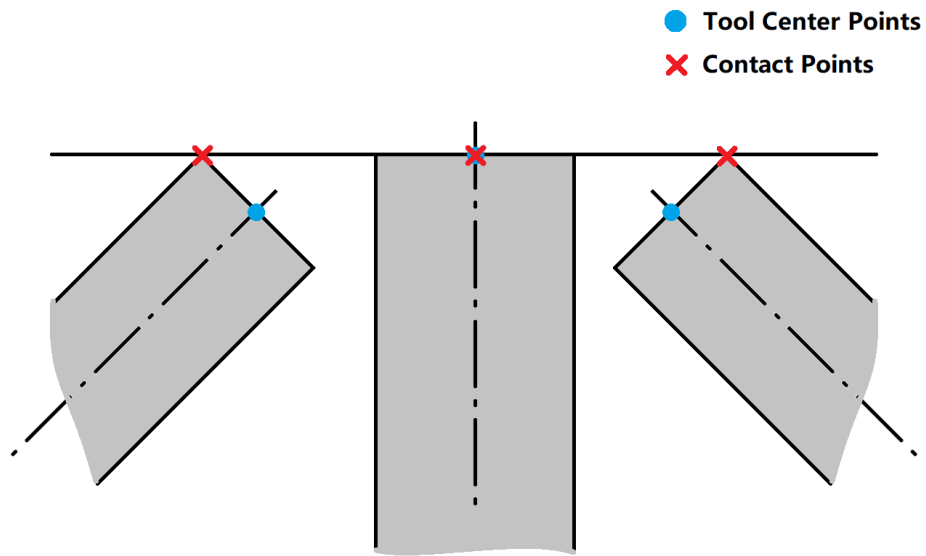


Figure 57. Condition before recovery: TCP might be wrongly recognized

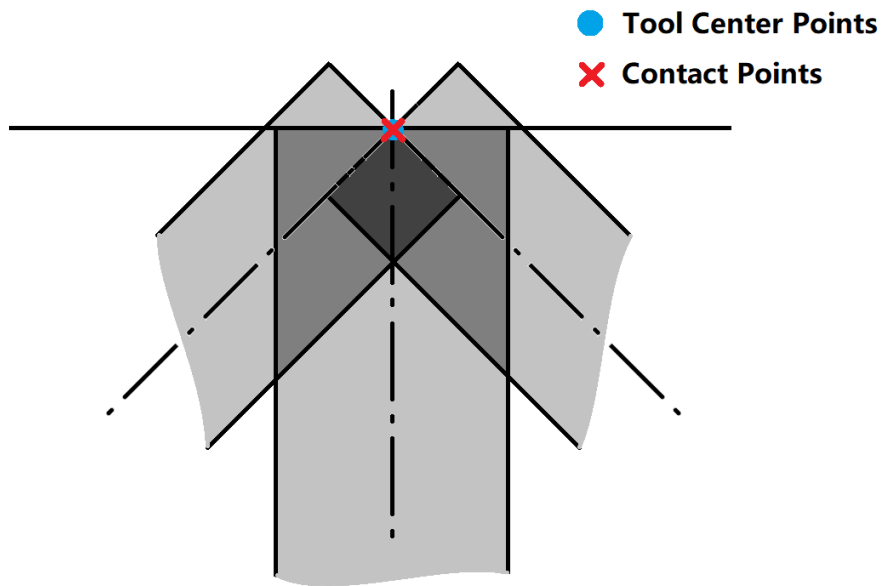


Figure 58. Condition after recovery

4.4.1.6 Source of Errors

In the calibration of the U-shape sensor process, there are several factors that might cause the errors in our result. The major factors include the 1) response time of the sensor, 2) hysteresis error in the instrument, 3) the communication delay.

a) Response time for the laser

The response time listed on a specification sheet for each of the laser sensors is 250 microseconds. Hysteresis error of the instrument can be measured by using the tool to approach in different directions: the result is 50-micrometer differences. The communication delay will be influenced by the length of the cable and the length of the package data. The length of the cable is less than 3 meters and the signal is 1-bit digital. As we measured based on the situation, the communication delay is less than 50 microseconds. In our application, the robot arm will move in a speed range from 300mm/s to 500mm/s. The amount of offset caused by the laser response latency and the communication delay largely depends on the speed of the robot movement. When an object moves across the U-shaped tool, the minor offsets occur. They are generated towards the moving direction. The offset will be discussed in detail below.

b) How error in the result occurs due to the offset

As we shown in 4.4.1.2 Theoretical Method, the U-shaped tool calibrator follows a circular path in a random position within the U-shape area. The sensor records the robot's pose each time when the calibrator blocks and unblocks the laser beam, as shown in Figure 59.

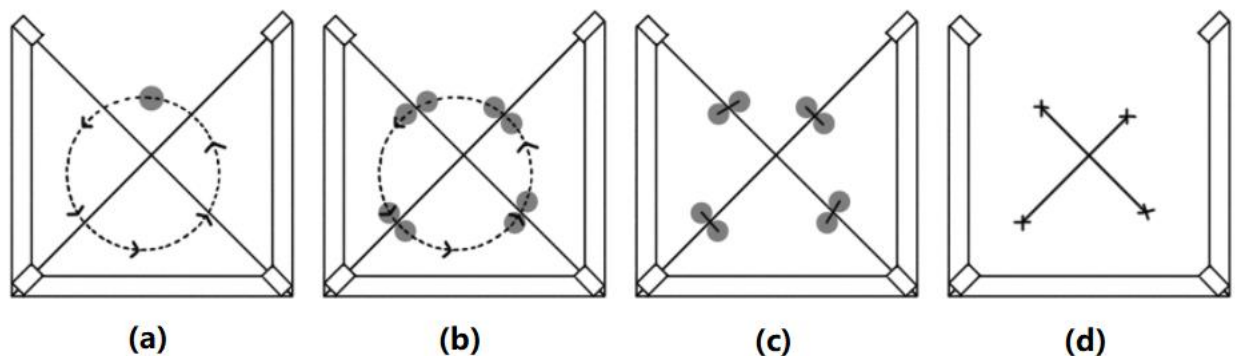


Figure 59. Theoretical Model For Calibrating U-shape Sensor

However, as the rod moves counter-clockwise in the real-world application, the response time for the laser and the other factors discussed in part a) will cause offset for each collected data, as shown in Figure 43(b). Thus, resulting in the inaccurate calculation result of the point on the laser as shown in Figure 43(c). Therefore, causing the error in the calculation of the final laser beam interaction.

In the simulation, if the offset is 2 millimeters and the center of the circle path is 20 millimeters away from the cross beam, the result is 1.9 millimeters away from the actual cross beam position. To help with this problem, we develop an optimized solution.

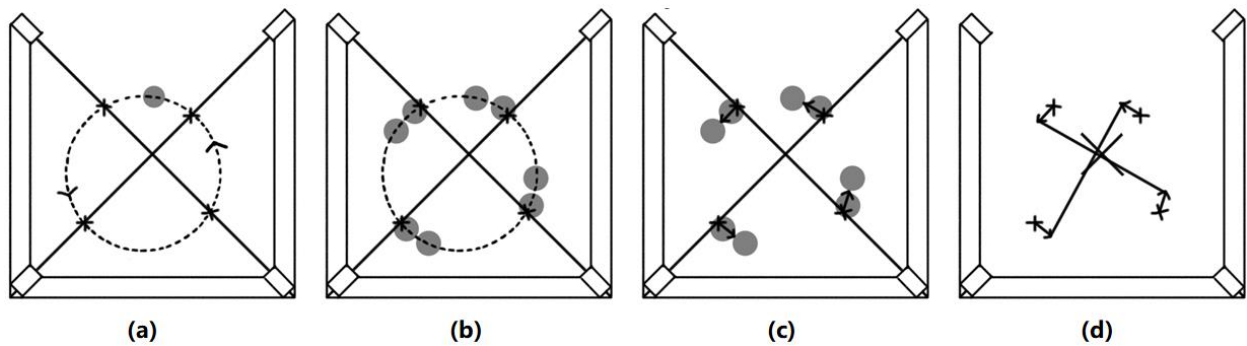


Figure 60. Compare Models For Calibrating The U-shaped Sensor When Offsets Occur

c) Optimized Solution

From Figure 61, we can check that the resulting error is caused by the deviation from the center of the circular trajectory to the intersection of the laser beams. Although the deviation is unavoidable, the center of the circular trajectory can be controlled to be minimized. The optimized solution is to approach the center of the circular trajectory as close as the intersection of the laser beams as shown in Figure 61 (a).

Theoretically, if the center of the circle path is the same position as the intersection of the laser beams and the offset is constant when collecting the data, the result would have no error, as

shown in Figure 61 (d). In the simulation, if the offset is 2 millimeters and the center of the circle path is 2 millimeters away from the cross beam, the result will have a 0.5 millimeters error.

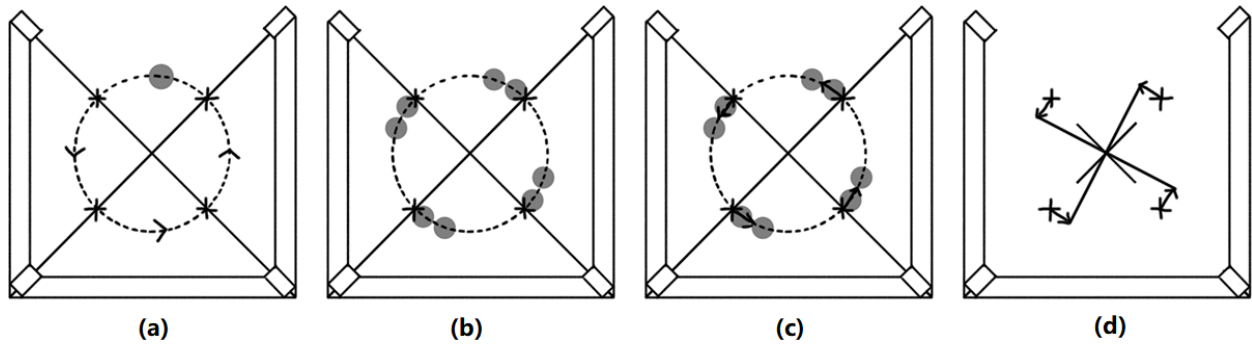


Figure 61. Optimized Model for Calibrating the U-shape Sensor

So, when performing the calibration of the U-shape sensor, the rod will follow a random circular path and try to find the position of the crossed beams. Then the rod will do the calibration again only this time it follows a circular trajectory that has a center at the previous result. Although this method does not reduce the offset, it largely reduces the error in the result by minimizing the distance between the center of the circle path and the crossed beams.

4.4.1.7 Program Design

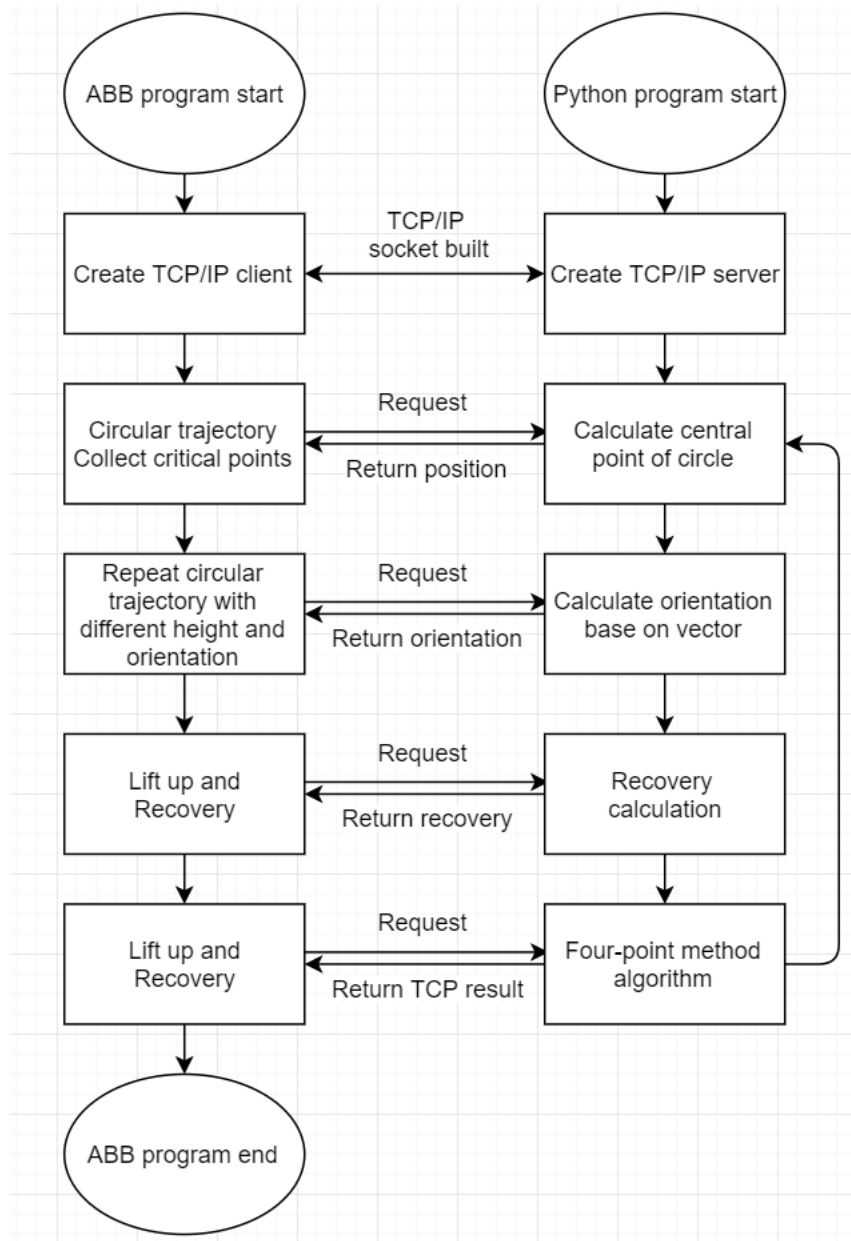


Figure 62. Flowchart of The U-Shape Measurement Tool Calibration

The program design is divided into two parts: ABB program and Python program. The purposes of these two programs are same as Section 4.3.1.6. These two programs are connected by TCP/IP socket mentioned in Section 4.2.3 Communication. Figure 62 is the flowchart of both

programs. In the U-Shape measurement tool calibration, the ABB controller is a client and the Raspberry Pi is a server. They are connected at the beginning of the program.

When the communication is built, the Python program runs as a state machine server. Each state can be treated as a mathematical operation. If the ABB program needs a mathematical operation, it will request the Python program to do the calculation, sending all the inputs to the Python program through a TCP/IP socket. After receiving a request from the ABB controller, the Python program goes to a certain switch case based on different requests. For example, when the switch case equals 1, the program starts to calculate the central point of a circle based on the input three edge points. When the switch case equals 2, the program starts to calculate an orientation based on the input vector. When the switch case equals 3, the program starts to calculate recovery based on the input vectors and the radius. When the switch case equals 4, the program starts to calculate the TCP result using the Four-point method algorithm. Since the Raspberry Pi is a server, the Python program will not stop until the keyboard interrupt.

On the other hand, the ABB program follows a specific procedure according to the theoretical method. After communication is established, it will follow the circular trajectory and collect the critical points by the I/O interrupt. Then, repeat the circular trajectory with different heights and orientations. The center points and the orientations of an object are calculated from the Python program. After that, the ABB program starts to lift and do a recovery to get all the inputs for the Four-point method algorithm. The Python program calculates the Four-point method algorithm and transfers the result to the ABB program. Finally, the calibrated results are updated to the ABB internal program data.

4.4.2 Object Measurement and Calibration

4.4.2.1. Overview

The purpose of this section is to give the explanation on how to calibrate a coaxial and symmetric shape object using U-shaped measurement tool (USMT). Specifically, we are going to discuss the method of getting the relative coordinate system between the object and the ABB base. Section 4.4.1 Measurement Tool Calibration is the part of the preparation before calibrating the object. From this section, we can get the USMT's coordinate system relative to the ABB base. The USMT's position and orientation cannot be changed while calibrating the object. Otherwise, recalibration of the USMT is necessary.

The principle of using USMT to calibrate the object and using the cylinder shape calibrator (CSC) to calibrate the USMT is pretty much the same. They both require the calibrator/object to follow a circular path within the detection range of the USMT. Many algorithms and solutions discussed in 4.4.1 Measurement Tool Calibration are also valid in Section 4.4.2 Object Measurement and Calibration.

The major differences are between their purposes. Using the CSC to calibrate the USMT is part of the preparation process after USMT is mounted. Its calibration result will affect all the aspects performance of the calibration of the object later. So, we want to have the most accurate result from the calibration of the USMT. On the other hand, the object calibration will need to be performed every time for different objects, but only one calibration of the USMT is required to work. So, we would like this process to be faster and more efficient.

The other difference is that we don't use the Four-point method (discussed in section 4.3.1.4 Four-point Algorithm) to calibrate the object. The Four-point method requires a recovery solution (discussed in section 4.3.1.5) as an optimization solution. The recovery solution needs to know the radius of the CSC/object in order to work. In the process of calibrating the USMT, we have a standard CSC with a known radius. However, the radius of the various objects is unknown. Without knowing the radius of the object, we cannot recover the differences which could significantly influence our result. Therefore, we decided not to use the Four-point method

and its recovery solution in the calibration of the object. Also, the performance of the Four-point method will take about 30 to 40 seconds. By-passing it will speed up our calibration.

4.4.2.2 Theoretical Method

In this part, the method we implemented for the object calibration is going to be introduced. We could either have the object held by the end effector to calibrate or have the USMT held by the end effector to calibrate. Both situations depend on the condition of the work object. For example, the object may not be able to pick up due to some reasons, such as its size or weight. Here, we will introduce the object held situation first.

This particular method of calibration is designed for coaxial and symmetrically-shaped objects, which perhaps would be held by the end effector mounted on the end point of the ABB robot arm. In this case, the calibration of the USMT would give us the world position of the cross point of two laser beams. It is going to be used in Equation (13) as Part D.

First of all, the object would go around inside the U-shaped calibration tool for one round. Then lift up the tool for 20 millimeters and go for another round, and record the point of the flange plate. The details for this part are stated in Section 4.4.1.3 Measurement Tool Adjustment. By doing so, the orientation centroid of the object can be calculated as we stated in the same section.

Second, based on the orientation, the robot arm position would be adjusted so that the centroid of the object will be perpendicular to the laser cross beams shown in Figure 63. At this position, the object is in the perfect situation we have discussed in Section 4.4.1.5 Recovery. The advantage to having this way of calibration is that we do not have to deal with the recovery distance that we discussed in the recovery section. Since the radius or diameter of the object is unknown, the recovery distance cannot be calculated very accurately by using the recovery method. On the other hand, the Four-point method would take a lot of time to calibrate. By calibrating in this way, we would go to the U-shaped calibration tool for at least four rounds. The number of rounds could be more depending on the situation and the requirements.

Then, the tool is going to go one more round inside the U-shaped calibration tool with the height of the first round. In this round, using the method in Section 4.4.1.3 Measurement Tool Adjustment, the centroid of the tool will, at the cross point of the laser beams, be at the height of the first round.

Third, based on the plane of the cross point from Section 4.4.1 Measurement Tool Calibration and the orientation from Step 1 of the centroid, the object will move up along its centroid as shown in Figure 64. At the moment the laser does not detect anything, the end point of the object is at the cross point of the laser beams. The robot would record the position of the flange plate, represented as Part C in Equation (13), and the orientation of the flange plate, represented as Part A in Equation (13). Because Part A, C, and D are all known from previous steps, by using Equation (13), we can calculate the tool end point position (the result), with respect to Part B. This calibration will give us the object end point position and orientation with respect to the flange plate, which is exactly what we wanted.

$$\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \cdot \begin{bmatrix} {}^E P_{Tx} \\ {}^E P_{Ty} \\ {}^E P_{Tz} \end{bmatrix} + \begin{bmatrix} {}^B P_{Ex} \\ {}^B P_{Ey} \\ {}^B P_{Ez} \end{bmatrix} = \begin{bmatrix} {}^B P_{Tx} \\ {}^B P_{Ty} \\ {}^B P_{Tz} \end{bmatrix} \quad (13)$$

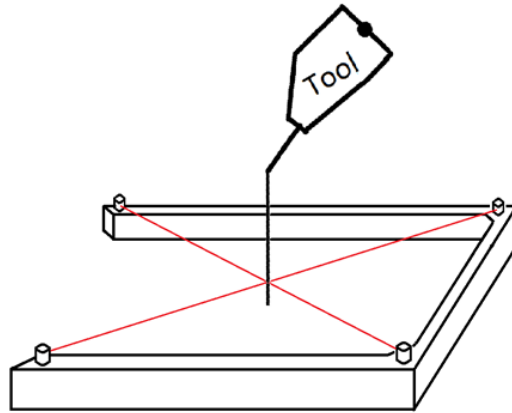


Figure 63. Tool That Is Perpendicular to The Laser Beams

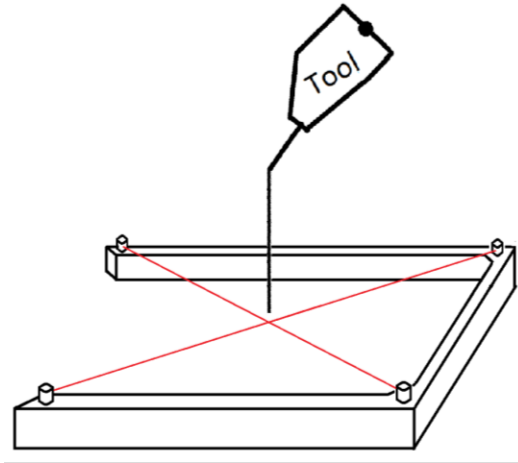


Figure 64. The Tool Moves Up

For the situation where the U-shaped calibration tool is held by the robot, the concepts and the procedures are basically the same. The difference is the variable matrix that we need to get is different. Part A and Part C remain the same as we discussed before for this situation. Part B could be calculated from Section 4.4.1 Measurement Tool Calibration, the world position of the cross beams in the calibration of USMT. This time, Part D is the calculated tool end point position (the result) using Equation (13). This calibration will give us the object end point position and orientation with respect to the world.

4.4.2.3 Source of Errors

When using the U-shaped calibration tool, there are several possible offsets that may cause an error in our result. The calibration process for the USMT and the work object are similar. They both require a cylinder shape calibrator/work object to follow an arbitrary path within the area of the U-shape calibration tool. Thus, the analysis of these offsets and our solutions are the same when calibrating the USMT. For more details, please refer to Section 4.4.1.6 Source of Error.

4.4.2.4 Program Design

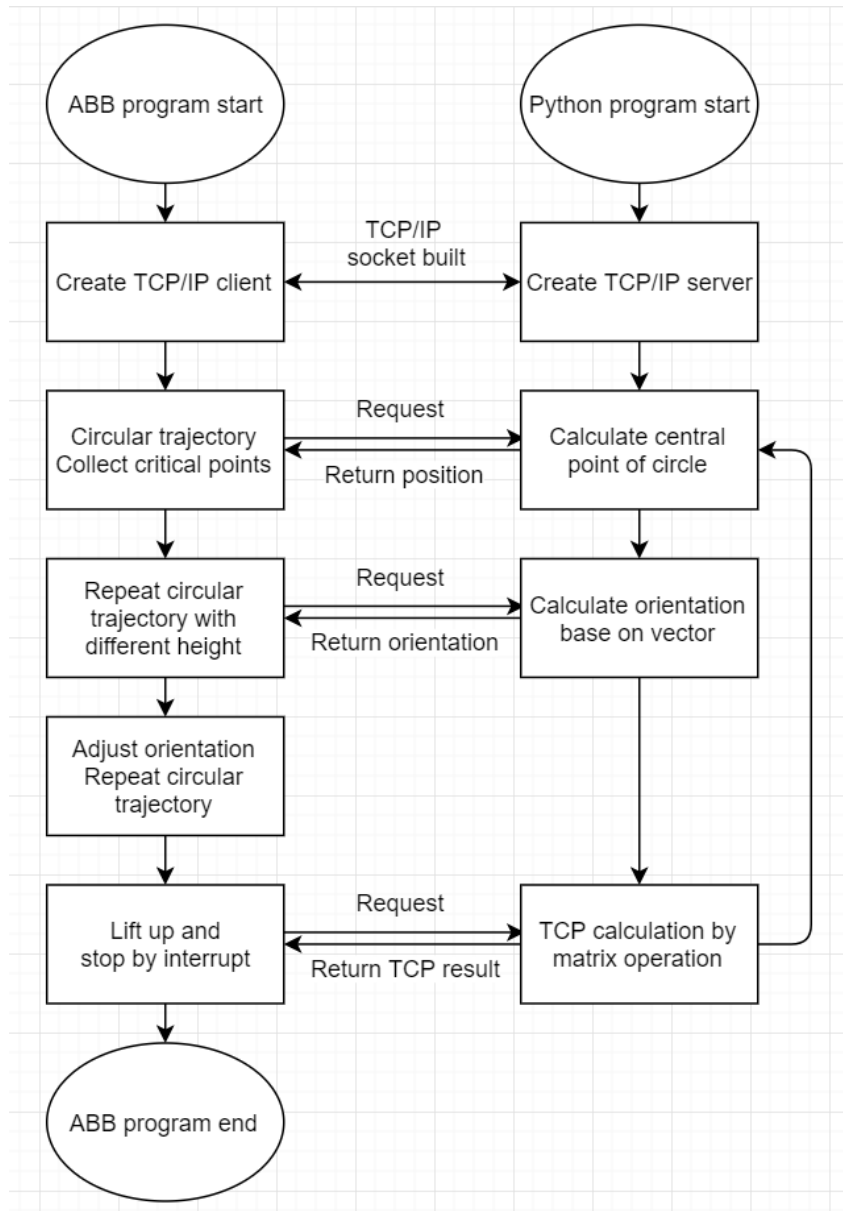


Figure 65. Flowchart of Object Calibration Using U-shaped Sensor

The program design of the U-Shaped object calibration is similar to the one of the U-Shaped measurement tool calibration in Section 4.4.1.7 Program Design. The flowchart of that is shown in Figure 65. The difference in the Python program design is using the TCP calculation by

the matrix operation instead of the Four-point method algorithm. The difference in the ABB program design is that the robot should adjust the pose perfectly to repeat the circular trajectory. In general, the program design of the U-Shaped object calibration is simpler than the one of the U-Shaped measurement tool calibration in Section 4.4.1.7 Program Design.

Chapter V: Product Performance Evaluation

In this chapter, the performance of our calibration system is evaluated. We looked at both physical components and algorithm. For the algorithm evaluation, we used the simulation to compare our Four-point Method algorithm with ABB's algorithm as shown in section next section. For the calibration method evaluation, we used a dial indicator to test the calibration precision in the experiment as shown in Section 5.2 U-Shaped Sensor Calibration Evaluation. The concepts, including error from the calculated TCP, accuracy of the TCP, repeatability of calibration result, and precision of calibration, will be presented to explain the evaluation results.

5.1 Four-point Method Evaluation

5.1.1 Overview

The Four-point method is used in both section 4.3.1 Measurement Tool Calibration and section 4.4.1 Measurement Tool Calibration to get the Tool Center Point. It is the only method to get the spatial position based on the robot base coordinates or the flange plant coordinates. Both the U-shaped measurement tool and the laser range sensor measurement tool are calibrated by the Four-point method fundamentally. Therefore, the performance of the Four-point method algorithm directly affects the performances of these measurement tools in the work object calibration. Evaluating the performance of the Four-point method algorithm can help us optimize the algorithm itself and improve the accuracy of the calibration in the end. On the other hand, we create our own Four-point method algorithm that is different from the ABB's algorithm. By comparing the performances of our algorithm with the ABB's algorithm, we can understand our algorithm better. In the following section, the performance of our Four-point method algorithm and the ABB's algorithm will be analyzed by the error from the calculated TCP and the accuracy of the TCP. Those concepts are introduced before evaluating our Four-point method algorithm.

5.1.2 Error from the calculated TCP

The Equation (14) is the general equation of the TCP, which is the same as Equation (5) in section 4.3.1.3. ${}^B P_{Tx}$, ${}^B P_{Ty}$, ${}^B P_{Tz}$ are the positions of the fixed tip point related to the base coordinate system. When the TCP has been calculated by the Four-point method algorithm, the robot pose and the TCP on the left side of Equation (14) are known. The input of the Four-point method is not perfect for the calibration. It means that the robot with a tool cannot approach the exact position of one fixed tip point, so the value of ${}^B P_{Tx}$, ${}^B P_{Ty}$, ${}^B P_{Tz}$ are different in each pose using the calculated TCP in Equation (14).

$$\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \cdot \begin{bmatrix} {}^E P_{Tx} \\ {}^E P_{Ty} \\ {}^E P_{Tz} \end{bmatrix} + \begin{bmatrix} {}^B P_{Ex} \\ {}^B P_{Ey} \\ {}^B P_{Ez} \end{bmatrix} = \begin{bmatrix} {}^B P_{Tx} \\ {}^B P_{Ty} \\ {}^B P_{Tz} \end{bmatrix} \quad (14)$$

An example is shown in Figure 66. The position of P1 can be calculated by Equation (14). The positions of P2, P3, and P4 will be calculated using the same way.

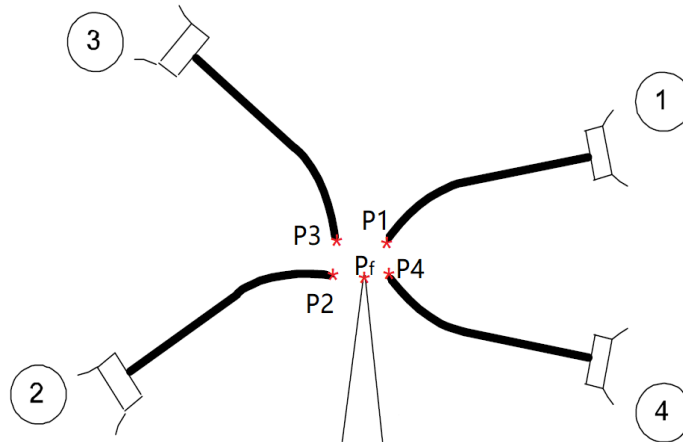


Figure 66. The Position Errors from Four-point Method Algorithm (ABB Robotics, RAPID Instructions, Functions and Data types Technical reference manual, 2013)

Then the position vector \mathbf{P}_f can be calculated by averaging the position vectors of \mathbf{P}_1 , \mathbf{P}_2 , \mathbf{P}_3 , and \mathbf{P}_4 shown in Equation (15). P_f is the fixed tip point in Figure 66.

$$\mathbf{P}_f = \frac{1}{n} \left(\sum_i^n \mathbf{P}_i \right) = \frac{\mathbf{P}_1 + \mathbf{P}_2 + \dots + \mathbf{P}_n}{n} \quad (15)$$

After getting the position vectors of \mathbf{P}_f , the distance d_i between P_f and P_i can be calculated by Equation (16).

$$d_i = \sqrt{(P_{fx} - P_{ix})^2 + (P_{fy} - P_{iy})^2 + (P_{fz} - P_{iz})^2} \quad (16)$$

After getting the distances between P_f and each point P_i , the mean error from the calculated TCP is the mean value of these distances as shown in Equation (17). The maximum error from the calculated TCP is the maximum value of these distances as shown in Equation (18). Generally speaking, the position of each approached point should be one exact point in the perfect situation, and the error from the calculated TCP is from these approached points. The error from the calculated TCP can be determined in each Four-point method calculation. It is one way to know the performance of calibration because errors of the input data of Four-point method may cause the obvious differences of positions of the approached points. For example, when the robot with tool approaches the fixed tip point by jogging manually, the mean error is often about 0.4mm.

$$\text{Mean error from the calculated TCP} = \frac{1}{n} \left(\sum_i^n d_i \right) = \frac{d_1 + d_2 + \dots + d_n}{n} \quad (17)$$

$$\text{Maximum error from the calculated TCP} = \max (d_1, d_2, \dots, d_n) \quad (18)$$

5.1.3. Accuracy of the TCP

The accuracy of the TCP is the difference between the position of the calculated TCP and the position of the correct TCP. The accuracy of the TCP is a more direct way to understand the performance of Four-point method than the error from the calculated TCP because the goal of the Four-point method is to find the most suitable TCP. It is hard to determine the accuracy of the TCP in the real calibration application because there is no way to get the position of real TCP. However, we can assume a correct TCP in the theoretical model and analyze the accuracy of the TCP in the software simulation.

5.1.4 ABB Four-point Method Algorithm Versus Our Algorithm

In this section, the performances of our Four-point methods algorithms and ABB's algorithm are analyzed by the error from the calculated TCP test and the accuracy of the TCP test. There are several steps to get the result of accuracy and error from the calculated TCP seen in Figure 67.

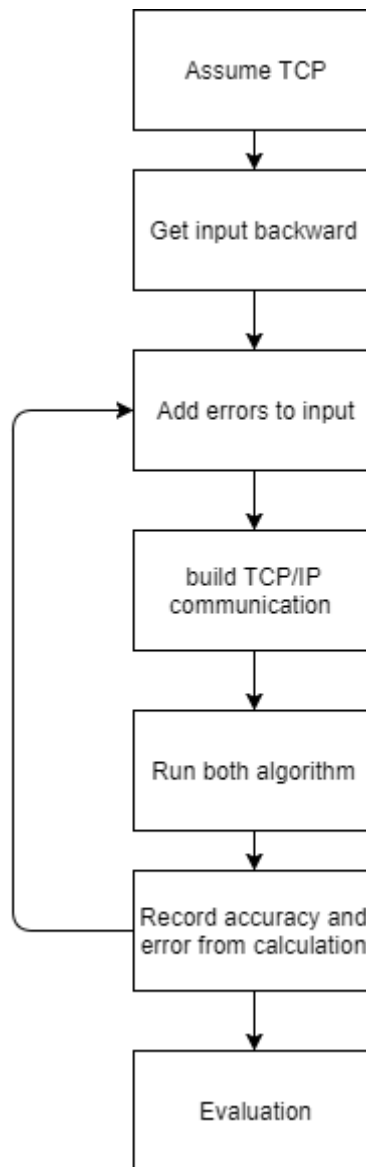


Figure 67. Flowchart of Error and Accuracy Evaluation

First, the TCP is assumed to be [10, -20, 40], which means that the TCP is located at point 40 millimeters straight out from the mounting flange, 10 millimeters along the x-axis of the wrist coordinate system, and -20 millimeters along the y-axis of the wrist coordinate system.

Second, the information of each pose can be obtained backward by jogging the robot in the tool orientation mode. In each pose, the data includes the position (in x-, y-, and z-axis) of the flange plate and the orientation of the tool, expressed in Quaternion form (q1, q2, q3, and q4). This data is the input of the Four-point method algorithm. Table 5 is an example of four samples.

Third, to simulate the arbitrary errors in a certain range, we create a program that generates the random error using the Gaussian distribution with certain standard deviation, as shown in Figure 68. Then we added the random error into the position and orientation matrix (the quaternions are transformed to Euler angle first).

Fourth, we build a TCP/IP socket communication between the ABB program and our Python program. We call the ABB interior Four-point method function. Then we create 5,000 samples of different robot pose sets, and each of them includes four robot poses with a generated error. The structure of one sample set is shown in Figure 69. We use the same sample sets to run our Four-point method program and the ABB's interior Four-point method program. After that, the results are recorded, including input dataset, accuracy, and mean error from the calculated TCP. In the end, we repeated the procedures above with the different standard deviations of error.

Table 5. Input Data of Four-point Method

Pose	Input data
P1	[882.012, -548.926,727.000], [0.552115,0.551683, -0.543629, -0.308678]
P2	[889.155, -556.921,723.499], [0.500194,0.645727, -0.252489, -0.518741]
P3	[869.674, -558.681,742.269], [0.386907,0.832488, -0.248404, -0.309132]
P4	[848.732, -537.252,730.856], [0.448262,0.679156, -0.577429,0.066217]

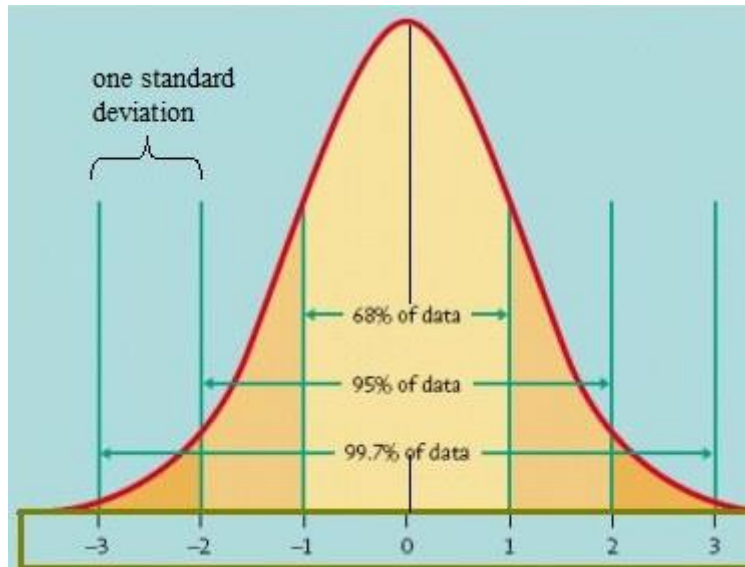


Figure 68. Normal Distribution Error Added Into Input (Statistics, 2017)

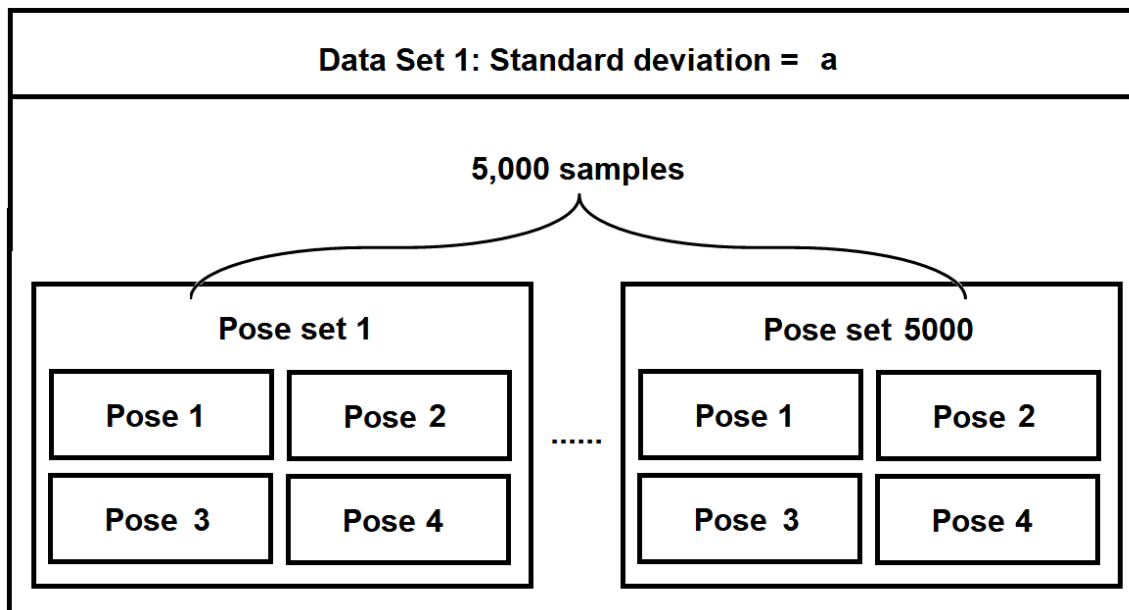


Figure 69. Structure of One Data Set

After that, we create several datasets with various standard deviation errors. Each of them includes 5,000 accuracy values and 5,000 errors from the calculated TCP values. We used each

dataset as input, and run the ABB's algorithm and our algorithm, respectively. The mean, maximum, minimum and standard deviation can be calculated from the two algorithms.

Finally, the data is moved into Matlab. Figure 70 shows the relationship between the TCP accuracy and the standard deviation of the input error. Figure 71 shows the relationship between the mean error from the calculated TCP and the standard deviation of the input error. Error bar is used to reflect the mean, maximum and minimum values in each standard deviation in Matlab. According to the tendency of the graph, both TCP accuracy and error from the calculated TCP varies linearly standard deviation of the input error. Most importantly, as the evidence from Figure 70 and Figure 71, our algorithm results have higher accuracy than those from the ABB internal program. Therefore, the performance of our Four-point method algorithm is better than that of the ABB's internal program entirely.

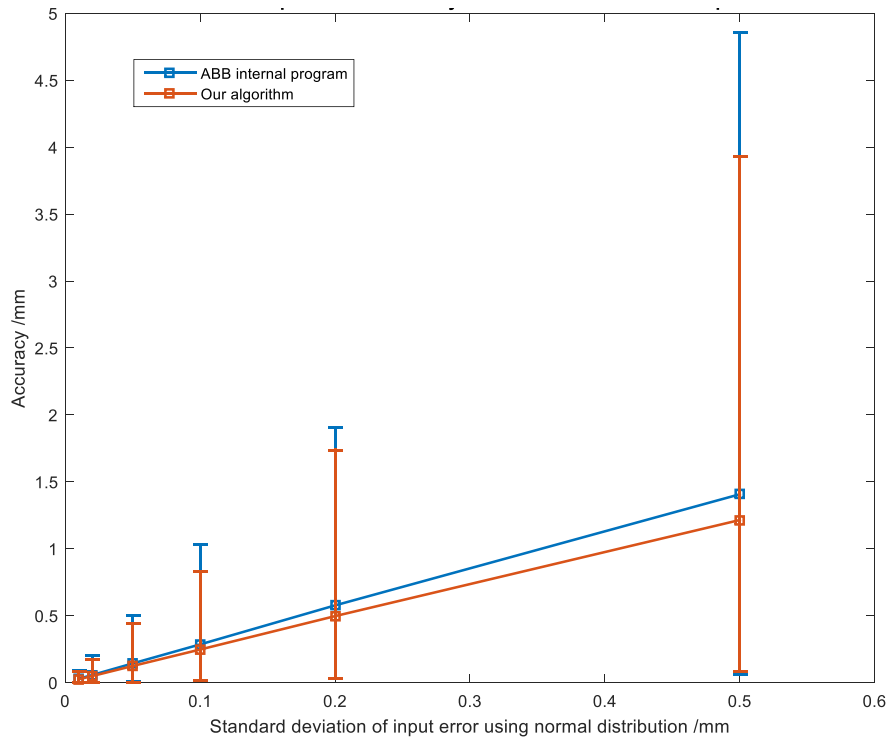


Figure 70. Relationship Between TCP Accuracy And Standard Deviation of Input Error in Simulation

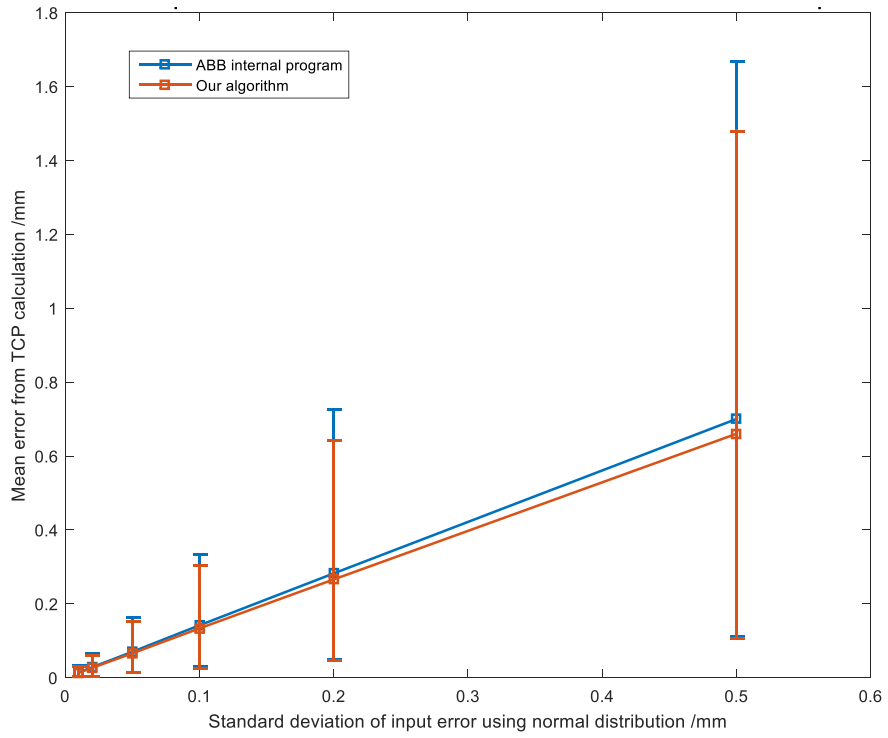


Figure 71. Relationship Between the Mean Error from the Calculated TCP And Standard Deviation of Input Error in Simulation

5.2 U-Shaped Sensor Calibration Evaluation

5.2.1 Overview

In this section, we used experimental methods to evaluate the U-shaped sensor calibration. The concept of repeatability, precision, and duration are introduced before evaluating U-Shaped sensor calibration.

5.2.2 Repeatability of the Calibration Result

We define that the repeatability of calibration result is the reproducibility of the calibration result. In the case of calibrating a steady state object multiple times, for examples, if the calibration results are close together, the results will show a high degree of repeatability. Specifically, in the experiment, there are nothing changes during each calibration. Then the results of each calibration, that the position vectors are recorded as $\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n$. Then the average position vector \mathbf{P}_f can be calculated by Equation (19).

$$\mathbf{P}_f = \frac{1}{n} \left(\sum_i^n \mathbf{P}_i \right) = \frac{\mathbf{P}_1 + \mathbf{P}_2 + \dots + \mathbf{P}_n}{n} \quad (19)$$

After getting the position vectors of \mathbf{P}_f , the repeatability of calibration, which is the result of standard deviation, can be calculated by Equation (20).

$$\text{Repeatability} = \sqrt{(\mathbf{P}_f - \mathbf{P}_1)^2 + (\mathbf{P}_f - \mathbf{P}_2)^2 + \dots + (\mathbf{P}_f - \mathbf{P}_n)^2} \quad (20)$$

The repeatability of our calibration system includes the repeat accuracy of the ABB robot, the repeat accuracy of the laser range sensor, the communication delay repeatability, and

sensor response repeatability. Some parts of them are analyzed in the source of errors in the previous sections. In the experiment, we record six sets of data. Figure 72 is the repeatability distribution of U-shaped calibration results in the experiment. The x-axis, y-axis, and z-axis in the graph are the standard deviations of x-, y-, and z-displacement. In the calculation, the standard deviation of x-displacement is 6.4 micrometers. The standard deviation of y-displacement is 4.4 micrometers. The standard deviation of z-displacement is 7.0 micrometers. Therefore, the overall repeatability of calibration result is 10.5 micrometers. It indicates that the repeatability of errors is small enough that the errors would barely impact the repeatability of the calibrated results.

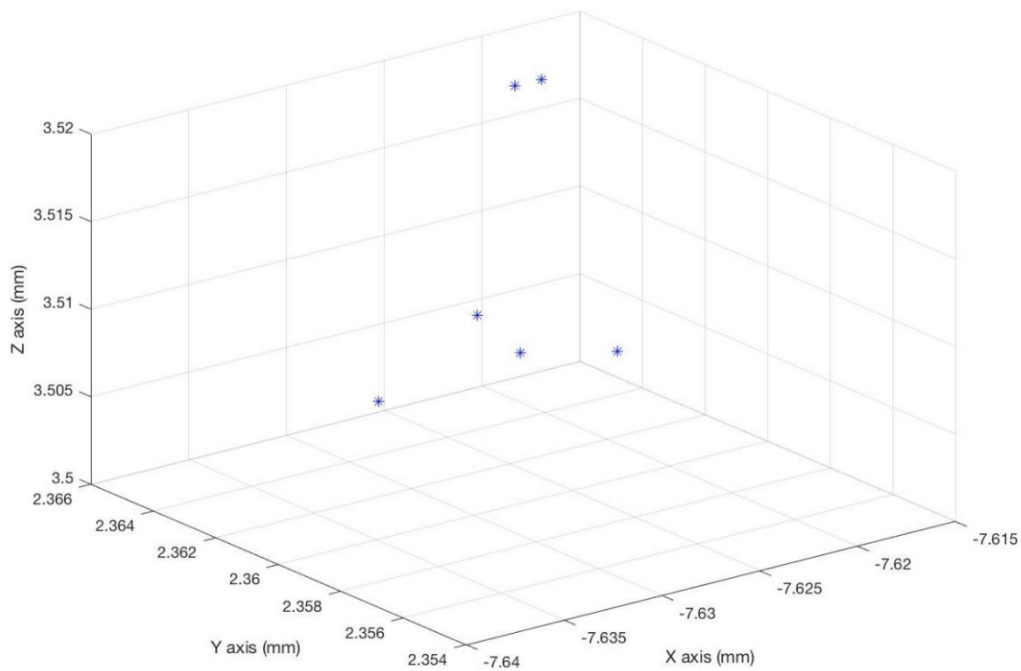


Figure 72. The Repeatability Distribution of U-Shaped Calibration Result in the Experiment

5.2.3 Precision of the Calibration

Different from the repeatability of the calibration result, we define that the precision of calibration is the repeatability of the robot moving to the same position based on the calibrated coordinate system. We used a dial indicator to test the precision of calibration in the experiment. The procedures are shown below

- 1) We program the robot to let the workpiece touch the indicator and record that position, which is based on the calibrated coordinate system so that the dial indicator has a reading, and we record the reading.
- 2) Then we randomly change the position and orientation of the object.
- 3) We calibrate the new coordinate system of the object.
- 4) We make the robot go the same position based on the calibrated coordinate system. The reading of dial indicator is recorded again.
- 5) Then we repeat the procedures from step two to step four to find next reading of dial indicator for six times.
- 6) Finally, the standard deviation of these readings is defined as the precision of calibration.

Note that multiple dial indicators can be used to record the precision of the calibration in many directions.

Figure 73 is the six sets of reading records in the experiment. We get the standard deviations of x-, y- and z-displacement are 168.5 micrometers, 141.8 micrometers, and 43.6 micrometers respectively. Overall, the standard deviation of distance is 224.5 micrometers. The result is not good enough. So, we start to analyze the sources of error from calibration. We adjust the measurement tool and make the recovery of the U-shaped measurement tool calibration mentioned in section 4.4.1 Measurement Tool Calibration. After optimizing our U-shaped calibration system, the precision of calibration is always within 0.2mm, which reaches our ultimate goal.

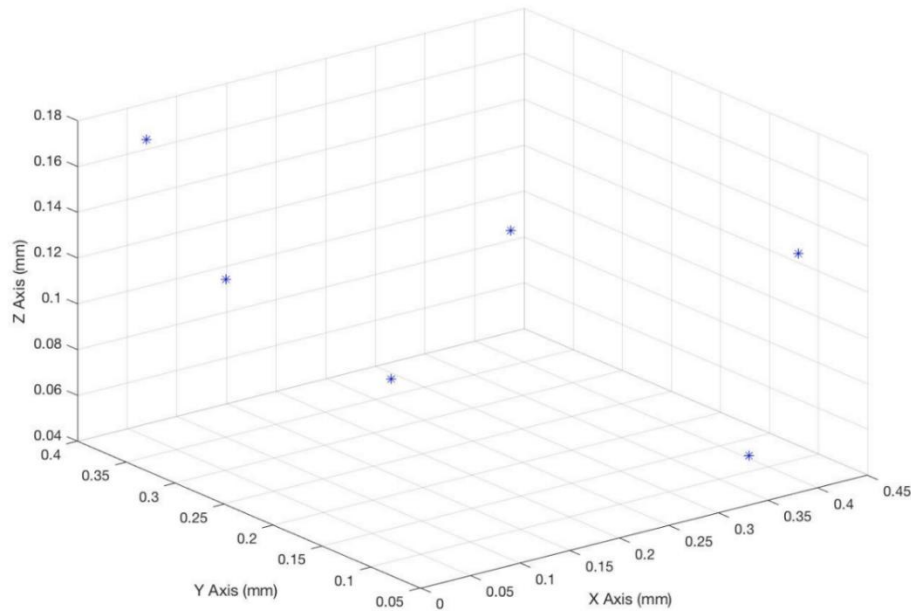


Figure 73. The Readings of Dial Indicator using U-Shaped Sensor Calibration in the Experiment

5.2.4 Duration of Entire Calibration

We define that the duration of entire calibration is the time cost from the beginning to the end of the calibration. The duration of the U-shaped sensor calibration at full speed is about 30 seconds. The duration of the U-shaped sensor calibration in manual mode is about 2-3 minutes. The duration is limited because the higher speed the robot achieve, the larger offset came from communication delay, which is analyzed in section 4.4.1.6 Source of Errors. It will impact the precision and repeatability of the calibration, so we should find a balance point between duration and other performances.

5.3 Laser Range Sensor Calibration Evaluation

5.3.1 Overview

In this section, we use experimental methods to evaluate the laser range sensor calibration. The laser range sensor calibration has two parts. The first part is about the Three-plane calibration method. The second part is the edge detection calibration method. The specific case we use for the second part is the blisk calibration. Due to the limited time and conditions, we cannot evaluate its performance.

The concept of the repeatability, precision, and duration has been introduced in the last section, so the following sections are details of our result directly.

5.3.2 Repeatability of the Calibration Result

For the Three-plane calibration method, the repeatability of the calibration result is negligible, approximately 12 micrometers. It means that the repeatability of errors are barely impacted the repeatability of the calibrated result.

5.3.3 Precision of the Calibration

For the Three-plane calibration method, the precision of the calibration result is within 150 micrometers, which is better than the performance of the U-shaped sensor calibration.

5.3.4 Duration of Entire Calibration

For the Three-plane calibration method, the duration of the entire calibration at full speed is about 10 seconds. In the manual mode, the duration of the entire calibration is about 30 seconds. The duration is limited because the robot needs to read the sensor reading while it is not moving. Otherwise, the sensor reading will be unstable due to the vibration of a robot.

For the edge detection calibration method, the duration of the entire calibration at full speed is about 8 seconds, which is the fastest out of all. In the manual mode, the duration of the entire calibration is about 20 seconds. The duration is limited because the higher speed of the robot motion, the fewer sensor readings will have while scanning the edge, which reduces the resolution of the calibrated result.

Chapter VI: Conclusion


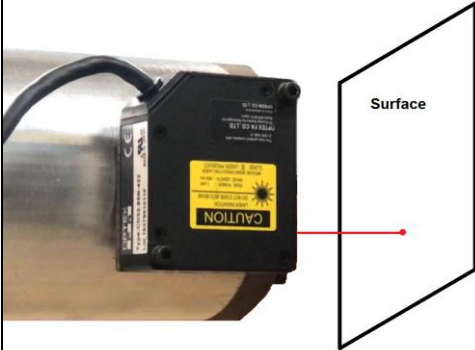
We successfully designed a new work object calibration system and built a prototype. Based on the performance of the prototype, we achieved the following requirements:

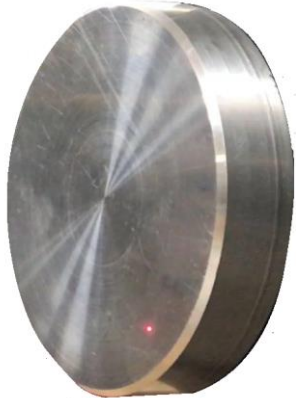
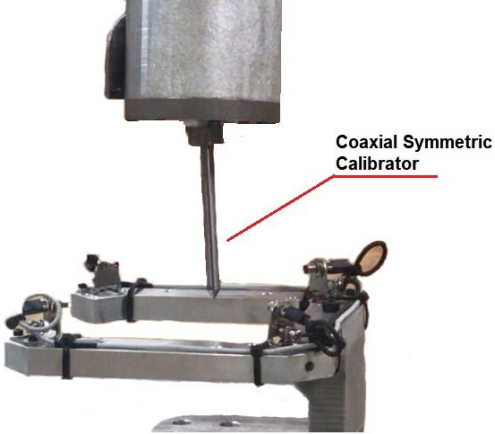
1. The system can calibrate more diverse types of the workpieces than those are available on the market, such as the application of the blisk calibration.
2. In the system, the two types of calibration methods achieve the qualified precisions, 0.2 millimeters.
 - a. The Laser Range Sensor Calibration Method can achieve the precision of 0.15 millimeters.
 - b. The U-shaped Tool Calibration Method can achieve the precision of 0.2 millimeters.
3. The runtime of the calibration system lasts a qualified duration.
 - a. The runtime of the Laser Range Sensor Calibration lasts around 10 seconds in the full-speed mode
 - b. The runtime of the U-shaped Calibration Method lasts around 30 seconds

Before achieving the goals above, we must calibrate the two measurement tools. The excellent performance of our Four-point method algorithm also leads to its success. Through the measurements and the calculations, we got the following results:

1. In the calibration of the Laser Sensor Measurement Tool, the precision is around 0.1 mm.
2. In the calibration of the U-shaped measurement tool, the precision is also around 0.1 mm.
3. The accuracy of the TCP computed by our Four-point algorithm is 10-15% better than computed by the ABB's four-point method algorithm.

Appendix I: Nomenclature Glossary

Abbreviations	Full Name	Note	Diagram
MT	Measurement Tool	General stands for the types of measurement tools in the calibration, including laser range sensors and U-shaped mount with sensor sets.	N/A
CR	Calibrator	General stands for the type of tools that are used for calibrating measurement tool, including Cylinder Shape Calibrator and Coaxial and Symmetric Shaped Calibrator.	N/A
USMT	U-shaped Measurement Tool	Our self-designed tool that consists U-shaped mount and two sets of sensor units. It is used for measuring in the work object calibration.	
LRSMT	Laser Range Sensor Measurement Tool	A laser range sensor that can measure the distance accurately. The sensor is mounted on the robot's end effector.	

CSC	Cylinder Shape Calibrator	A Cylinder Shape Calibrator is used for calibrating Laser Range Sensor Measurement Tool.	
CSSC	Coaxial and Symmetric Shaped Calibrator	A rod-shaped calibrator is used for calibrating U-shaped measurement tool.	
Process of calibrating the CT using CR	Process of calibration the Calibration Tool using Calibrator		N/A
Process of calibrating the WO using CT	Process of calibrating the Work object using calibration tool		N/A

Appendix II: Project Video Links

PDR: https://drive.google.com/open?id=0B49jk_zudCRbWFpfTGIYMGQ2Uzg
Final Demo: <https://drive.google.com/open?id=1m-AzTfdL3cMcqiFRP05Rv63tfQqWYD1e>
Blisk Calibration Demo: <https://drive.google.com/open?id=1CIRxtnUNoo9z8p5Ey3tvuBpu0ZqiRLd>

References

- ABB Inc. (2016). *ABB-Developer Center*. Retrieved from ABB Group:
<http://developercenter.robotstudio.com/pcsdk>
- ABB Inc. (n.d.). *FENA-21 Ethernet adapter module*. Retrieved from ABB Group:
<http://new.abb.com/drives/connectivity/fieldbus-connectivity/profinet/fena-21>
- ABB Inc. (n.d.). *IRB 1600 Data*. Retrieved from ABB Group:
<http://new.abb.com/products/robotics/industrial-robots/irb-1600/irb-1600-data>
- ABB Robotics. (2007). *Operating Manual RobotStudio*. Västerås, Sweden: ABB Inc.
- ABB Robotics. (2012). *PC SDK Application manual*. Västerås, Västmanland, Sweden.
- ABB Robotics. (2013, April 3). *RAPID Instructions, Functions and Data types Technical reference manual*. Västerås, Västmanland, Sweden.
- ABB Robotics. (2017). *IRB1600_PR10282EN-I.pdf*. Retrieved from ABB Inc.:
https://library.e.abb.com/public/1180c895f6b64cb8847631101c2e75ae/IRB1600_PR10282EN-I.pdf
- Bergstrom, G. (2011). *Method for calibration of off-line generated robot program*. Göteborg, Sweden: CHALMERS UNIVERSITY OF TECHNOLOGY. Retrieved from
<http://publications.lib.chalmers.se/records/fulltext/153281.pdf>
- Bonev, I. (2013, April 5). *Singularities in six-axis vertically-articulated industrial robots*. Retrieved from Control and Robotics Laboratory: <http://coro.etsmtl.ca/blog/?p=107>
- Campbell, A. T. (2012, 5 11). Retrieved 10 20, 2017, from CS 50 Software Design and Implementation: <http://www.cs.dartmouth.edu/~campbell/cs50/socketprogramming.html>
- Controller Area Network Sdn Bhd. (2013). *FAQ*. Retrieved from Cans:
https://web.archive.org/web/20080702143440/http://www.cans.com.my/modules.php?name=FAQ&myfaq=yes&id_cat=3&categories=DeviceNet
- DELFOI. (2016). Retrieved from ROBOT SIMULATION AND OFFLINE PROGRAMMING:
https://www.delfoi.com/web/solutions/robotiikka/en_GB/offline/

- Devo Engineering. (n.d.). *Production Equipment*. Retrieved from Devo Engineering:
<http://www.devoengineering.se/cms/pages/gb/references/drs---abrasive-waterjet-cutting.php>
- DynaCal Inc. (n.d.). *DynaCal - Robot Absolute Accuracy Calibration, Robot Cloning, Off-Line Programming, OLP*. Retrieved from DynaCal: Robot Calibration System:
http://www.dynalog-us.com/dynalogmainsite_030.htm
- Hao Gu, Q. L. (2015). Quick Robot Cell Calibration for Small Part Assembly.
- Keyence Inc. (2016). *Advantages and Disadvantages between Contact and Non-contact Sensors*. Retrieved from KEYENCE CORPORATION OF AMERICA: www.keyence.com
- Leoni. (n.d.). *advintec TCP – calculation and calibration of robotic tools and fixtures in up to 6 dimensions*. Retrieved from Leoni Inc.: <https://www.leoni-industrial-solutions.com/en/products-services/calibration-of-robotic-tools-fixtures/>
- National Instruments Inc. (2012, April 16). *Introduction to FPGA Technology: Top 5 Benefits*. Retrieved from National Instruments: <http://www.ni.com/white-paper/6984/en/>
- Shujun, K. C. (2016). *Study and Realization of Tool Coordinate Frame*.
- Simems AG. (2012). *Easy PROFINET implementation*. Retrieved from Siemens ProfiNet Components:
<https://web.archive.org/web/20150402183456/http://w3app.siemens.com/mcms/infocenter/dokumentencenter/sc/ic/Documentsu20Brochures/E20001-A24-M116-X-7600.pdf>
- Spasić, I. (2012, May 2). *Blisk - STEP / IGES, STL, SOLIDWORKS - 3D CAD Models - GrabCAD*. Retrieved from GrabCAD Community: <https://grabcad.com/library/blisk>
- Stallings, W. (2005). *Operating Systems, Internals and Design Principles*. Pearson.
- Statistics. (2017). *Normal Distributions: Definition, Word Problems*.
- Trompeter, M. (2012, Oct 18). *US Patent No. US20120265341*. Retrieved from
<https://www.google.ch/patents/US20120265341>

Wakefield, J. (2016, May 25). Retrieved from Foxconn replaces '60,000 factory workers with robots': <http://www.bbc.com/news/technology-36376966>