University of Manitoba

Improving Accuracy of Robotic Operations Through Detailed Robot Cell Calibration

Undergraduate Thesis

Course:	MECH 4162: Thesis
Report For:	Dr. David Kuhn, P.Eng.
Supervised By:	Dr. Matt Khoshdarregi, P.Eng.
Prepared By:	Peter Goltz
Submitted:	March 28th, 2022

Executive Summary

Robot cell calibration ensures that all areas in the robot cell, including tool pick up points and workspace tables, are calibrated with the robot. Often times, the solution to achieve higher accuracy with robot cell calibration is through manual programming which is time consuming and has limited accuracy. This thesis proposes a solution using a laser tracker to quickly calculate highly accurate workspace coordinate frames in a MATLAB algorithm to achieve a desirable robot cell calibration. The method uses 17 data points on and around the robot and calculates the coordinate frame using direction vectors and planes. Validation of this method is performed by comparing accuracy, precision and execution time to another traditional cell calibration method, the 3-point method. The results show that this method, compared to the 3-point method, excels in all three of these categories and is especially effective in environments that continually need highly accurate workspace coordinate frames.

Acknowledgements

The author would like to thank and acknowledge:

- Michael Newman for his guidance and assistance throughout the progression of this thesis.

- Dr. Matt Khoshdarregi for his guidance as thesis supervisor.

Contents

List of Figures

List of Tables

1 Introduction

This thesis will discuss a method for improving the accuracy of robotic operations through robot cell calibration. More specifically, an accurate workspace coordinate frame will be calculated using data points obtained from a Leica laser tracker. This section discusses the background of robotics and robot cell calibration, the rationale for the thesis topic, the scope of the thesis, the thesis layout and relevant literature.

1.1 Background

Robot technology is a rapidly growing area of research. Manufacturing is arguably the largest industry in which robots are being implemented. Robots are known for their high level of repeatability, but relatively poor absolute accuracy. In manufacturing settings, robots are typically implemented in what is called a robot cell. A robot cell is generally a full system that contains a robot, its respective controller, a table or conveyor belt, and other segments such as a part gripper and a safety system and can be seen in Figure 1.1. Robot cells are mainly used for completing repetitive and mundane tasks such as placing goods onto a pallet.

Figure 1.1: An example of a robot cell for palletizing. [1].

Accurate robotic operations are a fundamental element for robot cells to be successful and are a significant area of research. For a given industry project, the degree of accuracy required is subject to a given robot's application. For example, drilling holes on an aircraft body requires a high degree of accuracy [2] compared to a robot whose task is, say, simply to drop manufactured parts into a large storage bin. In any case, achieving accurate robotic operations to the desired degree can be difficult and often requires time consuming processes by jogging the robot either through manual programming or offline programming (OLP) touch-ups which

are discussed further in Section 1.2. This thesis proposes a jog-free method for increasing the accuracy of robotic operations using an algorithm that calculates an accurate workspace coordinate frame for the robot.

1.2 Rationale

As mentioned in the previous section, achieving a high degree of accuracy often requires manual programming or OLP touch-ups. Manual programming is a process in which the robot operator manually moves the robot, using a jog function on a teach pendant to desired targets, and recording the targets which are as accurate as naked eye measurements [3]. Because of this, the operator is able to account for discrepancies in the robot cell, such as an uneven work space, but can be very time consuming, especially if there are hundreds of targets. In contrast, OLP involves using a simulation program to replicate the robot cell which allows operators to very quickly create targets and robot programs, which is an improvement to manual programming [4]. One issue with OLP is that relative accuracy is sacrificed since the desired targets are no longer found using the naked eye, but rather are determined in a nominal version of the real world [4]. The nominal version does not account for real world errors such as manufacturing tolerances and, thus, targets often require touching up, which relies on the sight of a robot operator to adjust the target position. Having a method for determining robot targets that is quicker than manual programming and accounts for the discrepancies present in OLP would be ideal in robot cell calibration.

Many robot operations are executed relative to a workspace coordinate frame which is a coordinate frame that ideally is placed in a logical position, such as the corner of a table and close to the robot workspace. Several calibration methods, including the 3-point method, involve determining an accurate workspace coordinate frame to improve the robot cell calibration. The 3-point method is executed by manually jogging the robot to three separate points on a plane surface using a pointy end-of-arm-tool (EOAT) and recording the position of each point. A workspace coordinate frame is then calculated based on the three data points through vector geometry. This method is a built-in function in many robot brands, including ABB and KUKA, as it is simple, relatively quick, and accounts for discrepancies between the simulation and the real world. A common issue with the 3-point method is that it has poor accuracy at far distances from the origin. This issue occurs since the points are found using the naked eye and it is difficult to ensure all three points lie on the same plane as the table surface. Although the 3-point method is robust and widely used, its inaccuracy at far distances from the origin lowers its number of applications. It would be beneficial for robot cell calibration in industry to be more accurate than the 3-point method at farther distances from the origin and, thus, is the basis for this thesis.

1.3 Scope

The primary objective of this thesis is to improve the accuracy of robotic operations through detailed robot cell calibration. The scope for improving robot cell calibration is large and so, in this thesis, the scope is reduced to teaching the robot a more accurate workspace coordinate frame. This will be achieved by using a laser tracker to obtain positional data points and executing an algorithm that calculates an accurate workspace coordinate frame. The laser tracker and algorithm methodology is discussed further in Section 3.

Robot calibration will be excluded from the scope, which is a large area of robotics research. Robot calibration is different from robot cell calibration and involves improving the accuracy of the robot itself by analyzing a robot's kinematic model and altering the parameters to more accurately depict the robot's actual position [5]. Since robot calibration is complex and easily its own thesis topic, it will not be discussed in this thesis and the accuracy errors resulting from poor robot calibration will be considered negligible.

Real time error compensation is often used to calibrate robot cells. It is the process of reading positional data of a moving robot arm using an external measuring device and feeding the data into the robot controller to update its position. Real time error compensation will not be discussed since it is also a large area of research ans is beyond the scope of this thesis.

1.4 Thesis Layout

The layout for this thesis is as follows: the rest of this section contains a Literature Review. Included in the rest of the thesis are the following sections: Methodology in Section 3, Results in Section 4, Discussion in Section 5, and the Conclusion in Section 6. The end of the document contains the References.

2 Literature Review

To fully understand the importance of robot cell calibration and how laser tracking technology fits in, it is beneficial to understand the background of various cell calibration methods. This literature review will describe calibration methods such as the 3-point method and a variation of the 3-point method using a touch pad and where discrepancies with the real world occur with these methods. More complex methods have also been studied, some of which involved using external precision measuring equipment such as a laser pointer or by incorporating a virtual reality environment, which will be discussed.

There are already numerous methods used in industry to calibrate robot cells with one of the most notable being the 3-point method (which was discussed in Section 1.2). An arc welding robot application study was done by [6] using the methodology of the 3-point method. The results of this study showed high

accuracy with a translation error of <0.05 mm and <0.5°of rotational error. It is important to note that this accuracy level was only measured within the range of the small table (25 mm × 25 mm) used in the experiments. The accuracy of the 3-point method tends to become lower the farther away from the origin one gets and this is because it relies on the human eye to measure the points. The 3-point method is sensitive to minimal movements and can cause a large margin of error. Often in industry, a higher degree of precision is required. A method developed by [7] addresses the issue of relying on the human eye, and instead, uses a touch screen and a touch probe as the robot end effector. The method is similar to the 3-point method since the robot is jogged to the surface, but has increased accuracy since it relies on the input of the touch screen being contacted by the touch probe. The results of [7] are an error of approximately 0.02 mm if the robot is moving at 2 mm/s, an improvement in accuracy compared to [6]. Two assumptions made with the touch screen method are that the surface of the touch screen is parallel to the table and that the distance between the surface of the screen and the table is measured accurately. If these assumptions do not hold true, the accuracy of the method is not as reliable, especially at farther distances from the origin. It is important to note that both the 3-point method and the touch probe method make it difficult to position the coordinate frame in a sensible location, such as the exact corner of a table that has rounded corners, since the position is determined using the human eye.

The robotics research community has also explored cell calibration methods that use precise measuring equipment as opposed to relying on the robot controller for positional data and this is because there is a much higher achievable accuracy. One particular study conducted by [8] used a measuring system attached to the robot's flange that used a three cable mechanical system for measuring the x, y and z positions of the table relative to the flange. This method had an average error of approximately 1 mm within a work space of 2 m × 2 m × 2 m, but is possible to go well below 1 mm by accounting for some of the sources of error. The calibration required measuring 50 points, and took 25 minutes to perform, which is a longer process compared to other calibration methods. One of the more popular measuring devices, the laser line scanner (LLS), was used by [9] to develop a cell calibration algorithm. This algorithm achieved an absolute error of approximately 0.67 mm in ideal conditions, meaning the LLS position was optimal at 200 mm away from the measured surface as per the manufacturer. There were several sources of error with this method, one of the most notable being that the LLS was not exactly perpendicular to the table surface, which is very difficult to achieve in reality.

A more ambitious approach to robot cell calibration uses a virtual reality based system that creates a replica of the robot cell in a virtual environment. This method was used by [10] and the setup in this particular study proved to not be very accurate, having errors in the range of centimeters. Although the margin of error is large compared to other cell calibration methods, there are numerous sources of error

mentioned by [10], such as eliminating reflective surfaces in the work space. If these sources of error are accounted for, it can vastly increase its usability by combining accuracy with the utility of virtual reality.

There has been plenty of research on robot cell calibration, but there is a lack of options from which to choose to best fit all applications. This project will add a new method of finding a coordinate frame relative to the robot with a very high level of accuracy in a short amount of time using a laser tracker. This method is especially ideal in scenarios where new coordinate frames continuously need to be calibrated, for example, a cart of new parts is rolled in every hour and the corner of the cart needs to be calibrated every time. The position of the coordinate frame will also always be in the exact corner of the table which is an ideal location for pick and place applications since it facilitates making robot targets and programs.

3 Methodology

This section discusses the methodology used throughout experimentation. More specifically, it explains in detail how the frame transformations were calculated and used to obtain a workspace coordinate frame. Furthermore, it touches on what software was used to obtain positional data from the laser tracker and what EOATs were used in testing. Since the frame transformations in the algorithm rely on the laser tracker to obtain data points, it is important to fully understand how laser trackers operate before discussing how the obtained data was used.

Laser trackers measure the position of objects with very high accuracy; sometimes up to 15 μm. The laser tracker used in this project is a Leica AT960SR laser tracker, shown in Figure 3.1. They work by having a laser beam come out of the head of the laser tracker that reflects off a mirror-like apparatus called a spherically mounted retroreflector (SMR), back to the head, which gives a reading of the precise position of the SMR [11]. The head of the laser tracker follows the SMR in real time. SMRs are often held in magnetic nests that are manufactured to tight tolerances to facilitate data acquisition. The recorded position is sent from the laser tracker controller to an external computer via ethernet cable and can be read using the Pilot Tracker graphical user interface (GUI) or using the software development kit (SDK), both provided by Leica. The laser tracker used in this project is shown in Figure 3.1.

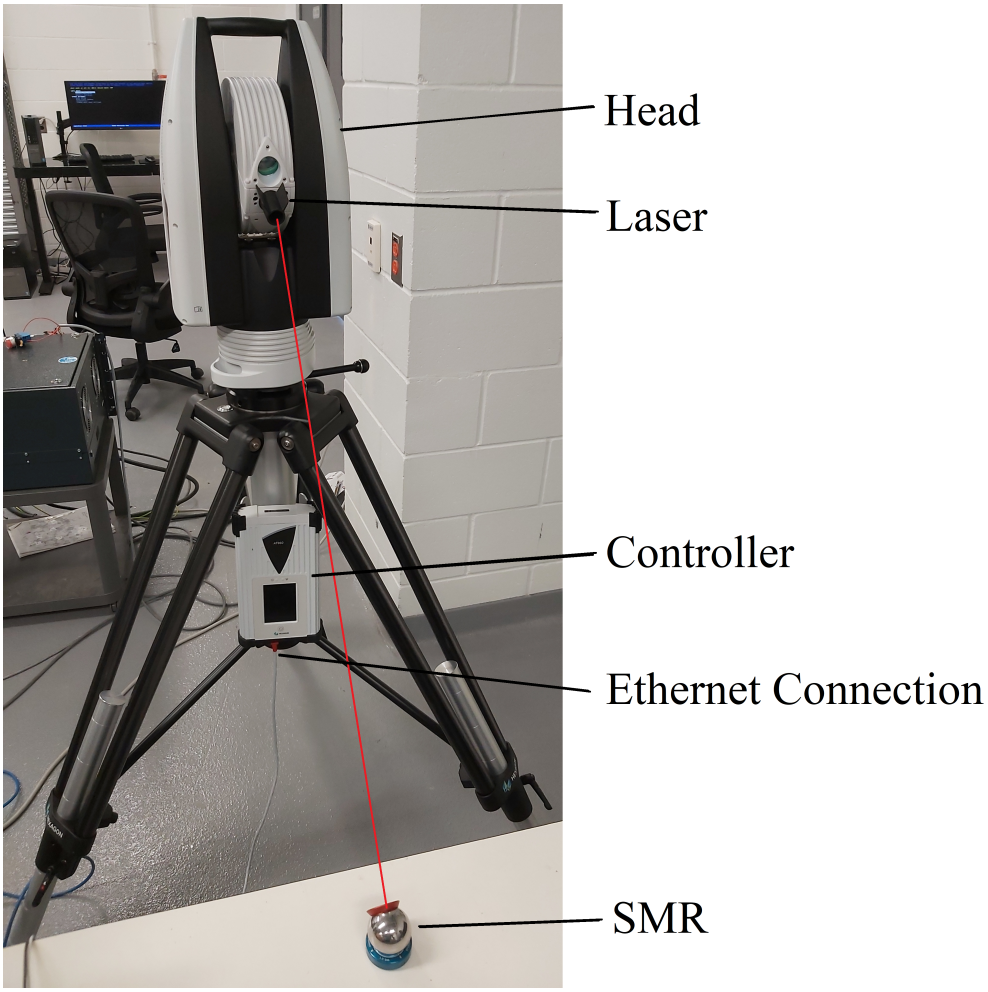

Figure 3.1: Components of a Leica laser tracker [12].

3.1 Data Acquisition Software

Two types of software were used to obtain positional data from the laser tracker controller to an external computer. These included Leica's Pilot Tracker software and Leica's SDK, which are discussed further in the following subsection.

3.1.1 Pilot Tracker

Pilot Tracker is a commercially available software provided by Leica that comes with the laser tracker. It is a GUI (shown in Figure 3.2) that shows the position of the centre of an SMR in either spherical or Cartesian coordinates in real time; this project uses the Cartesian coordinate system. Pilot Tracker also allows the user to record data points and export them as a CSV file which contains several categories of data for each obtained point including the position, the relative error, the temperature/pressure/humidity and the time

at which it was taken. There is also an option that allows the user to manually jog the head of the laser tracker to the desired position. Other useful options in Pilot Tracker are the ability to scan for nearby SMRs that are within a certain distance from the laser and an on board camera on the head of the laser tracker which allows the user to view at what the laser tracker is pointing. Pilot Tracker can be seen in Figure 3.2.

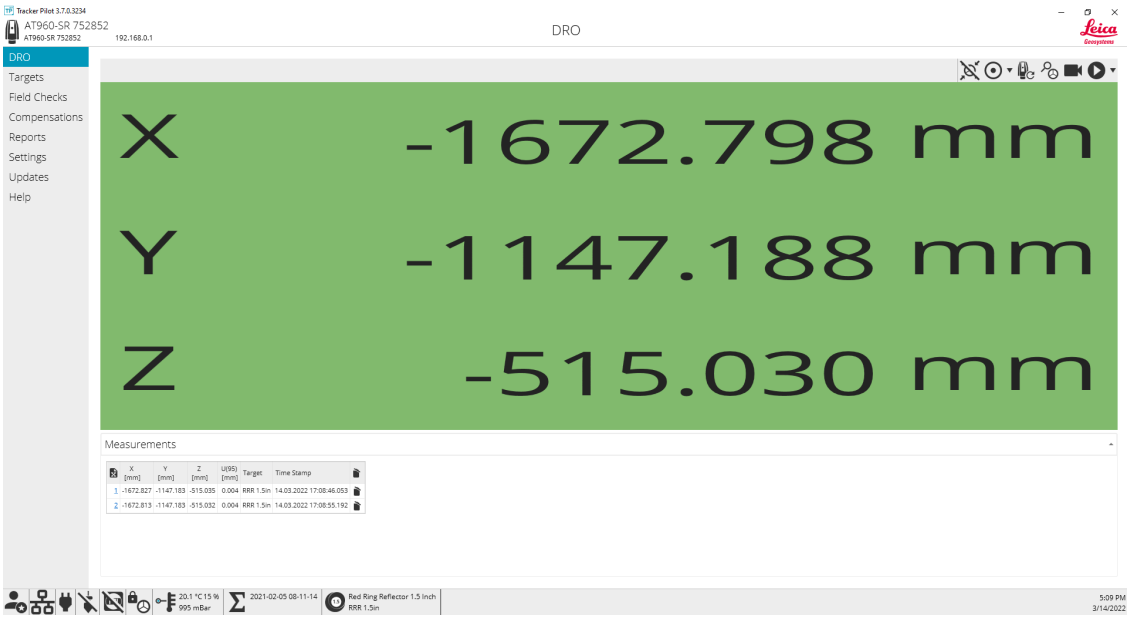

Figure 3.2: Pilot Tracker GUI.

The primary use for Pilot Tracker in this thesis is to obtain positional data for 17 positions which are discussed further in Section 3.3. The data points were exported as a CSV file and were manually saved into a specified directory that could directly be imported by the MATLAB algorithm when it was executed. Pilot Tracker was used more so in the earlier stages of the thesis for troubleshooting and quick data acquisition, but towards the end, a customized GUI was designed for data acquisition using the Leica SDK, which is discussed in the following subsection.

3.1.2 GUI Using the SDK

A customized GUI was designed, shown in Figure 3.3, using the Leica SDK to simplify data acquisition and to save time by creating a more automated data pipeline. The GUI was designed using the Microsoft Foundation Class Library (MFC) framework and was coded using the C++ interface [13]. Leica provides sample code of complete GUI examples and so the customized GUI used the framework of one of these samples. The customized GUI can be seen in Figure 3.3.

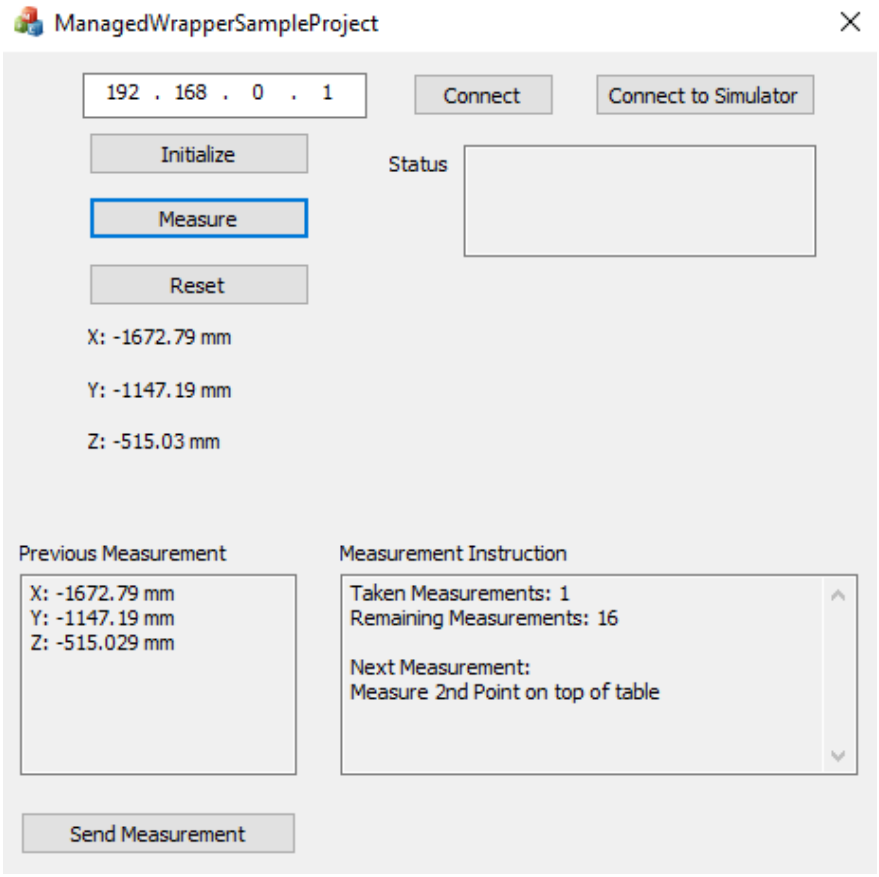

Figure 3.3: GUI created using the Leica laser tracker SDK.

Several features were added to the customized GUI including a counter on the number of remaining measurements to be taken. It also contains easy to follow steps for ensuring the 17 required data points were taken from the correct position so that any robot operator could execute the calibration routine. The customized GUI also has a 'Send Measurement' button which, when pressed, automatically saves a CSV file in a predetermined directory. This allowed the algorithm to execute more quickly since the file did not manually need to be saved in the appropriate directory. A lot of the preliminary testing of the GUI was done using the Leica laser tracker simulator since testing using the actual laser tracker was inefficient, and so there is an option to connect to a simulation environment on the GUI. The 'Reset' button restarts the 'Measurement Instruction' tab and clears the CSV file for a fresh start.

3.2 End-of-Arm-Tools

This subsection briefly discusses the EOATs used during experimentation for measuring and for verification: an SMR nest tool and a Sharpie measuring tool.

3.2.1 SMR Nest Tool

An SMR nest tool was used to hold an SMR to obtain accurate positional data when the robot was in specific positions and can be seen in Figure 3.4.

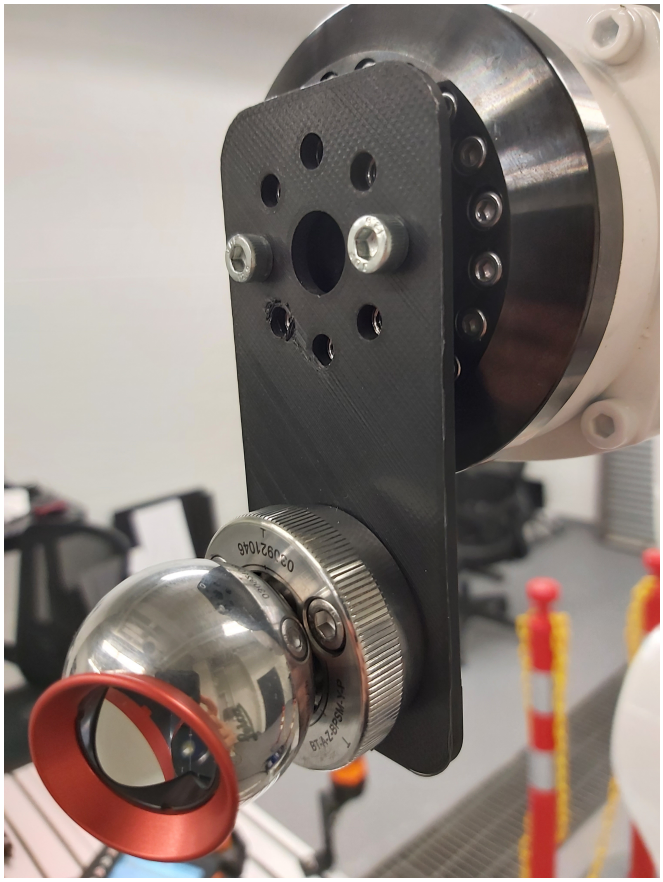

Figure 3.4: Complete SMR nest tool holding the 1.5" SMR.

The tool consists of 3 components: a 3D printed mounting plate, an aluminum SMR nest and the SMR itself. The mounting plate was designed to match the mounting patterns of the robot flange and the SMR nest and was printed using PLA. It was printed at 80% infill density to ensure it could hold the weight of the nest and SMR. The SMR nest was purchased from MetrologyWorks Inc. and is manufactured with high precision, lowering the potential positional error that it may cause in measurements.

3.2.2 Sharpie Measuring Tool

A Sharpie measuring tool was used as a verification tool for most measurements and was used to obtain the necessary data points for the 3-point method. When the calculations for the table coordinate frames were complete, this tool was installed on the robot end effector. The robot would then be jogged so that the tip of the Sharpie reached the origin of the calculated table coordinate frame to verify its accuracy. The

origin of the table coordinate frame was the corner of the table and so one could easily determine when the calculations were accurate or not; ideally the Sharpie would touch the exact corner of the table. It is important to note that the tip of the Sharpie is not extremely pointy and so rather than having one precise point be within acceptable accuracy, any point within close proximity (1-2 mm) would also be acceptable. The Sharpie measuring tool is shown in Figure 3.5.

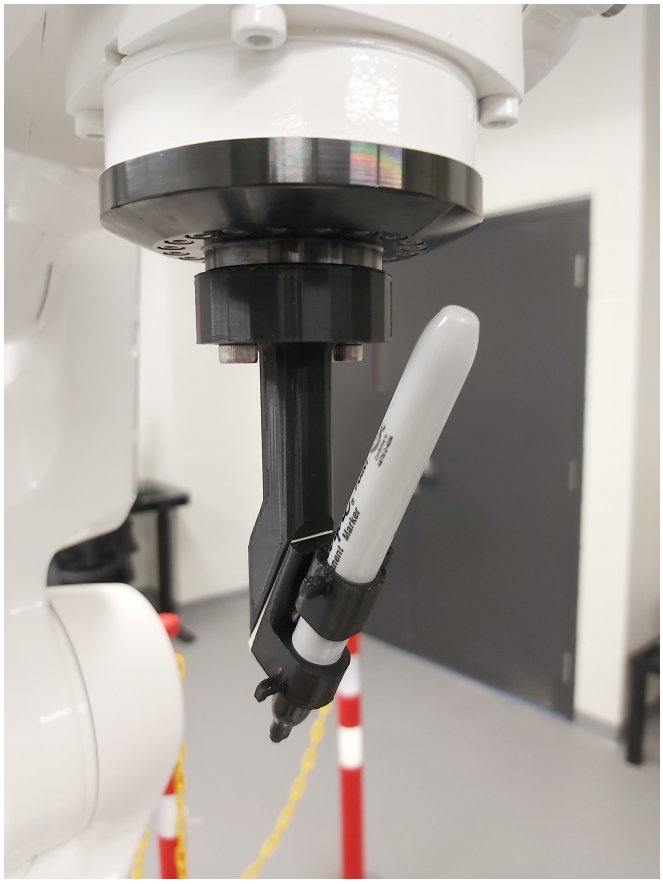

Figure 3.5: Complete Sharpie measuring tool used for verifying measurements.

The Sharpie measuring tool consists of 2 components: a 3D printed holder and the marker. The Sharpie was placed into the circular bracket on the holder and was held in place with friction during measurements to ensure the obtained data was consistent. The holder was designed so that the Sharpie was angled at 30° from the vertical to avoid robot singularities. Robot singularities are specific robot configurations that prevent certain movements of the EOAT and often occur when two joints are aligned along the same axis. A 30° angle misaligned the wrist joints of the robot, at joint four, five and six.

3.3 Frame Transformations

To achieve the goal of teaching the robot where the table coordinate frame is positioned, several frame transformations were calculated: $T1$ and $T2$. Figure 3.6 shows the three coordinate frames of interest and the associated transformations.

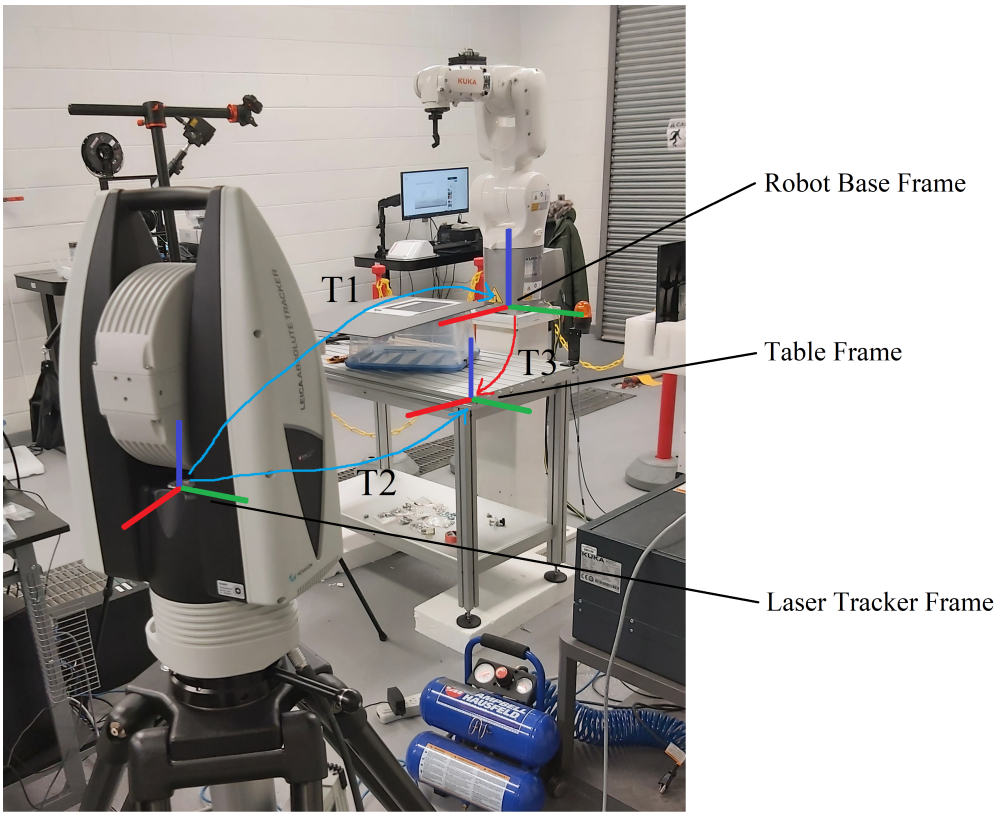

Figure 3.6: Laser tracker, table and robot base coordinate frames including the required transformations.

Each frame transformation can be represented as a 4×4 matrix containing both rotational and translational components. Equation 3.1 shows a standard transformation matrix where the r elements represent the rotational components and the x, y and z represent the translational components.

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The frame transformation between the laser tracker and the robot base ($T1$) and the frame transformation between the laser tracker and the table ($T2$) were calculated first, thereby, enabling the ability to calculate the frame transformation directly between the robot and table ($T3$) using Equation 3.2.

$$[T1][T3] = [T2] \quad \rightarrow \quad [T3] = [T1]^{-1}[T2] \tag{3.2}$$

Calculating these transformations with the laser tracker resulted in the robot having a more accurate table coordinate frame. The benefit of this is that robot targets and programs will have a much higher degree of accuracy, since they are relative to the new workspace coordinate frame.

The rotational components of a frame transformation matrix can also be represented as Euler angles: angles that represent how much each axis has rotated axially. They are commonly denoted as A, B and C. This means that, instead of representing the coordinate frame with a 4x4 matrix, it is represented with six values: x, y, z, A, B and C. On KUKA robots, the A value is the axial rotation of the z-axis; B is the axial rotation of the y-axis; and C is the axial rotation of the x-axis. The order in which these rotations are applied to a coordinate frame is crucial, since different orders will result in different coordinate frame positions. For KUKA robots, the proper order is to rotate about the z-axis, then the y-axis and then finally the x-axis.

The benefit of using the 4x4 matrices is that one can easily change the frame of reference simply by multiplying two of the matrices together, which is why they are used in the methodology portion of the thesis. The results and discussion portion of the thesis uses the Euler angles instead, since they are more logical as comparison metrics and easier to visualize.

3.3.1 Transformation from Laser Tracker to Table

This step involved finding an accurate coordinate frame associated with the workspace table relative to the laser tracker. More specifically, the frame transformation $T2$ is calculated, and is later used in Equation 3.2 to find $T3$. Nine data points were recorded in Pilot Tracker using the SMR; three sets of three points, where each set was taken from a different side of the table. The points could be taken arbitrarily as long as each group of three points was taken from the same side of the table, as they needed to lie on the same plane. Figure 3.7 shows how the points were obtained.

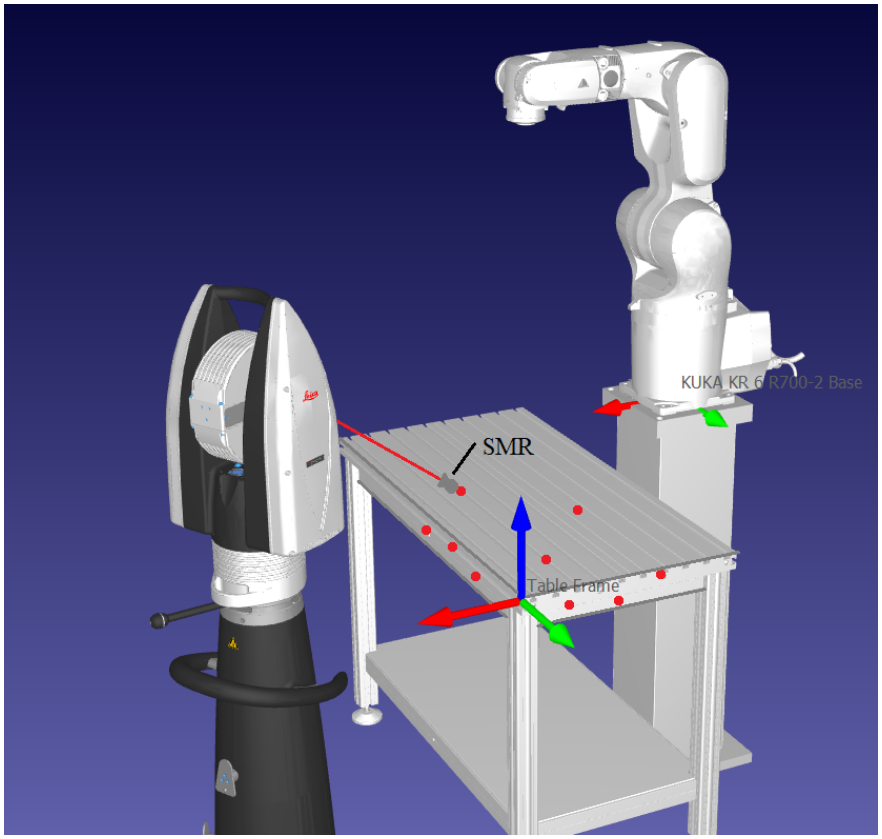

Figure 3.7: Obtaining nine data points, marked in red, using the laser tracker shown in the RoboDK environment.

The position data of the nine points was exported as an excel file and imported to a MATLAB script for further calculations.

Obtaining Positional Data of the Table Coordinate Frame

The position of the origin of the coordinate frame was found by calculating the intersection of the three planes where each plane represented one side of the table. Each plane was calculated using one of the three sets of points. The following steps were repeated for all three planes.

1. Find two vectors using the points obtained by subtracting them one from the other. Let P_1, P_2 and P_3 be the three points. It is important to note that each point is three dimensional and, thus, has an x, y and z component. Then:

$$\overrightarrow{V_1} = P_2 - P_1 = x_1\hat{i} + y_1\hat{j} + z_1\hat{k} \quad ; \quad \overrightarrow{V_2} = P_3 - P_1 = x_2\hat{i} + y_2\hat{j} + z_2\hat{k} \tag{3.3}$$

2. Calculate a vector, V_3, perpendicular to V_1 and V_2, the cross product, which is also a normal vector to

the plane.

$$\overrightarrow{V_3} = \overrightarrow{V_1} \times \overrightarrow{V_2} = det \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} = (y_1 z_2 - y_2 z_1)\hat{i} - (x_1 z_2 - x_2 z_1)\hat{j} + (x_1 y_2 - x_2 y_1)\hat{k} \tag{3.4}$$

3. With a point on the plane and a normal vector to the plane, a plane equation can be calculated according to Equation 3.5. Any one of the three points within the set can be used.

$$\overrightarrow{V_{3,\hat{i}}}(x - P_{1,x}) + \overrightarrow{V_{3,\hat{j}}}(y - P_{1,y}) + \overrightarrow{V_{3,\hat{k}}}(z - P_{1,z}) = 0 \tag{3.5}$$

Figure 3.8 shows one of the planes along with one of the sets of three points P_1, P_2 and P_3 used to calculate the vectors V_1, V_2 and V_3.

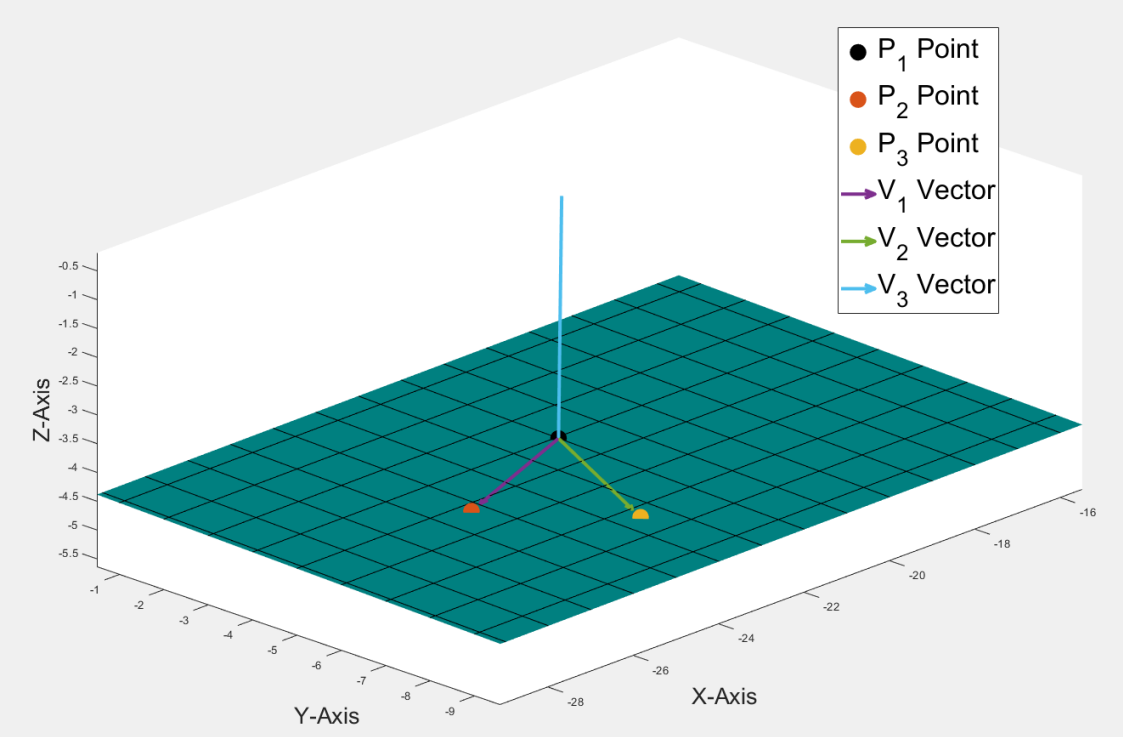

Figure 3.8: Calculation of one of the planes using measured data points to create vectors, plotted in MATLAB.

Using the calculated equations of the three planes, which have the form of Equation 3.5, finding the origin of the table coordinate frame simply involved solving a system of three linear equations for the x, y and z values. The plane equations first had to be converted to the form shown in Equation 3.6.

$$ax + by + cz = d \tag{3.6}$$

The a, b and c values are the coefficients for the x, y and z variables for each plane, respectively. The d values are the constants for each plane equation. The origin can now be calculated using Equation 3.7.

$$\begin{bmatrix} a_1x & b_1y & c_1z \\ a_2x & b_2y & c_2z \\ a_3x & b_3y & c_3z \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \tag{3.7}$$

The resulting point is the intersection of the three planes; therefore, it represents the corner of the table, and it will be substituted into Equation 3.1 at the x, y and z positions.

Obtaining Rotational Data of the Table Coordinate Frame

Finding the rotational data of the table coordinate frame involved finding the desired direction vectors of the three axes of the coordinate frame. It is also crucial that the vectors be perpendicular to one another. The subsequent steps were followed to obtain the rotational data for the new coordinate frame.

1. The normal vector of the plane that represents the table surface ($\overrightarrow{N_1}$) will be used as the z-axis. Take the cross product of $\overrightarrow{N_1}$ and the normal vector of one of the other planes ($\overrightarrow{N_2}$).

$$\overrightarrow{U_1} = \overrightarrow{N_1} \times \overrightarrow{N_2} \tag{3.8}$$

   This new vector, $\overrightarrow{U_1}$, is the direction vector that represents the edge of the table and will be used as the y-axis.

2. Take the cross product between $\overrightarrow{N_1}$ and the newly calculated $\overrightarrow{U_1}$ to find the vector that represents the other edge of the table, $\overrightarrow{U_2}$, perpendicular to the previous one. This will be used as the x-axis.

$$\overrightarrow{U_2} = \overrightarrow{N_1} \times \overrightarrow{U_1} \tag{3.9}$$

Shown in Figure 3.9 are the direction vectors of the axes of the table coordinate frame (red, green and blue) relative to the laser tracker frame (black).

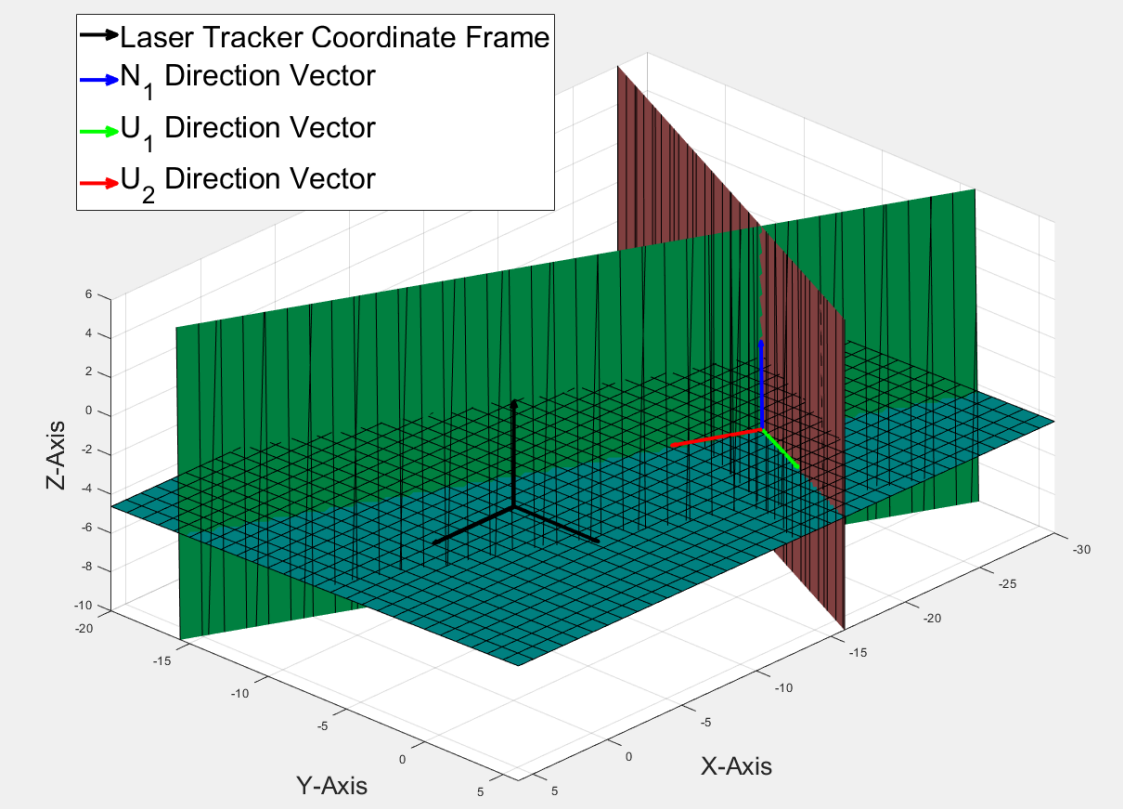

Figure 3.9: Table coordinate frame relative to laser tracker coordinate frame, plotted in MATLAB.

Combining the calculated direction vectors of the table coordinate frame with the origin coordinates creates a completely defined table coordinate frame. The x, y and z components of each of the table coordinate frame axes, $\overrightarrow{N_1}$, $\overrightarrow{U_1}$ and $\overrightarrow{U_2}$, totaling nine values, will be substituted into Equation 3.1 at the position of the nine rotational components. The relative positions of the table and laser tracker coordinate frames in Figure 3.9 are similar to the positions shown in Figure 3.6, which is what one would expect, since the data points were taken with the robot and table in those approximate positions relative to the laser tracker.

3.3.2 Transformation from Laser Tracker to Robot Base

This step involved finding an accurate coordinate frame of the robot base relative to the laser tracker, $T1$. The robot base frame is the coordinate frame that lies in the bottom and center of the physical robot base. To find the robot base relative to the laser tracker, eight data points were required. The robot was jogged to five different poses and position data was taken at each pose. To record the points, the robot had a plate end effector to hold the SMR nests. The robot poses are shown below in Figure 3.10.

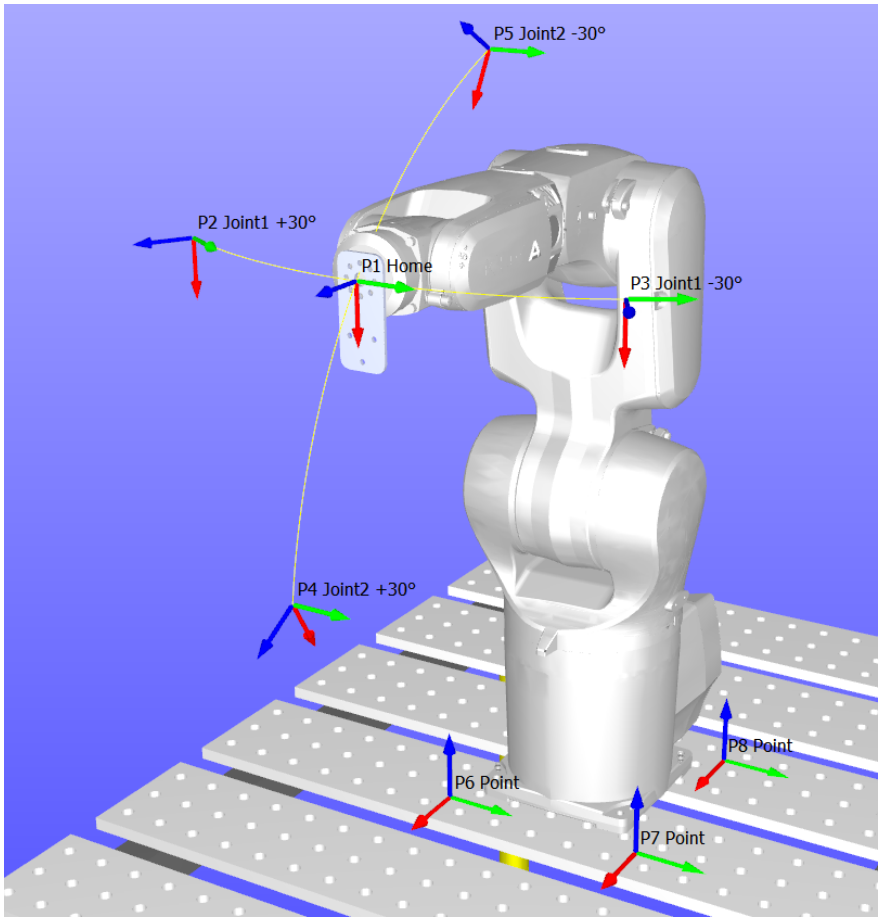

Figure 3.10: Eight points used to calculate the robot base frame shown in RoboDK.

The five target points and the three data points at the robot base were used to calculate several planes. By using geometry and plane intersection vectors, a robot base frame was calculated.

Obtaining Positional Data of Table Coordinate Frame

To find the x and y coordinates of the robot base frame, three of the robot pose positions were used: P_1, P_2 and P_3 from Figure 3.10. The subsequent steps were followed.

1. Create two vectors using P_1, P_2 and P_3; the first between P_1 and P_2; and the second between P_1 and P_3.

$$\overrightarrow{V_1} = P_1 - P_2 = x_1\hat{i} + y_1\hat{j} + z_1\hat{k} \quad ; \quad \overrightarrow{V_2} = P_2 - P_3 = x_2\hat{i} + y_2\hat{j} + z_2\hat{k} \tag{3.10}$$

2. Find the middle point of the vector by averaging the two points used to create the vector.

$$P_{m1} = \frac{P_1 + P_2}{2} = \frac{(x_1 + x_2)\hat{i} + (y_1 + y_2)\hat{j} + (z_1 + z_2)\hat{k})}{2} \tag{3.11}$$

$$P_{m2} = \frac{P_2 + P_3}{2} = \frac{(x_2 + x_3)\hat{i} + (y_2 + y_3)\hat{j} + (z_2 + z_3)\hat{k})}{2} \tag{3.12}$$

3. Using the three points P_1, P_2 and P_3, calculate a plane and a normal vector to the plane ($\overrightarrow{N_1}$) using the same method adapted in Section 3.3.1. Figure 3.11 shows the points measured from the laser tracker P_1, P_2 and P_3; the vectors between those points V_1 and V_2; the average points P_{m1} and P_{m2} and the plane on which they all lie.

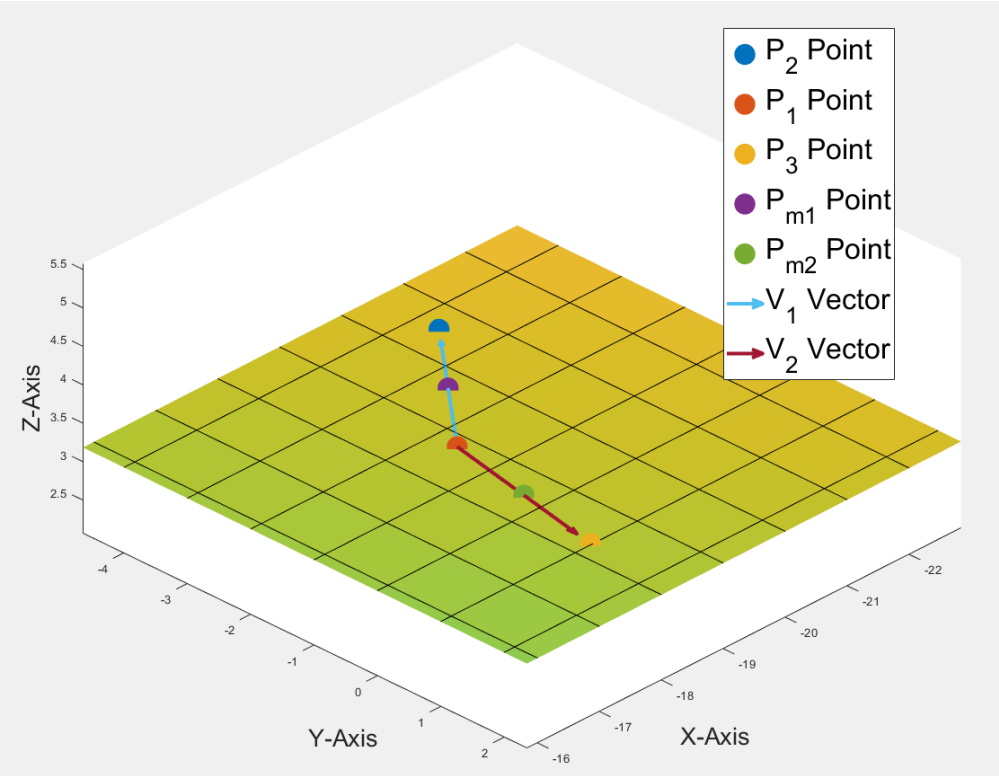

Figure 3.11: Calculation of the vectors and average points using the measured data points, plotted in MATLAB.

4. To find the the x and y coordinates that represent the center of the robot base, two new vectors were found, where one is perpendicular to $\overrightarrow{V_1}$ and the other is perpendicular to $\overrightarrow{V_2}$; and they must both lie in the same plane found in Step 3. The origin of the vectors must be the middle points found in Step 2. As a result, the intersection of the new vectors are the x and y coordinates of the robot base frame. In other words, two cross products were taken; the first being between the normal of the plane shown in Figure 3.11, $\overrightarrow{N_1}$, and $\overrightarrow{V_1}$; and the second being between $\overrightarrow{N_1}$ and $\overrightarrow{V_2}$.

$$\overrightarrow{U_1} = \overrightarrow{N_1} \times \overrightarrow{V_1} \quad ; \quad \overrightarrow{U_2} = \overrightarrow{N_1} \times \overrightarrow{V_2} \tag{3.13}$$

5. The vectors were then converted into parametric form in order to find the intersection.

$$X_1 = P_{m1,x} + t_1(\overrightarrow{U_{1,x}}) \quad ; \quad Y_1 = P_{m1,y} + t_1(\overrightarrow{U_{1,y}}) \quad ; \quad Z_1 = P_{m1,z} + t_1(\overrightarrow{U_{1,z}}) \tag{3.14}$$

$$X_2 = P_{m2,x} + t_2(\overrightarrow{U_{2,x}}) \quad ; \quad Y_2 = P_{m2,y} + t_2(\overrightarrow{U_{2,y}}) \quad ; \quad Z_2 = P_{m2,z} + t_2(\overrightarrow{U_{2,z}}) \tag{3.15}$$

Equating X_1 to X_2 and Y_1 to Y_2 reduced the problem to simply solving a system of two linear equations for the t_1 and t_2 values.

$$\begin{bmatrix} t_{1,1} & t_{2,1} \\ t_{1,2} & t_{2,2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.16}$$

Once these values were found, substituting them back into either Equation 3.14 or Equation 3.15 gave the x and y coordinates of the intersection of the $\overrightarrow{U_1}$ and $\overrightarrow{U_2}$ vectors which are also the x and y coordinates of the robot base frame. Figure 3.12 is an expansion of Figure 3.11 but also shows the newly calculated U_1 and U_2 vectors along with their intersection.

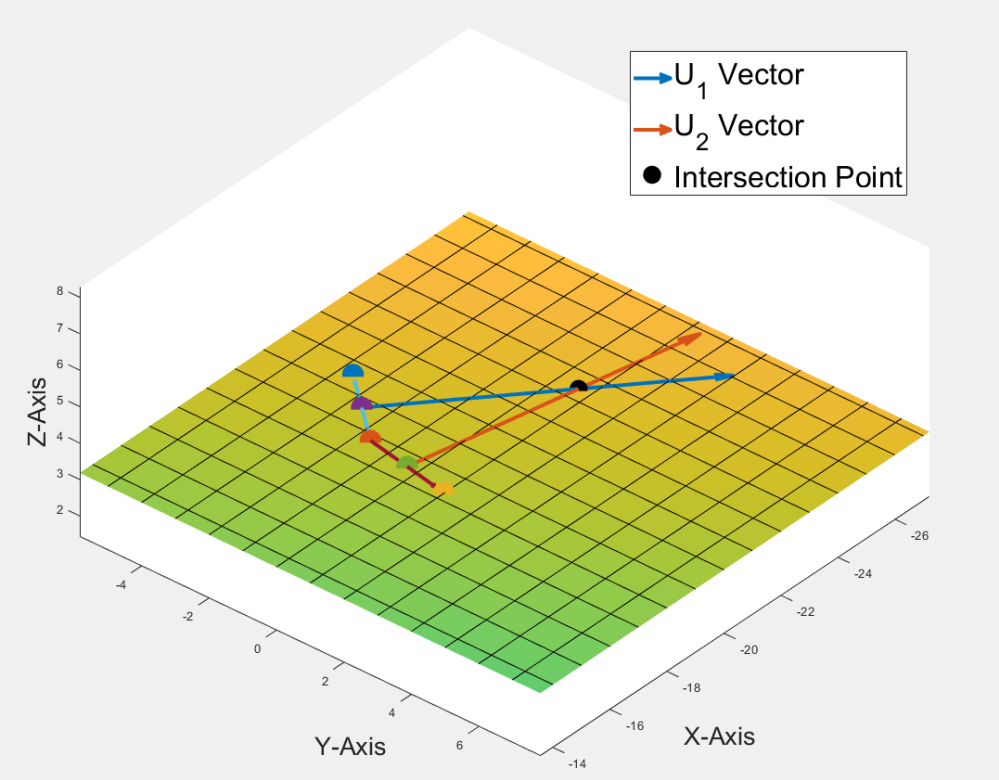

Figure 3.12: Finding the x and y coordinates for the robot base frame, plotted in MATLAB.

Finding the z coordinate of the robot base frame involved calculating a plane equation (Π_1) using the P_6, P_7 and P_8 points shown in Figure 3.10, which represents the table surface on which the robot sits. This was achieved using the same methodology used in Section 3.3.1 and so the plane equation has the form of Equation 3.17.

$$\overrightarrow{V_{3,\hat{i}}}(x - P_{1,x}) + \overrightarrow{V_{3,\hat{j}}}(y - P_{1,y}) + \overrightarrow{V_{3,\hat{k}}}(z - P_{1,z}) = 0 \tag{3.17}$$

The x and y coordinates were then substituted into the plane equation, making it possible to solve for the z coordinate. Obtaining the rotational data of the frame was the next task, as the steps for finding the origin were completed.

Obtaining Rotational Data of Table Coordinate Frame

Since the position of the robot base frame is fully defined, the direction vectors of the three axes were determined through the following steps.

1. The z-axis of the robot base frame was taken as the normal vector of Π_1, $\overrightarrow{N_2}$, which was calculated in

the previous step and substituted in Equation 3.17.

2. Using P_1, P_4 and P_5, a third plane equation (Π_2) was calculated using the same methodology as Section 3.3.1. The vector represented by the intersection of Π_1 and Π_2, $\overrightarrow{V_3}$, was set as the x-axis vector. This intersection vector was calculated using the same method from Section 3.3.1.

3. The cross product between $\overrightarrow{N_2}$ (z-axis) and $\overrightarrow{V_3}$ (x-axis) gave the direction vector, $\overrightarrow{V_4}$, of the y-axis.

$$\overrightarrow{V_4} = \overrightarrow{N_2} \times \overrightarrow{V_3} \tag{3.18}$$

Now that the direction vectors for the three axes of the robot base frame have been found, $\overrightarrow{N_2}$, $\overrightarrow{V_3}$ and $\overrightarrow{V_4}$, the robot base frame is completely defined. The calculated planes and the position of the axes of the robot base frame can be seen in Figure 3.13, where $\overrightarrow{V_3}$, $\overrightarrow{V_4}$ and $\overrightarrow{N_2}$ are the x, y and z axes, respectively.

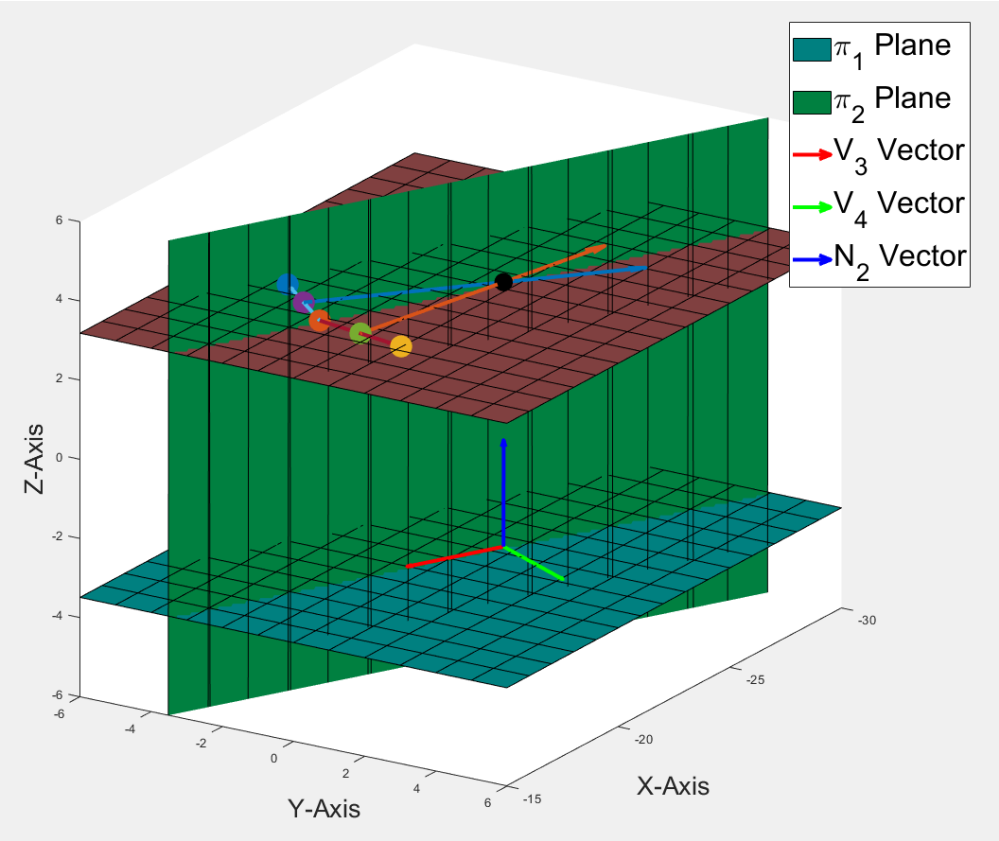

Figure 3.13: Updated version of Figure 3.12 showing the position of the robot base frame.

Combining the calculated direction vectors of the robot base frame with the origin coordinates creates a completely defined table coordinate frame.

3.3.3 Table Coordinate Frame Relative to the Robot Base

The direction vectors of the axes and the position of the origin were then substituted into the rotational and translation components, respectively, of the matrix with the form shown in Equation 3.1 for both the table frame and robot base frame. Using these matrices, Equation 3.19 was used to obtain the position of the table frame relative to the robot base frame.

$$[T3] = [T1]^{-1}[T2] \tag{3.19}$$

4 Results

The following section displays the results of the laser tracker algorithm in several categories including precision, accuracy, execution time and the use of the algorithm with various table geometries, i.e. rounded table corners. The laser tracker algorithm was compared to the 3-point method in all of the aforementioned categories to better understand its relative performance. The 3-point method was selected as the method of comparison since it is widely known in the field of robotics; it is included in the programming in many robot controllers; and it is simple to execute.

4.1 Precision Comparison

To determine the relative precision of the laser tracker algorithm, three table coordinate frames were calculated using the algorithm; and three more table coordinate frames were determined using the 3-point method. It was important that, for all six tests, everything remained as uniform as possible, for example, keeping the table in the exact same spot to ensure the results were valid. The 3-point method tests used a Sharpie measuring tool to measure the points. The results for the six tests are shown in Table 4.1. It should be noted that the coordinates shown in Table 4.1 are the coordinates of the table coordinate frame relative to the robot base frame.

Table 4.1: Precision Test Results (n=3)

Test #	**Description**	**x (mm)**	**y (mm)**	**z (mm)**	**A (°)**	**B (°)**	**C (°)**
1	LT Algorithm	430.645	-256.571	-92.498	1.431	0.158	1.419
2	LT Algorithm	430.215	-257.123	-92.640	1.385	0.159	1.441
3	LT Algorithm	430.345	-257.461	-92.636	1.323	0.164	1.419
4	3-point Method	432.95	-258.72	-92.59	-178.94	0.04	-1.51
5	3-point Method	432.16	-259.53	-92.23	-179.14	0.62	-1.33
6	3-point Method	432.02	-259.31	-92.95	-179.27	0.65	-1.25

The results for all six tests were relatively similar to one another (<1 mm), with the exception of the A values between the laser tracker algorithm and the 3-point method being approximately 180° apart. This variation was expected because the 3-point requires a point along the x-axis, which is easier to measure on the edge of the table rather than in open air. Because of this, the x-axis was pointing in the opposite direction of the x-axis of the laser tracker algorithm, towards the robot, causing the 180 ° difference. Since this subsection discusses repeatability only, the difference of the A values between methods will not affect the results. Shown in Table 4.3 are the mean and standard deviations (S) for the x, y, z, A, B and C values for each method.

Table 4.2: Mean and Precision Results From the Six Tests (n=3)

Metric	**Description**	**x (mm)**	**y (mm)**	**z (mm)**	**A (°)**	**B (°)**	**C (°)**
Mean	LT Algorithm	430.401	-257.051	-92.591	1.379	0.160	1.426
Mean	3-point Method	432.376	-259.186	-92.59	-179.116	0.436	-1.363
S	LT Algorithm	0.221	0.449	0.08	0.05	0.003	0.012
S	3-point Method	0.501	0.418	0.36	0.166	0.343	0.133

Table 4.3: Mean and Precision Results From the Six Tests (n=3)

Metric	**LT Algorithm**	**3-point Method**
Std Dev.	**0.136**	**0.320**
X Error (mm)	**0.52**	**5.3**
Y Error (mm)	**0.32**	**2.86**
Time (s)	**59**	**85**

The averages between the methods were relatively close (<1 mm), but some values had consistent discrepancies, namely, the x and y coordinates had an average difference of approximately 2 mm. This is likely attributed to the inaccuracy of the Sharpie measurement tool since the tool tip does not come to a sharp point, but rather a blunt tip. Another potential cause for this discrepancy is that the corner of the table was not a perfect square and so it was difficult to locate the Sharpie at an exact corner.

The standard deviations for the six coordinate frame values were significantly lower for the laser tracker algorithm compared to the 3-point method, with the exception of the y coordinate. The discrepancy in the y coordinate is likely attributed to poor measurements taken from the laser tracker since small movements (<5 mm), such as the table/SMR shaking slightly, can cause poor measurement readings.

4.2 Accuracy Comparison

This subsection discusses the accuracy of the laser tracker algorithm compared to the accuracy of the 3-point method. To determine the accuracy, each method was executed on the same workspace table to obtain table coordinate frames. The position of the origin of the coordinate frames were compared between methods, along with two other points: a point along the x-axis and a point along the y-axis. These points were compared by jogging the robot to each of the three positions for each method. Choosing these three positions would make it obvious if there were inaccuracies with the methods since the axes of the coordinate frame represent the edges of the table and so the robot should be exactly at the edge. The point along the x-axis was 400 mm away from the origin and along the y-axis was 250 mm away. These values were chosen since they were within the robots reach and in uncomplicated robot configurations. To quantify the inaccuracy errors, a correction to the position of the robot was made by jogging the robot to an ideal position (exactly on the corner/edge of the table) which was visually determined using the Sharpie measurement tool.

Table 4.4, 4.5 and 4.6 show the hypothetical position (x, y and z) and the corrected position (x', y' and z') at the origin for both methods. Additionally, the coordinates shown in the tables are relative to the table coordinate frame itself. Table 4.4 shows the comparison of the origin for both methods.

Table 4.4: Accuracy Test Results at Origin for Both Methods (n=1)

Method	**x (mm)**	**x' (mm)**	**y (mm)**	**y' (mm)**	**z (mm)**	**z' (mm)**	**Absolute Error (mm)**
LT Algorithm	0	0	0	0	0	0	**0**
3-point Method	0	0	0	2.46	0	0	**0.82**

The position of the origin for the laser tracker method was very accurate (effectively 0 mm) so the

corrected position was left as the hypothetical position. It is important to note that there likely is a small error, but since the errors were determined using eyesight, its value is indeterminate and, thus, negligible for this project. The 3-point method was slightly off in the y direction at the origin and so the robot was jogged 2.46 mm in that direction for the corrected position. The mean error for the 3-point method was 0.82 mm. Overall, the origin positions for both methods were very accurate, which was expected. Figures 4.1 and 4.2 show the uncorrected positions for the laser tracker method and 3-point method at the origin, respectively.

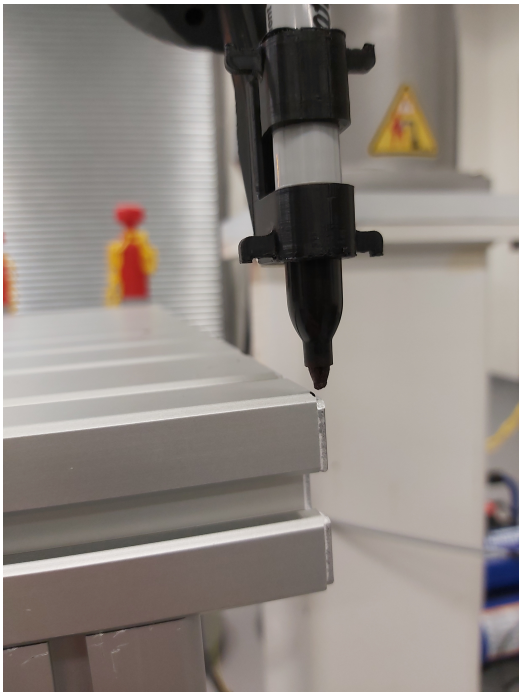

Figure 4.1: LT method origin position, uncorrected.

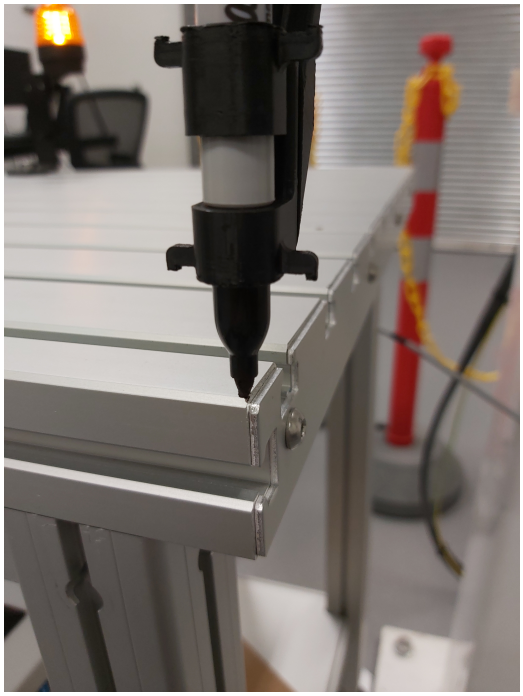

Figure 4.2: 3-point method origin position, uncorrected.

The uncorrected position for the laser tracker algorithm may look like it is slightly off the table, but that is the desired position. This is because the origin is the intersection of the three planes that represent the three sides of the table and the plane on the right side of table lies on the extended aluminum plate that is slightly lower. The accuracy comparison results for the point on the x-axis are shown in Table 4.5.

Table 4.5: Accuracy Test Results on x-axis for Both Methods (n=1)

Method	**x (mm)**	**x' (mm)**	**y (mm)**	**y' (mm)**	**z (mm)**	**z' (mm)**	**Absolute Error (mm)**
LT Algorithm	-400.00	-400.00	0	1.09	0	-0.46	**0.52**
3-point Method	-400.00	-400.00	0	10.28	0	0.32	**5.3**

The uncorrected position was very close to the hypothetical value for the laser tracker method; off by 1.09 mm in the y direction and -0.46 mm in the z direction. Alternatively, with the 3-point method, the y direction was off by a significant margin (10.28 mm) while only 400 mm away from the origin. It was also off by 0.32 mm in the z direction. The mean error for the laser tracker method was 0.52 mm; and 5.3 mm for the 3-point method. It is clear from this test that the x-axis calculated using the 3-point method was not as accurate as the laser tracker method. Figures 4.3 and 4.4 show the uncorrected positions for the laser tracker method and 3-point method on the x-axis, respectively.

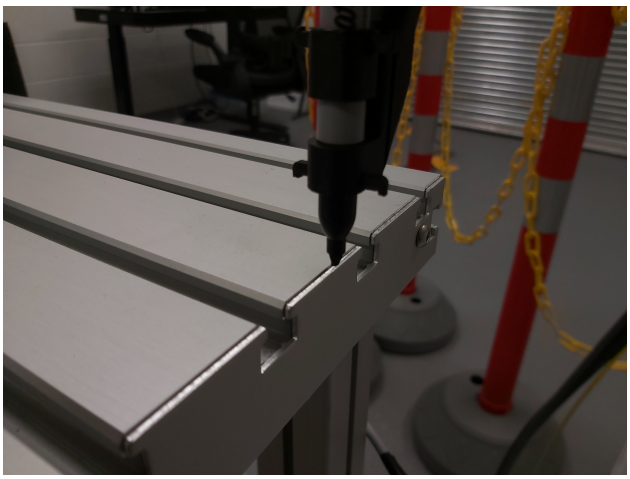

Figure 4.3: LT method x-axis position, uncorrected.

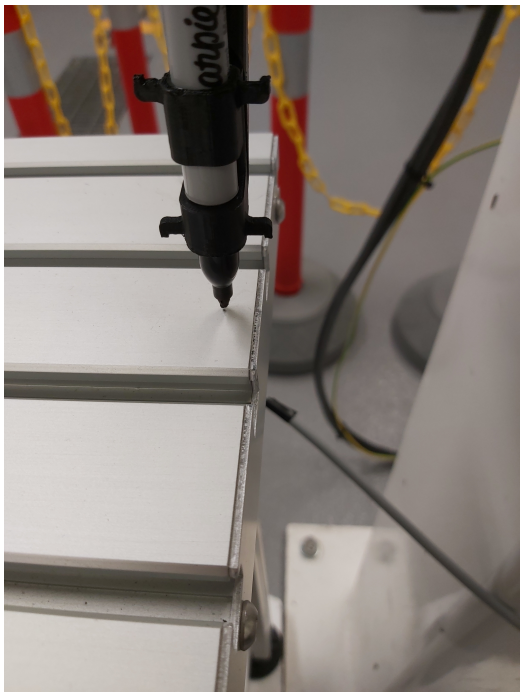

Figure 4.4: 3-point method x-axis position, uncorrected.

It is clear from these figures that the laser tracker method has a more accurate x-axis calculation. Table 4.6 shows the accuracy comparison results for the point on the y-axis.

Table 4.6: Accuracy Test Results on y-axis for Both Methods (n=1)

Method	**x (mm)**	**x' (mm)**	**y (mm)**	**y' (mm)**	**z (mm)**	**z' (mm)**	**Absolute Error (mm)**
LT Algorithm	0	0.96	-250.00	-250.00	0	0	**0.32**
3-point Method	0	-2.58	-250.00	-250.00	0	-6.00	**2.86**

The accuracy for the laser tracker method on the y-axis was satisfactory, with a discrepancy of only 0.96

mm in the x direction. The 3-point method, alternatively, was off by a significant margin in the z direction (-6 mm). The error in the x direction was also relatively significant, -2.58 mm. Additionally, the point at which these discrepancies occurred was only 250 mm away from the origin, which is quite close. The mean error for the laser tracker method was 0.32 mm; and 2.86 mm for the 3-point method. The uncorrected positions for the laser tracker method and 3-point method on the y-axis are shown in Figures 4.5 and 4.6, respectively.

Figure 4.5: LT method y-axis position, uncorrected.

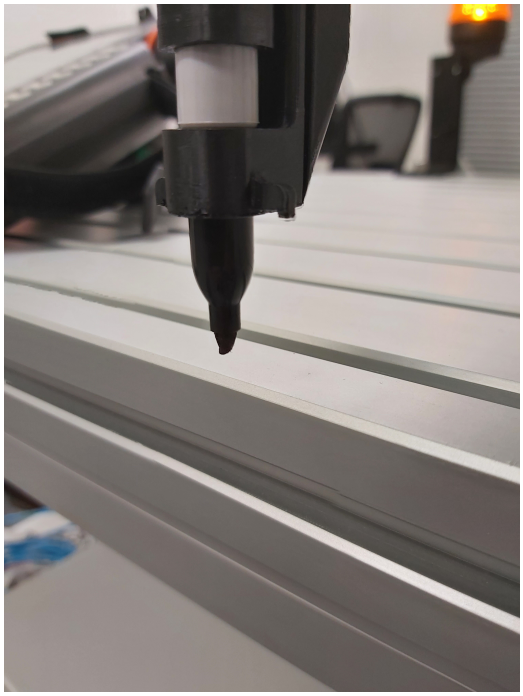

Figure 4.6: 3-point method y-axis position, uncorrected.

Overall, the laser tracker method was much more accurate compared to the 3-point method, with the maximum error being approximately 1 mm. The maximum error of the 3-point method was approximately 10 mm. One source of error for the laser tracker method is the inaccuracy of a Sharpie measurement tool when determining corrected positions. Another error could be poor measurements taken from the laser tracker which may be caused by a slight movements in the SMR. There are likely several causes for the inaccuracies in the 3-point method, including the inaccuracy in the Sharpie measurement tool, for obtaining the three points and also for determining the corrected positions. Another source of error for the 3-point method is the presence of robot calibration inaccuracies.

4.3 Execution Time Comparison

The execution time of the laser tracker algorithm was compared to the execution time of the 3-point method in Table 4.7 in this subsection. For the laser tracker algorithm, the execution time was reduced to the time it took to obtain the nine measurements on the workspace table, rather than the full 17 measurements. This is because in an industry setting, it is assumed that the position of the five measurements on the robot and the three measurements at the robot base will almost never change; the robot will not move. For the remaining nine points, it is assumed that they will change often i.e. when a new cart of parts is rolled into the robot cell for processing and the robot needs an updated coordinate frame of the cart's position to accurately process the parts. For the 3-point method, the execution time was the time it took for the robot to leave the home position, jog to the three points, and return to the home position. There were no time reductions for this method since the hypothetical cart of parts would require three new points every time to ensure an accurate workspace coordinate frame. Table 4.7 shows the results for the execution time of the two methods.

Table 4.7: Relative Execution Times of the LT Algorithm and the 3-point Method (n=1)

Method	**Execution Time (s)**
LT Algorithm	59 s
3-point Method	85 s

The laser tracker algorithm is executed approximately 25 seconds faster than the 3-point method. It is important to note that the execution time of the 3-point method is often inversely proportional to the quality of the results. The quicker the robot operator jogs the robots to the three points, the more likely the results will be inaccurate. Alternatively, with the laser tracker algorithm, the execution time is relatively constant and the quality of results are often uncorrelated. The robot operator simply places the SMR in the correct position and the laser tracker takes the measurement.

4.4 Table Geometry

All of the accuracy and precision comparisons were completed using a square cornered table, which might not be the case in all scenarios where these methods may be required. Many tables have rounded corners and so a comparison will be made for the laser tracker method and the 3-point method on how they perform using a table with a rounded corner. Figure 4.7 shows the 3D printed rounded corner piece on which the testing was done.

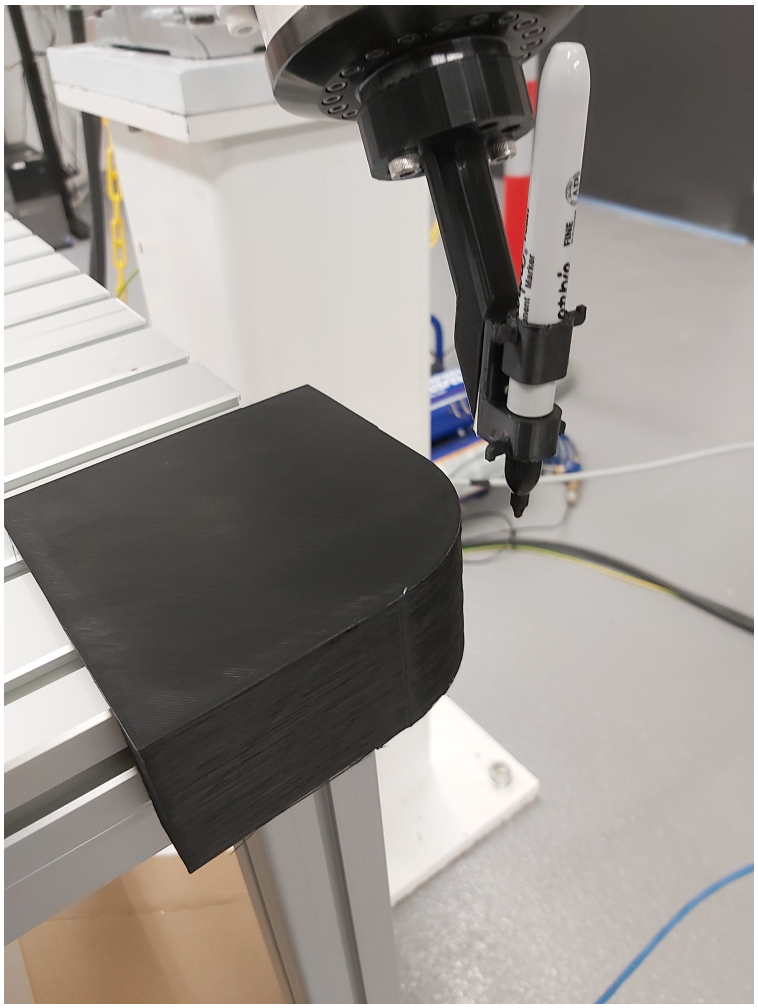

Figure 4.7: 3D printed rounded corner.

A coordinate frame was found using both methods on the rounded corner. The procedure for the laser tracker method was the same as it was for a square cornered table and so no issues arose. For the 3-point method, it was difficult to accurately determine the origin point since it was in open air. The x-axis and y-axis vectors for the laser tracker algorithm along with the origins for both methods can be seen in Figure 4.8.

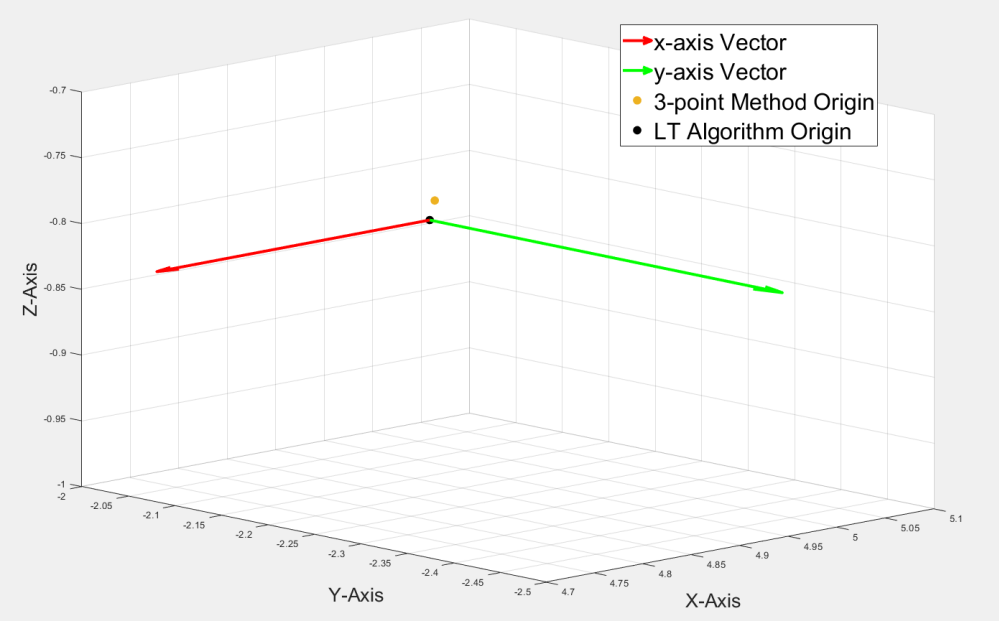

Figure 4.8: Comparison of origins of the two calibration methods on a rounded table corner, plotted in MATLAB.

As expected, there was a discrepancy between the origins of the two methods, albeit a minor one. Since the origin is determined using geometry for the laser tracker method, an assumption can be made that it is closer to the true corner than the 3-point method whose origin was approximated using eyesight alone.

5 Discussion

This section will discuss some of the non-result oriented aspects of the thesis. These include a safety comparison between the laser tracker algorithm and the 3-point method, a discussion on the required EOATs between the methods and a challenge encountered while creating the laser tracker algorithm.

5.1 Safety Comparison

When comparing the safety of the laser tracker algorithm and the 3-point method, the laser tracker algorithm is generally safer. In order to obtain an accurate coordinate frame for the 3-point method, the operator is required to bring their eyes very close to the sharp EOAT that is measuring the points, which could result in a severe accident with a sudden inadvertent robot movement. The laser tracker algorithm does not require a sharp EOAT and thus lowers the risk. Most of the required points for the laser tracker algorithm do not require moving the robot, but for the five that do, the robot can be jogged into position and then the robot operator can enable the emergency stop. The operator can then safely walk into the robot cell, place the

SMR in the necessary position, and take a measurement.

5.2 Required EOATs for Calibration

Both calibration methods have required EOATs to obtain the desired results and the tool functionality is discussed in Section 3.2.1. A benefit with the laser tracker algorithm is that its only required EOAT, the SMR nest tool, is solely required during initial calibration of the cell, assuming the robot's position does not change. The nine points measured on the table that will likely change often, simply require the SMR and no robot tool. Alternatively, the 3-point method requires a sharp EOAT every time a new coordinate frame needs to be calibrated. This is not ideal in robot cells that use EOATs with an unclear tool centre point (TCP), for example a vacuum gripper, shown in Figure 5.1.

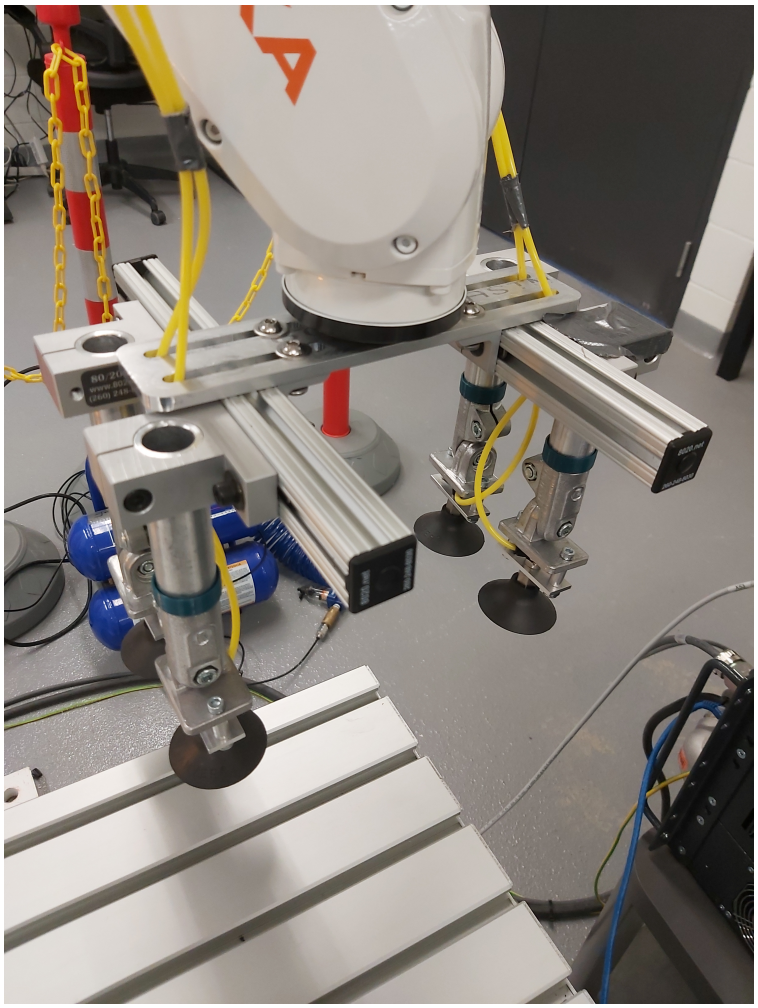

Figure 5.1: Unclear TCP of a vacuum gripper.

There is no obvious point on the vacuum gripper from which to take the three points for the 3-point

method. Another option is to change out the tools, but this option is not ideal in an industry setting since it requires significant time. Overall, the laser tracker method is more efficient since it does not require an EOAT, post initial cell calibration. The 3-point method is less efficient in cells that have EOATs with unclear TCPs, since it sacrifices time by swapping out EOATs.

5.3 Challenge

This section discusses a challenge that arose when trying to obtain an accurate table coordinate frame relative to the robot base frame using the laser tracker for data points.

5.3.1 Non-Perpendicular Coordinate Axes

One issue that arose was that the direction vectors of the x-axis and y-axis were not exactly perpendicular when calculating the table coordinate frame relative to the laser tracker. This is a major issue since coordinate frames axes must all be perpendicular to one another, and errors will be incurred on the robot and in the robot programming simulation environment if this is not the case. This issue occurred because initially the x-axis vector was defined as the intersection vector of the top plane and right plane of the table. Additionally, the y-axis vector was defined as the intersection of the top plane and front plane of the table. The issue with this method is that the table planes are not exactly perpendicular to one another due to manufacturing and measurement tolerances, thus making the x-axis and y-axis not perpendicular. The chosen solution was to use the normal vector of the top plane as the z-axis and to continue calculating the x-axis as the intersection vector of the top plane and right plane of the table. Rather than calculating the y-axis in the same manner as before, the cross product between the z-axis and x-axis was taken and used as the y-axis vector.

6 Conclusion

This section discusses the final results of the thesis and some recommendations for future work.

6.1 Final Results

This thesis presents a novel solution for improving robot cell calibration by calculating an accurate workspace coordinate frame. Accurate robotic operations are fundamental in industrial robot cells and there is a lack of calibration methods to choose from to best fit all robotic applications. The solution presented in this paper was executed through calculations in a MATLAB algorithm from positional data points obtained using a laser tracker. To achieve this goal, the laser tracker was used to record the 17 required positions for the MATLAB

algorithm to calculate the frame transformations between itself and the robot and between itself and the work space table. Using these transformations, the frame transformation between the robot and the table were calculated. The laser tracker algorithm was compared to the 3-point method in areas including accuracy, execution time, repeatability and varying table geometry to better understand the method's robustness. The results showed that the laser tracker algorithm was more accurate, having a maximum error of approximately 1 mm whereas the 3-point method reached a maximum error of approximately 10 mm. It was also more precise than the 3-point method, as shown by the calculated standard deviations, with the exception of the y coordinate. The laser tracker algorithm also had a faster execution time compared to the 3-point method, was accurate on tables with varying geometry and is relatively safe to execute by the robot operator. The benefit of this work is that it reduces the amount of down time a robot may have in a manufacturing environment due to requiring fewer manual touch-ups by a robot operator, making it more productive and thus profitable.

6.2 Future Work

New SMR tooling: One area for future work on this method would be to add SMR nest tooling that makes it easier to take measurements on any side of the table, regardless of the laser tracker position. The available tooling in this thesis made it so that the laser tracker could only take data from the close side of the table. This meant that all movements on the table relative to the origin would be in the negative x and y directions. Improving this tooling is not crucial but it would improve the efficiency when creating targets and programs on the workspace table.

References

[1] Packaging Strategies. *Go from Manual to Automated Palletizing with Configurable Robotic Cell*. Oct. 2018 [Online]. URL: https://www.packagingstrategies.com/articles/90766-go-from-manual-to-automated-palletizing-with-configurable-robotic-cell.

[2] Todd Szallay Russell DeVlieg. "Applied Accurate Robotic Drilling for Aircraft Fuselage". In: *SAE International Journal of Aerospace* 119.1 (Sept. 2010 [Online].). DOI: https://doi.org/10.4271/2010-01-1836.

[3] Bright Hub Engineering. *What are Manual Robots?* Dec. 2009 [Online]. URL: https://www.brighthubengineering.com/robotics/58965-manipulation-robotic-system-manual-type-robots.

[4] RoboDK. *Offline Programming*. Mar. 2020 [Online]. URL: https://robodk.com/offline-programming.

[5] Zvi S. Roth, Bahram Ravani, and Benjamin W. Mooring. "An Overview of Robot Calibration". In: *IEEE Journal on Robotics and Automation* 3.5 (Nov. 1987 [Online].). DOI: 10.1109/JRA.1987.1087124.

[6] Wenzeng Zhang, Xiande Ma, and Leqin Cui. "3 Points Calibration Method of Part Coordinates for Arc Welding Robot". In: *Intelligent Robotics and Applications, First International Conference, ICIRA*. 2008 [Online]. URL: https://www.researchgate.net/publication/221104984_3_Points_Calibration_Method_of_Part_Coordinates_for_Arc_Welding_Robot.

[7] Yuye Cai et al. "Easy industrial robot cell coordinates calibration with touch panel". In: *Robotics and Computer-Integrated Manufacturing* 50 (Apr. 2018 [Online].). DOI: https://doi.org/10.1016/j.rcim.2017.10.004.

[8] P. S. Rocadas and R. S. McMaster. "A Robot Cell Calibration Algorithm and its Use With a 3D Measuruing System". In: *IEEE International Symposium on Industrial Electronics*. 1997 [Online]. URL: https://ieeexplore-ieee-org.uml.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=651779.

[9] Petter Johansson. "Robot Cell Calibration using a laser pointer SME Robot". M.Sc. thesis. SE-221 00 Lund, Sweden: Department of Machine Design, Lund University, 2007 [Online]. URL: https://www.lth.se/fileadmin/maskinkonstruktion/robotteknik/arkiv/2007/msc.pdf.

[10] Morten Andre Astad et al. "Vive for Robotics: Rapid Robot Cell Calibration". In: *The 7th International Conference on Control, Mechatronics and Automation*. 2019 [Online]. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2649122/Astad_ICCMA_2019.pdf?sequence=1.

[11] HexagonMI. *Laser Tracker Systems*. Nov. 2021 [Online]. URL: https://www.hexagonmi.com/en-US/products/laser-tracker-systems.

[12] MetrologyWorks. *1.5 Laser Tracker Ball Probe SMR Silverback Series – Standard Accuracy*. Feb. 2022 [Online]. URL: https://www.metrologyworks.com/product/1-5-laser-tracker-ball-probe-smr-standard-accuracy-silverback/.

[13] *Interface Programmers Manual*. 3.8. Leica Geosystems AG. 5035 Unterentfelden, Switzerland, 2017, pp. 313–344.