CODEN:LUTEDX/(TEIE-5518)/1-75/(2024)

Industrial Electrical Engineering and Automation

Software-Driven Calibration Method for a Pick-and-Place Process

Oskar Joelsson
Rasmus Kalvenes

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

Software-Driven Calibration Method for a Pick-and-Place Process

Oskar Joelsson and Rasmus Kalvenes

4th June 2024

M.Sc. Thesis

Department of Industrial Electrical Engineering and Automation
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

Machine calibration is important for optimal performance throughout the lifespan of industrial processes. A prime example is the pick-and-place process, where robots must manipulate objects accurately. Currently, this process is manually calibrated, which can be a complex practice. This thesis studies a software-based approach to error detection within the process, aiming to reduce the need for manual calibration.

By utilizing existing sensors within the process, the method identifies translation and rotational errors within the system. The proposed approach involves measuring and comparing the perspective of mechatronic devices observing the same object. Thus, the differences in perspectives can be identified and compensated for.

The findings indicate promising results, demonstrating that the process could be calibrated with an accuracy of a few millimeters using the proposed method. Moreover, the method is cost-effective and quick to execute.

In summary, this thesis demonstrates the possibility of implementing a software-based approach for calibrating the pick-and-place process. This approach has been demonstrated to be accurate, quick, and simple to use, making it a potential alternative to manual calibration.

Sammanfattning

Maskinkalibrering är viktigt för optimal prestanda under industriella processers hela livslängd. Ett utmärkt exempel är en pick-and-place process, där robotar måste hantera objekt med hög noggrannhet. Denna process kalibreras idag manuellt, vilket är arbetsmässigt besvärligt. Därför studerar denna uppsats en mjukvarubaserad metod för feldetektering inom processen, med målet att minska behovet av manuell kalibrering.

Genom att använda befintliga sensorer inom processen identifierar metoden translations- och rotationsfel inom systemet. Den föreslagna metoden innefattar att mäta och jämföra perspektiven hos mekatroniska enheter som observerar samma objekt. På så sätt kan skillnaderna i perspektiv identifieras och kompenseras för.

Resultaten visar lovande resultat och demonstrerar att processen kan kalibreras med en noggrannhet på några millimeter med den föreslagna metoden. Dessutom är metoden kostnadseffektiv och snabb att utföra.

Sammanfattningsvis visar denna uppsats att det är möjligt att implementera en mjukvarubaserad metod för att kalibrera pick-and-place processen. Metoden har visat sig vara noggrann, snabb och enkelt att använda, vilket gör den till ett potentiellt alternativ till manuell kalibrering.

Acknowledgements

This thesis project was completed in the spring of 2024 and marks the conclusion of our studies at Lund University. Throughout the project, we, the authors - Rasmus Kalvenes and Oskar Joelsson - contributed equally to the project.

We would like to extend our sincere gratitude to Peter Valdt and Erik Ottosson at B&R Industrial Automation for offering us this opportunity to work on this Master's thesis. Their guidance and dedication have been invaluable and inspiring throughout the project. Despite Erik Ottosson transitioning to a role at the Department of Industrial Electrical Engineering and Automation (IEA) at Lund University midway through our project, he continued to provide us with valuable insights and comments.

Additionally, we would like to thank Roland Riepl and Sebastian Brandstetter from B&R's research and development department in Austria. Their initial concept for this thesis project and their ongoing support have been crucial to our success.

Our appreciation also goes to Morten Hemmingsson, our supervisor at the Department of IEA, for his guidance and support, as well as to Ulf Jeppsson, our examiner, for his valuable feedback.

We are grateful to Ádám Péter, Samuel Werner, and Alexander Lenander for their excellent opposition, which provided us with valuable comments on our work. Lastly, we extend our thanks to Thomas Noerby and Anders Welander for dedicating their time to helping us implement a B&R robotics system.

Notations and Symbols

Latin letters

x - x-coordinate
y - y-coordinate
z - z-coordinate

x_c - x-coordinate in camera coordinate system
y_c - y-coordinate in camera coordinate system

X - Translation along x-axis in global coordinate system
Y - Translation along y-axis in global coordinate system
Z - Translation along z-axis in global coordinate system

Greek letters

α, β, γ - Euler angles
θ - Camera angle μ - Pixel conversion number

Theory

Euclidian space - This is a coordinate system with orthogonal x, y and z axes representing the real world.
TCP - Tool Center Point: This represents the tool piece held by robotic arms. It also represents the position the tool has in the robot coordinate system in order to operate with it.
Homography - This is a transformation between two planes.
Projective transform - This is a visual transformation of images using homography to make pictures look like they were taken from a different angle.
Pivot Point - The point around which an object is rotated.

Contents

1 Introduction

1.1 Background

Calibrating machines is crucial for various hardware applications, as it increases accuracy and reliability, important aspects for delivering high-quality products. Presently, machines require manual calibration, which is labor-intensive. However, if manual calibration is neglected to reduce costs, it will negatively affect the quality of the products. Nonetheless, if there was a way to calibrate accurately while reducing the labor intensity, this would greatly benefit producers. Finding a universal solution to this issue is complicated due to the complexity of production systems. However, as industries today are pushing towards Industry 4.0, collecting ever more data, this data can be leveraged to automate calibration processes. For instance, data can be collected as measurement data from cameras and encoders. This led to the idea of a software-driven calibration process, which aims to alleviate the drawbacks of manual calibration. By utilizing sensors already in place for collecting data on positions and rotations of the mechatronic devices, calibration can be done while reducing the need for manual intervention.

The pick-and-place process is an integral part of many industrial processes and it is a process that is highly dependent on accurately calibrated hardware. This process involves, as its name suggests, picking up an object and relocating it. The pick-and-place process can be seen in Figure 1.1. Moreover, as cameras have advanced, and been integrated with complex image processing techniques, the system has become increasingly autonomous. Today, objects that enter the system are detected by the camera, analyzed, and then predicted to end up at a point downstream, where a robot picks them up. However, for the robot to successfully pick the object up, it has to be calibrated correctly. Therefore, accurate calibration is crucial for this process. In 2020, a paper from China University of Mining and Technology [1] proposed a calibration method utilizing data from already integrated sensors. The results were promising, improving accuracy compared to traditional methods, with cumulative errors below 4 mm. Although not fully autonomous, it demonstrates the potential of leveraging available sensors for calibration.

1.2 Motivation

B&R Industrial Automation, which focuses on automation solutions in various industries, wishes to implement a software-driven calibration solution for its hardware systems, particularly its pick-and-place process. According to B&R, manufacturers are often challenged by having to manually calibrate their pick-and-place process, which can be complex. Describing manual calibration precisely, however, is challenging because it depends on the specific process and machines that are involved. For example,

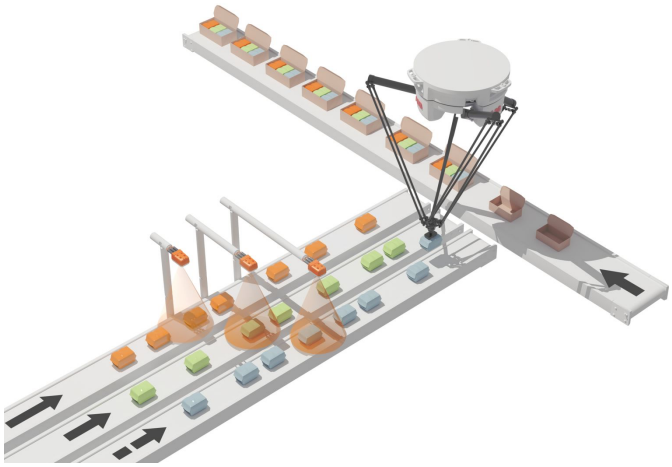


Figure 1.1: A visual representation of a pick-and-place process. [2]

pick-and-place processes vary greatly in accuracy and complexity depending on which industry they are applied to. Furthermore, the calibration procedure can vary depending on the machine being calibrated. For example, robots require calibration of their kinematic structure, while cameras require calibration of their settings. However, both require calibration of their positions and orientations. In the pick-and-place process illustrated in Figure 1.1, three types of mechatronic devices - cameras, conveyor belts, and a robot - have to be calibrated to ensure accurate operations. Occasionally, these devices may be installed or transported incorrectly, affecting their positions and orientations, and causing a need for physical adjustments. To find these errors a service technician has to perform a variety of measurements to identify the errors. The variety in measuring alignments and orientations for all devices, and also adjusting for them, is what makes manual calibration complex. Furthermore, calibrations require a skilled technician who is knowledgeable in the field [3], which can be expensive if a technician has to be hired to calibrate several devices. It is especially true if one also includes potential travel expenses for the technician. Therefore, simplifying the calibration process would benefit producers by enabling individuals without specialized knowledge to perform the calibration. This reduction in complexity is the primary motivating factor for this thesis.

This report will focus on the pick-and-place process that B&R proposed, where a demand for a solution is clear. The pick-and-place process is a widespread industrial process, and an improved calibration method would therefore affect many industries.

Additionally, this report will take a similar approach to the paper from China University of Mining and Technology [1], as this was proven to work. However, there is a need for added features to advance the area, which will act as a second motivating factor in this thesis.

1.3 Objectives

The objective of this Master's thesis project is to reduce the necessity for manually tuning the pick-and-place process by implementing a software-driven calibration solution.

1.4 Limitations

There are a few necessary limitations within the scope of this project. Firstly, the thesis focuses only on the pick-and-place process, thus neglecting all other industrial processes requiring calibration. Secondly, the lack of variability in the test setup limits the project. The test setup only consists of one robot and one conveyor, while industrial processes may involve multiple conveyor belts and robots. Thus, this simplification implies that a universal solution will not be studied. Thirdly, the calibration method should not involve sensors that require cables, as this would hinder the robot's movements. Also, laser techniques for tracking should be neglected as the technology is expensive. These were some of B&R's demands. Lastly, the focus of this project should be on proof of concept to demonstrate that it is possible to reduce the need for manual calibration.

2 Project setup

2.1 The physical setup

The physical setup of the pick-and-place process, consists of a camera, a conveyor belt, a robot, and lastly, a workpiece, which could be any object. The setup can be seen in Figure 2.1.

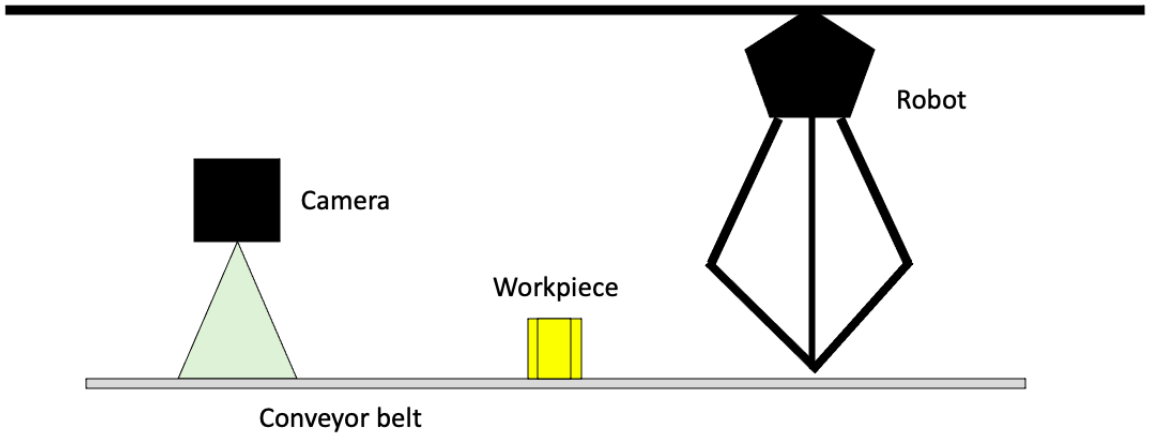


Figure 2.1: The physical setup.

It is this physical setup to which the calibration will be applied. In the following sections, the components of the setup and how the calibration will take place will be discussed.

2.2 Components and software

2.2.1 Components

In this section, the 3 machines that were brought up in the previous section - the camera, the conveyor belt, and the robot - will be introduced.

Robot

The robot that is used in this project is a delta robot, series D4 from Codian. It has a weight load capacity of 3 kg and a precision of 0.1 mm [4]. The delta robot is used for pick-and-place solutions where four joints are used to move the tool in three dimensions. This robot utilizes four arms that work to move a plate, with the possibility of attaching a variety of tools. The D4 series moves the tool center

point freely in three dimensions, but can also rotate its tool point to pick up radially misaligned objects.

Camera

The camera that is used in the physical setup is B&R's Smart Sensor, which combines image processing with real-time communication. It is equipped with the possibility to do image analysis operations, e.g., edge detection and QR detection. In this manner, sophisticated algorithms can easily be integrated into processes. It also enables the possibility to incorporate filters, e.g., a Sobel filter that enhances edge detection.

Conveyor belt

The conveyor belt is run by a B&R synchronous motor with a maximum speed of 9000 rpm and a maximum torque of 9.2 Nm [5]. This motor is equipped with an encoder so that position, velocity, and acceleration can be measured when using the conveyor belt.

2.2.2 Software

In this section, the software for implementing the autonomous calibration solution will be introduced.

Automation Studio

Automation Studio is a software platform for industrial automation projects developed by B&R. The program enables the user to integrate the programming of PLCs, HMI design, motion control systems, and vision systems in a unified space. It also has the possibility to produce projects using different programming languages, e.g., C, C++, and Structural Text.

mapp

mapp is B&R's own written software that handles a wide selection of tasks. This will be used as a basis for the software implemented in this project, where it is used for robotics, conveyor motion and visual feedback.

2.3 Important assumptions

Throughout the project, several assumptions will be made to succeed with the project. These assumptions are summarized below:

- The camera's view does not include the robot's workspace.
- The conveyor belt is straight, i.e., the bending of the belt is neglected.
- The conveyor belt is parallel to the floor.
- The conveyor belt is equipped with an encoder.
- The physical setup exists in a global coordinate system, but only relative errors between devices are relevant.

In the physical setup introduced in Section 2.1, the camera does not have a view over the robot's workspace. This can be seen in Figure 2.1, where the green area is the camera's view. Assuming that the bending of the belt can be neglected is desirable to predict the future position of the workpiece. If the belt is not straight, the workpiece will travel non-linearly, and its position will be difficult to predict. The third assumption is that the conveyor belt is equipped with an encoder. This is important to know how far a workpiece has traveled and where it is at a given point in time. The fourth assumption dictates that only relative errors of parts are needed. The global coordinate system is decided by factory blueprints dictating devices' positions and orientations. There may exist errors in all the devices, but their relation to the global coordinate system will not be tested, only the relative errors between devices will be extrapolated. This is because there are no sensors for the global coordinate system available. Only sensors that can sense relative position to other parts. For example: camera to conveyor or Tool Center Point (TCP) to workpiece.

2.4 Navigating the reference frames

There are several reference frames in the physical setup of the pick-and-place process. First of all, there is the global reference frame. This intends to capture all devices within the given environment. In the case of this project, those devices are the camera, workpiece, conveyor belt, and, lastly, the robot. Furthermore, all of the devices mentioned, within the global reference frame, also have their own local coordinate frames. It is through establishing these reference frames, and how they are related to each other, that one can achieve an accurate working system.

2.5 Finding relations in the coordinate systems

The global coordinates will be used to find the relations between the devices in the setup. As mentioned in the assumptions, it is only the relative errors that are of

interest, not whether the robot or camera is correctly installed relative to the global coordinate system. Therefore, the errors will be derived using the robot's reference frame relative to the camera's. Here it is important to note that these local coordinate systems are incorrect to some degree, since nothing is installed perfectly, but as long as the relation between the systems is known, the two can work together.

Each reference frame is divided into the (x, y, z) dimensions.

- $O_g = (\hat{x}_g, \hat{y}_g, \hat{z}_g)$ is the global reference frame.
- $O_c = (\hat{x}_c, \hat{y}_c, \hat{z}_c)$ is the camera's reference frame.
- $O_b = (\hat{x}_b, \hat{y}_b, \hat{z}_b)$ is the conveyor belt's reference frame.
- $O_w = (\hat{x}_w, \hat{y}_w, \hat{z}_w)$ is the workpiece's reference frame.
- $O_r = (\hat{x}_r, \hat{y}_r, \hat{z}_r)$ is the robot's reference frame.

In a perfect world, O_c, O_w, O_b and O_r have parallel unit vectors. This, however, is not the case, which is illustrated in Figure 2.2.

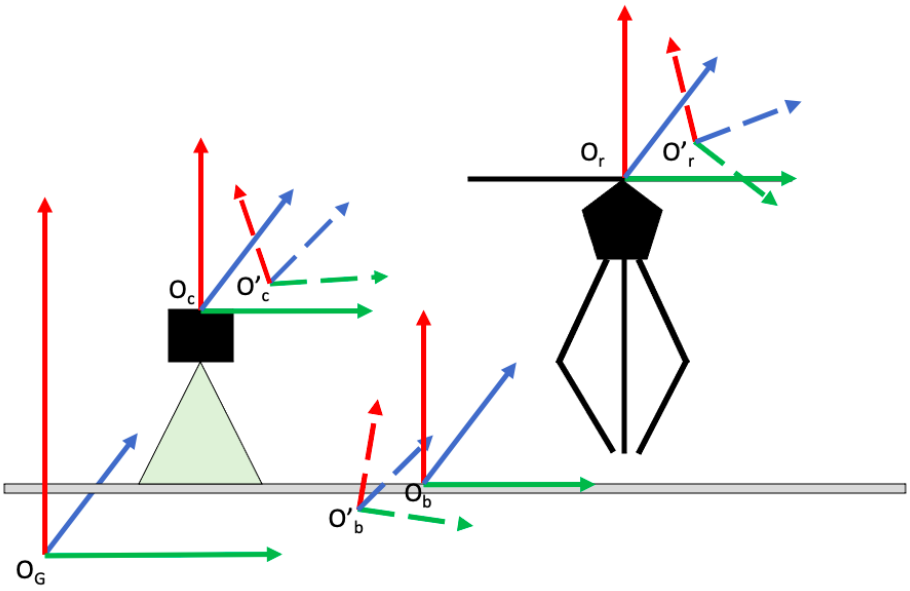


Figure 2.2: All local coordinate systems should be parallel with the global coordinate system, however, in reality, they are both translated and rotated from where they should be. This is represented by the coordinate systems with dashed lines.

By compensating for the misalignments, one can achieve an accurate system. Finding these misalignments is where it gets tricky, however. Since the camera can not see the robot, their reference frames can not be connected relative to each other directly. Instead, the robot's reference frame can only be compared to the workpiece, and the workpiece is compared to the conveyor through the camera. This becomes a chain, connecting the reference frames. How one connects the chains is not straight forward and there are many possible solutions. In this project, however, the chain of reference systems will be realized through the following workflow.

- First step; derive the orientation of the workpiece in relation to the camera. Since the images are in 2D, only the orientation of $(\hat{x}_w, \hat{y}_w)$ relative to $(\hat{x}_c, \hat{y}_c)$ can be found. There is an assumption here that, while $\hat{z}_w$ is not aligned with $\hat{z}_c$, the mapp Vision software will be able to extrapolate accurate data as long as the workspace is in field of view. This filtering process will be explained in Section 3.1. The conveyor belt's height is assumed to remain constant and is seen as a base height, which further calculations will depend on. This assumption will not affect the error estimation at a later stage, which will be discussed in Chapter 3.
- Second step; derive the conveyor belt's reference system in relation to the camera's. This is done to estimate the path on which the object will travel along. This is also detected through mapp Vision.
- Third step; derive the robot's coordinate system in relation to the workpiece's coordinate system, which is done by mapping out where the workpiece is found in the robot's reference frame O_r.
- Fourth step; compare the position and orientation of how the robot viewed the workpiece to how the camera viewed the workpiece. By doing this, the relation between the camera and the robot can be derived. This is further discussed in Chapter 4.

It is through these four steps that one can derive the chain of reference systems, and therefore, accurately trace the workpiece.

2.6 Process overview

In the previous section, Section 2.5, finding the relations between the different reference systems were discussed. In this section, the discussion will be summarized into a process overview, which the concepts of this project will be based upon. This overview is concluded in Figure 2.3.

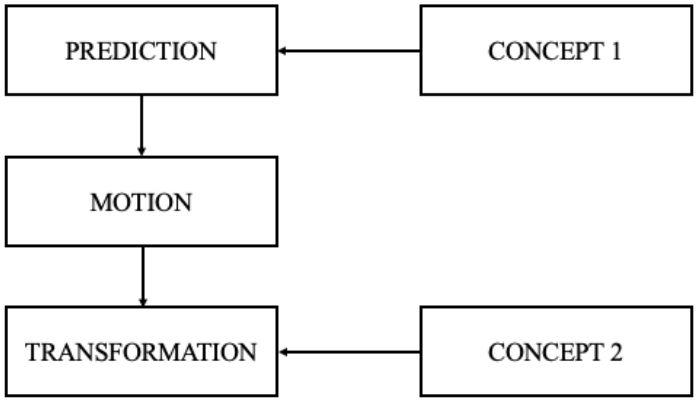


Figure 2.3: An overview of the process of calibrating the robot.

In the first part of the suggested process, which will be referred to as PREDICTION, the camera will find points on the workpiece and the conveyor belt's direction. Since

the conveyor belt is assumed to be straight and is equipped with an encoder, the future position of a point can be predicted, hence its name. For this, a concept is needed, in order to get the direction of the conveyor belt. All concepts are presented in Chapter 6. Using the position-controlled conveyor belt, the workpiece can be moved along the conveyor belt at a known distance and speed. This part is called MOTION. After the workpiece is moved to the desired distance, where the robot may reach the workpiece, a concept is needed to move the robot to the same points as the predicted points. If the system is incorrectly calibrated, then the robot will be off by some distance. This is where the TRANSFORMATION part is introduced. This part will derive a transformation between where the points on the workpiece are found by the robot and the prediction, which can be used to implement our calibration. For this, a concept is needed. Now, in theory, every workpiece can receive a calculated future position, using the conveyor belt's direction, and all errors can be compensated for using the known transformation. With that said, the process overview is concluded and the following chapters will describe this process in more detail.

3 Path estimation

3.1 Projective transformation

As mentioned, the setup has a camera in the beginning that will observe the objects entering the process. Depending on where the object is situated under the camera, it will have to be correctly projected, so that its position can be found. To succeed with this, projective transformations will be used. Since a 2D camera is available, 2D projective transformations will be used, where the height, according to the mechanical drawings, is assumed to be constant and correct. To project the camera's image to a grid, homography will be used. When the image has been projected to a grid, one can perform image analysis and derive accurate coordinates. Homography, or projective transformation, for the 2D case, has the following structure [6],

$$w \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{c} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ c_1 & c_2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{3.1}$$

By utilizing the $\mathbf{M}$ matrix, one can alter any image coordinate that is input to (3.1) and create a new image. By inputting points in a coordinate system, a new set of points in the same coordinate system are created. The transformations can also be used to move a coordinate system and create a new coordinate system. The M matrix consists of three parts: the square matrix $\mathbf{R}$, the column vector $\mathbf{t}$, and the row vector $\mathbf{c}$. The $\mathbf{R}$ matrix performs affine transformation, which creates rotation, scaling, and skewing. The $\mathbf{t}$ vector creates translation. The $\mathbf{c}$ vector distorts the image taken with a perspective view and creates a grid view for flat surfaces, in this case, the conveyor. In theory, the image is distorted in a third dimension, then projected back down to a 2D image with the scalar w at height 1.

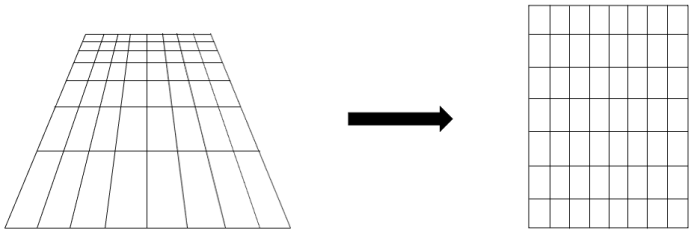


Figure 3.1: An example projective transform from a perspective plane into a grid plane.

3.2 Linear regression

Linear regression is a method for finding a linear relationship between data points. More specifically, it is the linear relationship between one dependent and one independent data point. Furthermore, simple linear regression is the case where there is only one independent variable. The general approximated regression line can be formulated as [7]

$$\hat{y} = \beta_0 + \beta_1 x. \tag{3.2}$$

The goal is to estimate the regression coefficients β_0 and β_1, for a set of n data points $\{\{x_i, y_i\}, i = 1, 2, ..., n\}$, so that can be predicted for some x. This can be done by finding a line that minimizes the sum of squared residuals from the data points, which is done by using the least-squares method. β_0 and β_1 are then found using the equations from [7]:

$$\beta_0 = \hat{y} - \beta_1 \hat{x} \tag{3.3}$$

$$\beta_1 = \frac{\sum_{i=0}^{n}(x_i - \hat{x})(y_i - \hat{y})}{\sum_{i=0}^{n}(x_i - \hat{x})^2} \tag{3.4}$$

With the use of simple linear regression, where the data points are the previous x and y positions of the workpiece, the future position and path on the conveyor belt can be estimated. This is under the assumption that the conveyor belt is straight so that the object travels linearly along the belt. Simple linear regression will be implemented in Subsection 6.1.1.

4 Error estimation

4.1 Deriving the transformation matrix

The error estimation part consists of finding a rule for how the robot should compensate for each movement to pick up objects accurately. For this task to be possible it is assumed that a future position of the workpiece, one that is situated near the robot, can be predicted using the camera. If this holds, then to find the error between the predicted position of the workpiece and where the robot finds it, one has to compare the points, which is seen in 4.1.

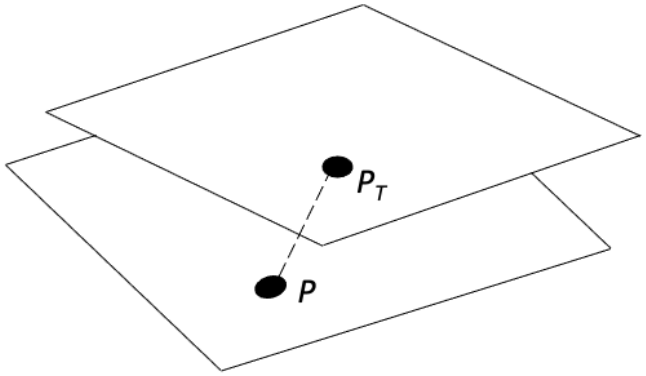


Figure 4.1: The error between the predicted (bottom) and the actual (top) plane found by the robot.

Each object on the predicted plane exists somewhere on the correct plane, the one that the robot finds, but with a different orientation and a translation. There should not be alterations in scale due to the object existing in Euclidean space, however, there may exist some small variations due to measurement errors. Meaning the distances between the points should be the same on the predicted plane and the transformed plane as they are measured from the same object.

The error is found using a homogeneous transformation matrix, which has the following structure [8],

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{4.1}$$

This is similar to (3.1) but without the c vector and w scalar. This is because this matrix does not project in an extra dimension. In this matrix, $\mathbf{R}$ is a 3x3 matrix consisting of the rotations with possible scale errors due to measurements, and $\mathbf{t}$ is a

3x1 vector that includes all translations in 3 dimensions. To solve **H**, which has 12 unknown variables, 9 variables in **R**, and 3 in **t**, one needs 12 independent equations. Then the problem can be stated as

$$\begin{aligned} x_i' &= r_{11}x_i + r_{12}y_i + r_{13}z_i + t_1 \\ y_i' &= r_{21}x_i + r_{22}y_i + r_{23}z_i + t_2 \\ z_i' &= r_{31}x_i + r_{22}y_i + r_{33}z_i + t_3 \end{aligned} \tag{4.2}$$

where each point pair (p_i, p_i') gives three equations. Therefore, four non-colinear point pairs are needed to get a well-determined system. Each point p_i is determined by the camera and therefore has constant height, i.e., the base height, z. An approximation of **H** can be derived using the least-squares method. By including n number of non-collinear point pairs, the equation 4.2 can be re-written in matrix form as

$$\begin{bmatrix} x_0' \\ y_0' \\ z_0' \\ \vdots \\ x_n' \\ y_n' \\ z_n' \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & z & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_0 & y_0 & z & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_0 & y_0 & z & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & z & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_n & y_n & z & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_n & y_n & z & 1 \end{bmatrix} \begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \\ t_1 \\ r_{21} \\ r_{22} \\ r_{23} \\ t_2 \\ r_{31} \\ r_{32} \\ r_{33} \\ t_3 \end{bmatrix}, \tag{4.3}$$

which can then be re-written as

$$b = Ax. \tag{4.4}$$

A problem here occurs, which is that A is not a square matrix. If $n > 4$, A will be an over-determined system. To resolve this issue, one can use the pseudo inverse, A^+. The pseudo inverse is an approximation of A^{-1} and can be used to solve the equation (4.4). In general, A^+ is defined using the Singular Value Decomposition (SVD). SVD, a mathematical factorization technique for decomposing a matrix into three other matrices, and he equation is defined as [9]

$$A = U\Sigma V^H, \tag{4.5}$$

where U and V are unitary matrices of size $m \times m$ and $n \times n$, respectively. Σ is a non-negative, real-value diagonal matrix, of size $m \times n$. H is the conjugate transpose. This factorization is advantageous to use because of the products' properties, such as

U and V being unitary matrices. Computing efficiency and mathematical handling are improved, compared to the A matrix, which enables usage of more advanced operations, e.g., the pseudo inverse. Nevertheless, with the values U, Σ and V^H, A^+ can be written as [9]

$$A^+ = V\Sigma^+ U^H. \tag{4.6}$$

Notice the matrix Σ^+, which is the pseudo inverse of Σ. Σ is a diagonal matrix, and to derive the pseudo inverse, Σ has to be transposed and the non-zero diagonal values have to be replaced with their inverses. Furthermore, both V^H and U in Eq. 4.5 have been conjugate transposed.

Now, using A^+, the approximation of x, $\hat{x}$, can be solved as

$$\hat{x} = A^+ b. \tag{4.7}$$

$\hat{x}$ is a 12x1 matrix, but, as seen in (4.3), it contains the approximated elements of the homogeneous transformation matrix $\mathbf{H}$. Therefore, one can rearrange the elements in $\hat{x}$ into $\mathbf{H}$. Since $\mathbf{H}$ is approximated, it has an error, but by using more non-collinear point pairs, one will achieve more accurate results. Although one can derive a transformation matrix using only four non-collinear point pairs, this might result in an unsatisfactory transformation matrix that is not accurate enough.

4.1.1 Centering the predicted plane

To get the correct difference between the predicted plane and the one that the robot finds, the predicted points have to be centered around the origin. This is so that the rotations occur around the predicted plane and not somewhere else. This is done by deriving the centroid of all predicted points and then subtracting the centroid from all predicted points. The centroid has to be subtracted from the correct points, found by the robot, as well.

4.1.2 Point transformation in MATLAB

By using the method for error estimation presented in Section 4.1, in MATLAB, one can successfully transform a point into another. This is done by using the function *pinv()* in MATLAB, which calculates the pseudo-inverse matrix, A^+. After determining the transformation matrix, $\mathbf{H}$, a point $\mathbf{P}$ can be transformed into $\mathbf{P}_T$ using

$$\begin{bmatrix} \mathbf{P}_T \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix}. \tag{4.8}$$

4.2 Determining the orientation of an object

Using the method in Section 4.1, an object's predicted position can be transformed into the actual position. However, this is only accurate for points on the transformed plane. The reason for this is that the homogeneous transformation matrix does not have the correct information on the behavior outside of the plane as the camera may only give information in a 2D plane. This can be seen in Figure 4.2, where the blue cube on the predicted blue plane has a different shape on the actual red plane, the black rhombohedron. The red cube signifies the desired transformation where 3D shapes are only rotated and translated

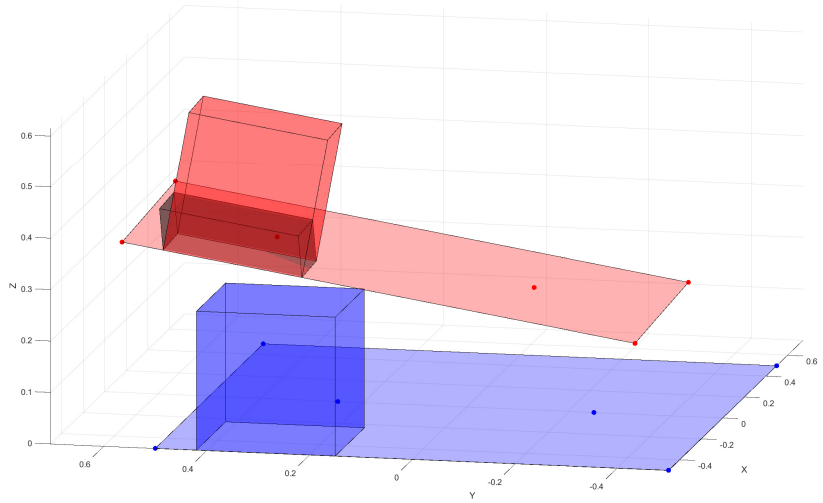


Figure 4.2: Example of an incorrectly transformed cube (black rhombohedron) and a correctly transformed cube (red cube). The shapes are different transformations but with the same input. The blue plane resembles the predicted plane, and the red plane resembles the actual plane, measured by the robot.

This issue can be solved by creating a vector orthogonal to the transformed plane. By using three of the corner points in the transformed plane, two vectors that stretch the plane can be derived. After normalizing these vectors, one can use the cross-product,

$$\begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \times \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}, \tag{4.9}$$

to derive the third vector orthogonal to the plane. Doing this creates the red line in Figure 4.3, which is orthogonal to the transformed plane. The blue line is orthogonal to the predicted plane with an identical calculation. By marking one point using this vector, and three using the plane, for both the predicted space and the transformed space, one can recalculate the transformation matrix using (4.3). This matrix is more accurate for 3D shapes than the previous, as all the points from (4.3) now have full rank. For future reference, this matrix will be labeled as H'.

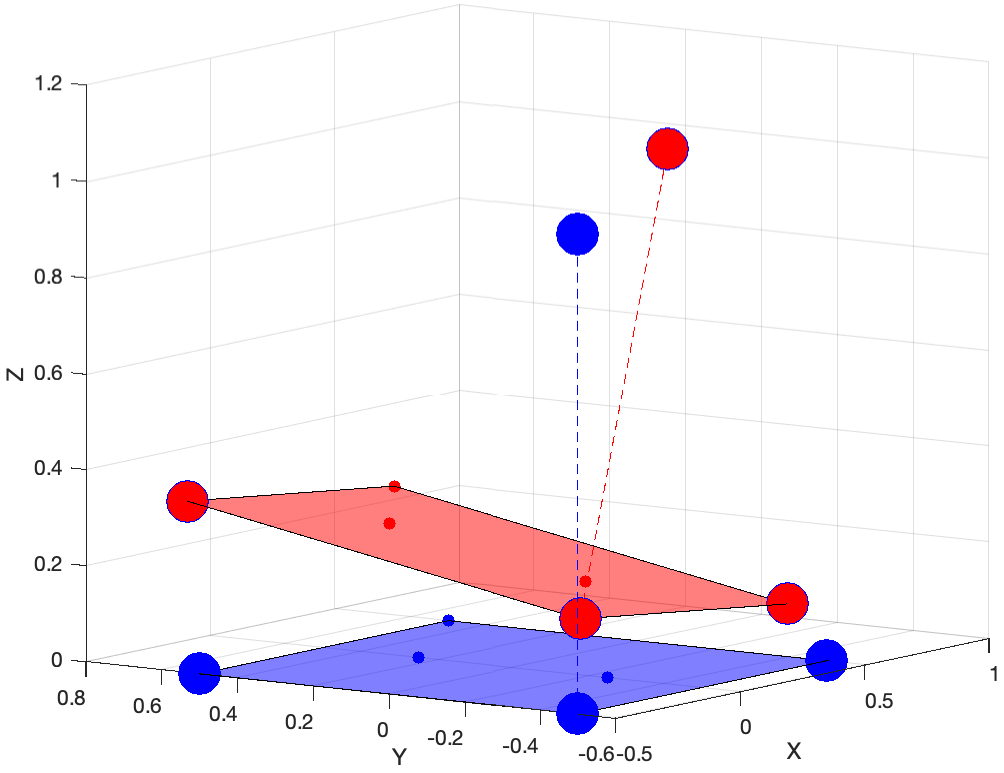


Figure 4.3: The figure displays the orthogonal vectors (dashed lines) to both the predicted plane (blue) and the actual plane (red), measured by the robot. Also, the relative orientation of the planes can be seen.

Using H', several important values can be derived from it. These values will be used for the rule of how the robot should compensate for each movement. As mentioned in eq.(4.1), the translations are described by the t elements in the fourth column of the matrix. They represent how much the robot should compensate for each movement in the x, y, and z axes. However, since the conveyor belt or the robot might tilt in some direction, these axes are not correct. Therefore, one would also need to compensate for the orientation of the original axes. One possible way of doing this is through Euler angles. Euler angles are represented by three angles, usually denoted as α, β and γ. These angles describe the orientation of a rigid body, relative to a fixed coordinate system. Therefore, since the robot will compensate for angles in the orientation, this will be done relative to another frame, e.g., the global coordinate system. The compensation, or rotation in this case, is done in a specific sequence. Applying the rotations is illustrated in Figure 4.4 and can be explained as the following sequence of events. Firstly, the fixed coordinate system is rotated around the z-axis with the angle α. Secondly, a rotation with magnitude β will be performed around the new x'-axis, that has been formed by rotating around the original z-axis. This gives a new z" axis which will be rotated around with the angle γ. Applying this sequence of rotations in the future, the robot can be oriented correctly towards the conveyor belt. To derive the Euler angles from the transformation matrix H', one will only observe the rotation matrix that is part of H'. The rotation matrix consists of the first 3×3 elements in H'. Using the workflow in [10], the Euler angles can be expressed as

$$\beta = -arcsin(r_{31}), \tag{4.10}$$

$$\alpha = arctan2(\frac{r_{32}}{cos\beta}, \frac{r_{33}}{cos\beta}), \tag{4.11}$$

and

$$\gamma = arctan2(\frac{r_{21}}{cos\beta}, \frac{r_{11}}{cos\beta}). \tag{4.12}$$

In Matlab, α, β, and γ can be calculated using the rotation matrix with the command *rot2eul()*.

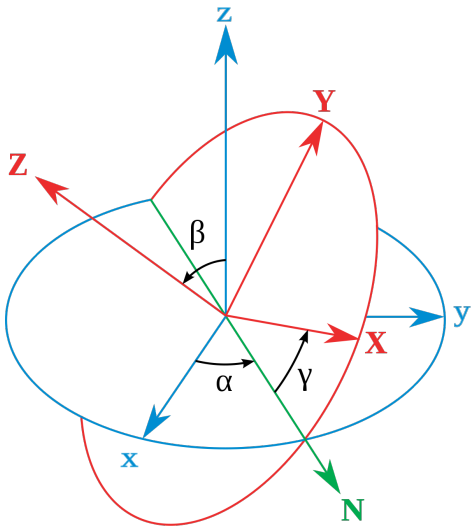


Figure 4.4: The blue coordinate system represents the fixed coordinate system. The red coordinate system is the rotated coordinate system, relative to the fixed. The green arrow is an orbital node, which intersects the points where the rotated xy plane intersects the fixed xy plane. [11]

5 Implementation

This chapter aims to describe the finished product and how the process is set to be implemented into existing software. Once the errors between the path and the robot have been estimated, the transformation matrix can compute the differences between the predicted plane and the one that the robot finds.

5.1 Implementing the calibrated values

With all the previous key parts at hand, discussed in Chapters 2, 3 and 4, it is time to put them together to finalize the calibration process and to implement it. At the start, when a workpiece enters the camera view, it is detected. Thereafter, a prediction is made, where the workpiece will end up using the methods in Chapter 3. However, this prediction is wrong if the orientation of the camera and the conveyor belt is wrong, relative to the global coordinate system. Nonetheless, the prediction is made and compared to where the robot finds the workpiece. By comparing these positions and orientations, methods presented in Chapter 4, derive their relative positions and orientations. From this point, the software configuration of the mechatronic devices is altered so that the errors are compensated for. The new configuration is based on the previous configuration, but where the rotations and translations found in the error are subtracted. The new configuration will now, instead of using point P', use P'_T, as seen in Figure 5.1.

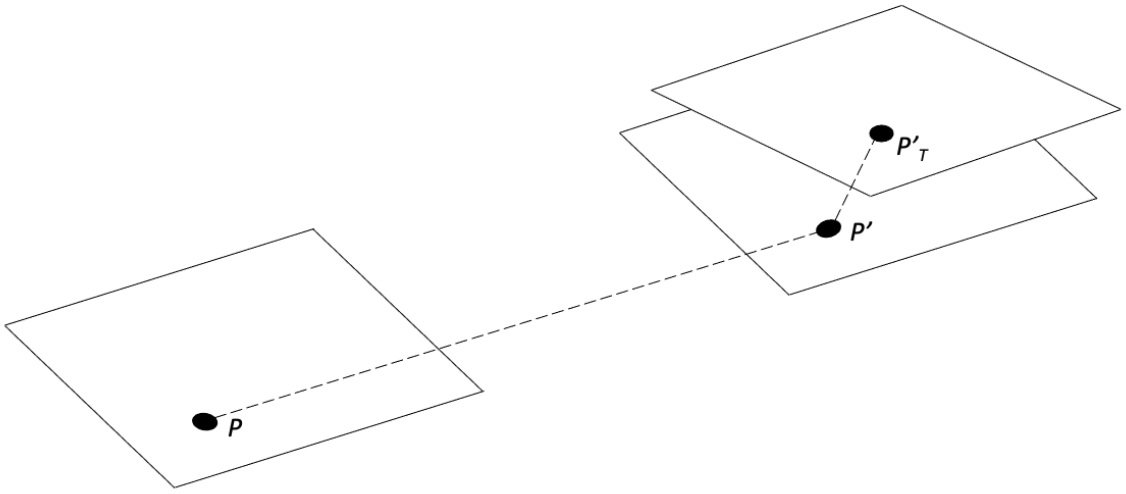


Figure 5.1: Display of all data points used to find the errors. The point P is predicted to end up at P', however, the actual position is at P'_T, the transformed version of P'. The new configuration will move P'_T into P'

Subtracting the errors will be done in the following manner. The current configuration

has a robot frame and the robot should be compensated for the errors. The initial thought might be to subtract the errors from the robot frame. However, applying the errors at the robot frame would cause translations, that are not taken into account, at the conveyor. The reason for this is that the robot frame and the base of the robot, which are positioned far away from the conveyor, will be rotated, and even small rotations at large distances cause unwanted translations. Even though the robot will move perpendicular to the correct plane after being rotated, the unwanted translations cause unacceptable errors. There is also another way of subtracting the errors, which is to create a new frame, unrelated to the robot frame. The errors can be subtracted from the newly created frame, by adding an additional frame, called compensating frame. Figure 5.2 and Figure 5.3 represent how the compensating frame is subtracted from the first frame. The first frame in the hierarchy, after the global coordinate system, will be placed at the conveyor belt since this is where the error estimation is performed. In this way, the compensated rotations occur at the actual pivot point and not at some point far away. Furthermore, this solves the problem of having several input and output conveyor belts, since a new frame can be created for each conveyor belt. The robot can then switch between coordinate frames depending on its task and move according to the current set frame. Switching between coordinate frames can be written into the code for the robot, and switching frames can therefore be done while running the specified process.

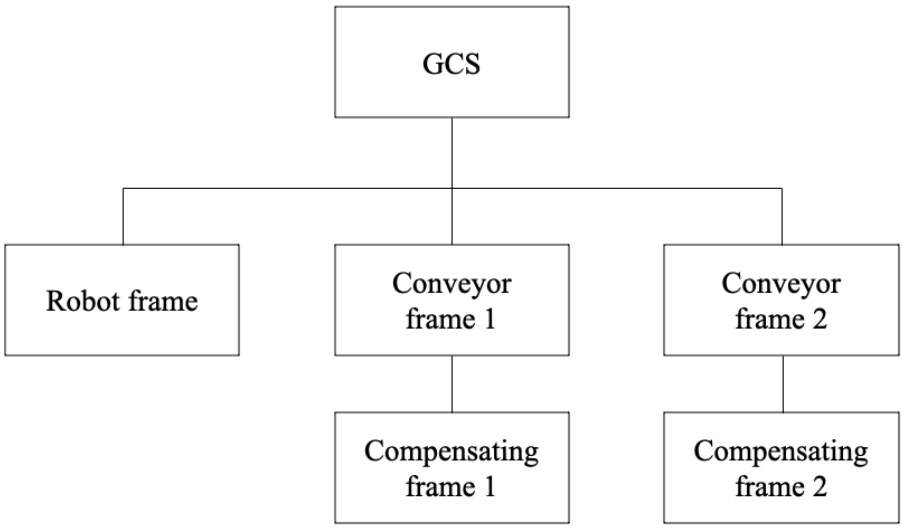


Figure 5.2: Representation of an example of a hierarchy. In this case, there are two conveyor belts that each have their own compensating frame.

Furthermore, note that the order of rotations matter, rotating around x, then y, and then the z-axis has a different result compared to z, then x, and then y-axis for example. This project will only operate in the x-y-z order.

Using the new configuration, the robot can pick up objects from the conveyor accurately despite misalignment in the mechatronic devices. However, this calibration will not consider future errors caused by accidents or load deformations. If sufficiently large misalignments arise, a new calibration will have to be performed using the presented process.

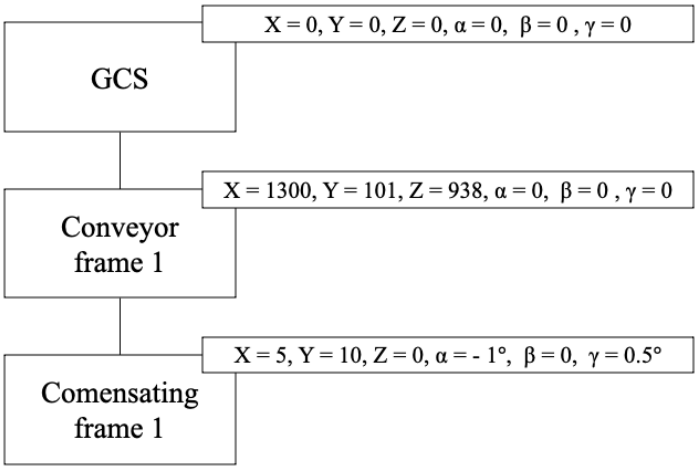


Figure 5.3: Example of how each frame is added. In the global coordinate system (GCS) the conveyor belt frame is placed, and in the conveyor belt frame, the compensating frame is placed. This means that, e.g., the origin of the compensating frame is in $X = 0 + 1300 + 5 = 1305$ mm in the GCS. The origin of the compensating frame is the pivot point of the applied rotations.

5.2 The general case

In a broader context, the issue is two machines not communicating properly due to alignment errors. This problem is universally solvable if there is enough data of full rank to solve for the transformation matrix. Other machine applications may need different tests to get this data, but the principle would be the same no matter the machine. It is only important that the machines share common reference points so that (4.3) in Section 4.1, can be used. Regardless of the difficulties in finding the errors, some robots may be unable to compensate for those within their own constraints. In the case of a pick and place process, the robot needs to be able to move the TCP in 6 dimensions to compensate for all possible errors in the conveyor. However, the robot used in the project setup can only compensate in 4 dimensions, restricting its capability to adapt to the x and y rotations.

6 Concepts for path and error estimation

This chapter will present concepts that were generated by us, and also, a discussion on their advantages and disadvantages. Lastly, final comments will be given on what concepts were chosen for this project.

6.1 Concept generation

6.1.1 Concepts for path estimation

Following the object

For estimating the path of an object in the pick-and-place method, one can take advantage of the camera. By taking several snapshots of an object, the object's change of position can be calculated, and with the historical change of position, one can derive an estimated path for future movement. By using simple linear regression, defined in Equation (3.2), one can estimate this path, in terms of the x and y coordinates. Having estimated β_0 and β_1, it is possible to derive the angle of the path, which is expressed by

$$\theta = \arctan(\beta_1). \tag{6.1}$$

In Figure 6.1, the idea of taking snapshots of an object and anticipating its path can be seen.

There were two ideas on how these snapshots are to be taken. Firstly, one can stop the conveyor belt repeatedly and take snapshots. The main disadvantage of this approach is that the process is choppy. It also takes time to start and stop the conveyor, which one wants to minimize. An advantage of this approach, however, is that the snapshots are taken when the object is completely still. This results in higher accuracy, which is analyzed better. Secondly, one can let the conveyor belt run and take snapshots continuously. This approach is, opposed to the first, faster since it does not have to start and stop repeatedly. This also means that one can take more pictures of the object which in turn might generate more reliable results. On the other hand, taking snapshots while the belt is running, can cause lower accuracy. However, B&R's cameras are built for handling high speed applications. Therefore, the decrease in accuracy might be negligible.

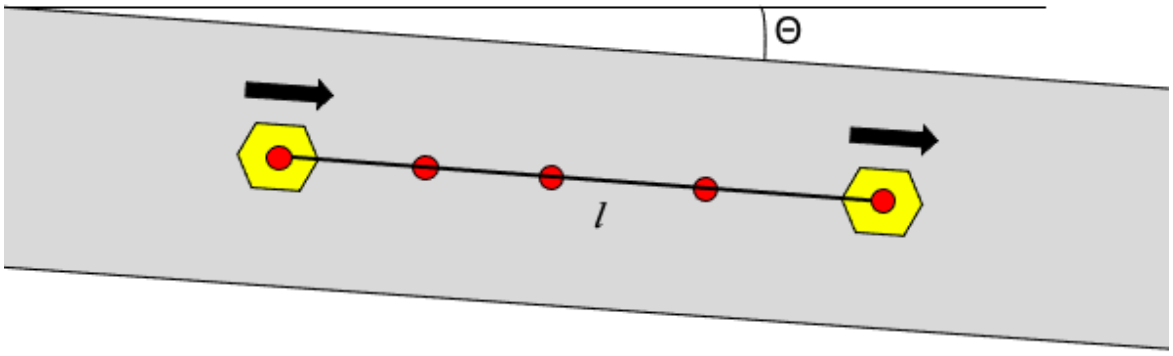


Figure 6.1: The grey area is the conveyor belt. The red dots represent the points where snapshots of the object are taken. Line l represents the path for the object and θ represents the angle error of the conveyor belt, which is causing the object to move diagonally instead of straight forward.

Following the conveyor belt

A second concept, for deriving the path of the object, is to follow the edges of the conveyor belt, and by assuming that the conveyor belt is straight, one will get the estimated path of the object. An illustration of how it will work can be found in Figure 6.2.

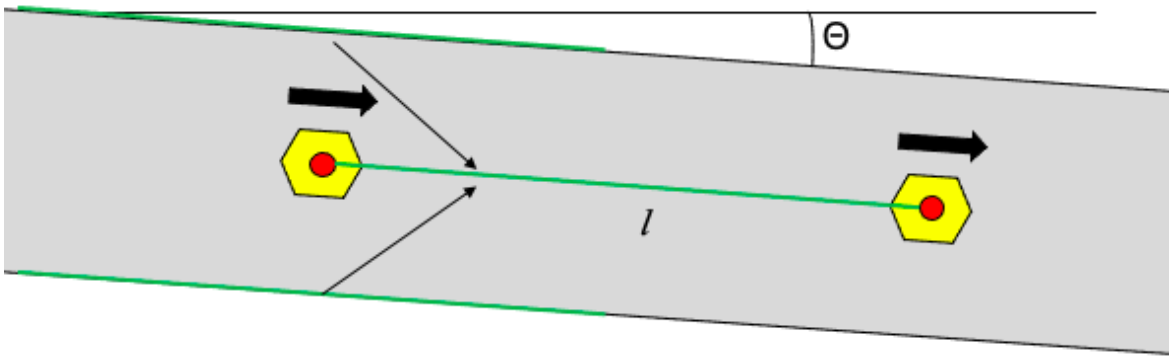


Figure 6.2: Two green lines can be seen on the edges of the conveyor belt. These lines will result in the line l, which is the path of the workpiece.

The benefit of using this method is that the line, that the workpiece will follow, can be detected without having to calculate the workpiece's position in several steps. Also, since the conveyor belt will continue to be straight, this method will only have to be performed once. Furthermore, only one image has to be analyzed instead of several, which was the case for the previous concept, which lessens the computing time. A drawback of this approach is that it is based on the assumption that the conveyor belt is straight. This might not be the case, but since any bending of the conveyor belt will be constant this will be included in the error estimation. Therefore this will not have a large effect on the final result. Another assumption is that the edges are detectable by the camera, which there are methods for, but it might not result in a satisfactory result.

6.1.2 Concepts for error estimation

In this section, concepts for estimating the error of the robot will be presented. It is now assumed that one can calculate the future position of the workpiece using the camera at the beginning of the pick-and-place process. This was done in the previous section, concepts for path estimation. By assuming that the estimated future position is correct, one can compare this position to the robot's position. Moving the robot to the workpiece will result in different coordinates than the estimated ones. It is this error, or difference, that the following concepts aim to find. A visual representation of the error can be seen in Figure 6.3.

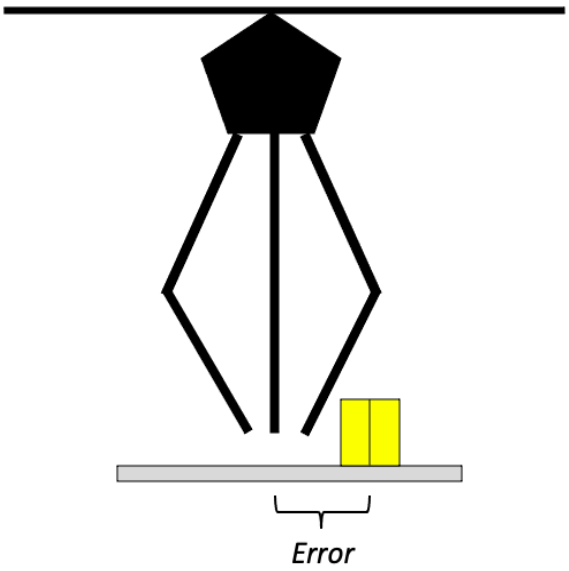


Figure 6.3: Seen from along the conveyor belt, this is an exaggerated view of the error that the robot makes.

Jogging

Jogging is a term for manually moving a robot using a controller. The concept involves this maneuver and is named thereafter. The concept has the following process steps. Firstly, move the robot using the controller to the workpiece. Since the robot is not calibrated correctly, these coordinates will differ from the anticipated ones. This coordinate, seen from the robot's perspective, is then saved. The software can now measure the difference between the saved points and the anticipated points, which is the error. In the future, this error will be compensated for and the process can therefore be seen as calibrated.

There are several disadvantages to using this method. The main one is that it involves human interaction. There will always be a risk of introducing human errors using this approach. The human errors will depend on how complicated it is to jog the robot, which includes difficulty in controlling the robot and having a good view of the robot's workspace. Another disadvantage is that the task can be tedious. The effort increases with the number of robots that have to be re-calibrated.

There are also several advantages to using this method. Most of the process is autonomous, meaning that the software will handle the process. This means that it aims to be easy to use. Another advantage is that there is no need for more mechatronic devices, all devices used are in the pick-and-place setup already. Therefore, it is cheap and easy to implement.

Visual calibration

Visual calibration means precisely what the name states. With the help of an additional camera, the robot will adapt to the error. This is done by estimating the error between the TCP and the object and then compensating for it. There are two proposals for how this can be carried out. The first option is to place a camera on the side of the conveyor belt so that the camera overlooks the robot's workspace. The second option is to attach a camera to the TCP so that it looks down at the conveyor belt. In this way, one can get a visual of the robot's workspace from the TCP's point of view. The two approaches can be seen visually in Figures 6.4 and 6.5.

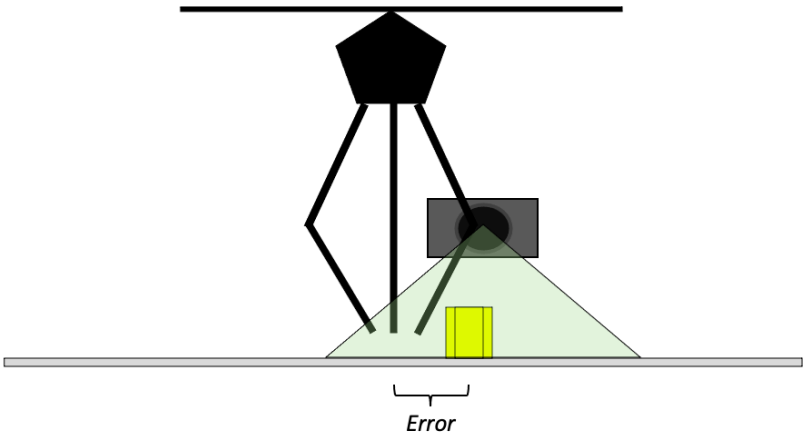


Figure 6.4: A visual representation of the setup with the camera placed to the side of the conveyor belt.

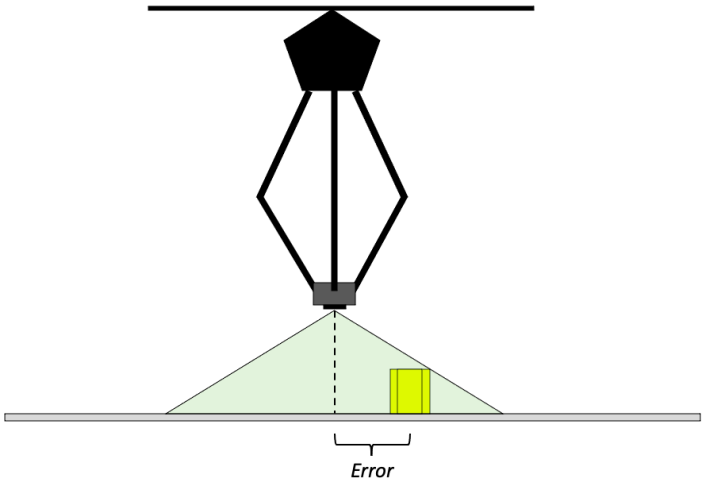


Figure 6.5: Seen from the side of the conveyor belt, the camera is mounted on the TCP.

By placing the camera at the side, one can automate the calibration. The robot will need a tool mounted on the TCP, one that the camera can easily detect so that the error is calculated accurately. A disadvantage of using this method is that it is not scalable. For each robot, one will need an additional camera. There may also be difficulties in finding room for the camera, as it needs a satisfactory view over the robot's workspace. An advantage, however, is its lack of need for human interaction. The idea is that the program will handle all error estimation without human interaction, apart from the operator mounting the tool on the TCP. And the calibration can always be carried out, one only has to mount the tool.

Placing the camera on the TCP will, similar to placing it at the side, also automate the calibration. Using this method one has to mount the camera on the TCP, instead of a visible tool like in the previous method. The disadvantage of using this method, is, firstly, that one has to mount a camera on the robot for each calibration, and for each robot. Secondly, it is not scalable for the same reasons as the previous method of placing a camera at the side. However, it is rather flexible in the sense that, one can attach the camera to TCP when a calibration is necessary, instead of always having a camera in the setup. Additionally, one can share the same camera with each robot, by attaching it to the other robots' TCPs. Precisely as having the camera at the side, the human interaction is limited after changing the tool.

Sensors

In the interest of complete autonomy, it is possible to implement other sensors instead of a camera to detect coordinates in the robot space. This will remove the human interaction required in jogging but with some key differences compared to the other techniques. It is very similar to the camera TCP technique where sensors are placed on the conveyor searching for the TCP. Some sensory techniques that were found to be applicable are Time-of-flight and Electromagnetic sensors. Both measure distance to obstructions in one dimension and by communicating with the robot a 3D position can be found. The difference between using cameras and sensors is that B&R only has an internal structure to supply the cameras to customers. Therefore, to include other sensors, new products have to be developed which is costly.

Time of flight A time-of-flight sensor is made up of a transmitter and receiver. The transmitter sends a signal in some direction and calculates the time it takes for the signal to bounce back and reach the receiver, this time is proportional to the distance traveled as the waves travel at constant speed. This will in turn provide us with absolute distances from the predicted position. The sensor could also work in sync with the robot to search for a position with minimal distance from the sensor. The signal sent is usually ultrasound waves, laser, or infrared rays. The principle is the same for all versions but the choice is mostly based on price and what disturbances can be found in the environment. The test works by placing a flat sheet of multiple emitters all sending signals perpendicular to this sheet. Once the signal bounces from the TCP the sensor will calculate the distance to it. The TCP then explores the area and tries to find a line in parallel with the signal thus revealing the height axis of the

sheet. Then it finds the signals of the other sensors and by finding coordinates for all sensors at equal distances the conveyor's width and depth axes are found. A single time-of-flight sensor is one-dimensional but by utilizing three sensors one can find 3 dimensions of data as long as the sheet is flat.

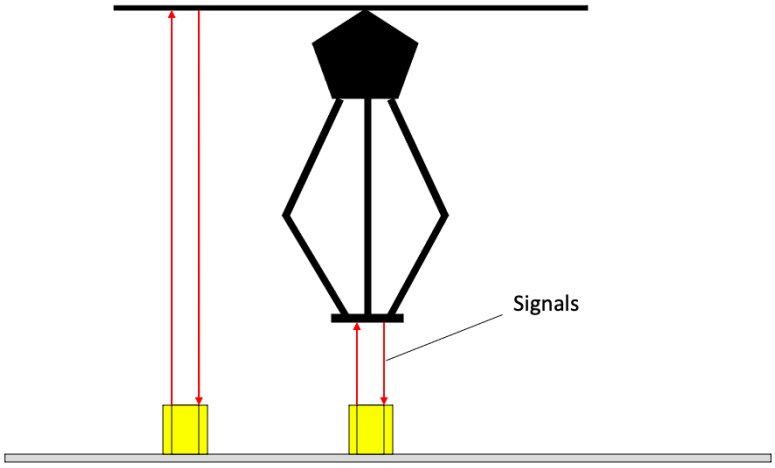


Figure 6.6: Seen from the side of the conveyor belt, the object sends signals and if the TCP is directly over the object, the coordinate will be saved from the TCP.

Inductive proximity sensor This technology is similar in many ways to the time-of-flight sensor in that it also measures distance to some obstruction. The difference is that the distance to the obstruction creates a difference in electrical potential. This is due to the electromagnetic field being obstructed so the physical obstruction acts as an electrical resistor. This is known as the Hall effect and is the basis for many inductive sensors. Inductive sensors notice obstructions in electron flow, which can be implemented by placing a piece of metal at the TCP and a sensory circuit on the conveyor. When the TCP is sufficiently close, the coordinate can be saved and compared to the predicted position that was calculated using the camera. Even though the TCP is in metal, which is detectable by inductive sensors, it is a good idea to attach an object to the TCP that makes it even easier to detect. This may be a protruding sheet of metal, for example, providing a larger surface for interrupting the inductive electron flow.

6.1.3 Concept for measuring points

For the error estimation, discussed in Chapter 4, a concept for an object that has points that can be measured, and which can be measured by both the camera and the robot, has to be generated. The simplest concept is using a sheet of paper with black squares, as seen in Figure 6.7. The paper is then taped onto the conveyor belt so that its position does not change while running the process. The camera can then find these black squares and through image processing techniques, estimate their positions. When in combination with the concepts presented in Subsection 6.1.1, the squares' future positions can be predicted. This concept also works well for the jogging and camera concepts in Subsection 6.1.2, since the robot can be observed while moving to the center of the squares and then measure the positions.

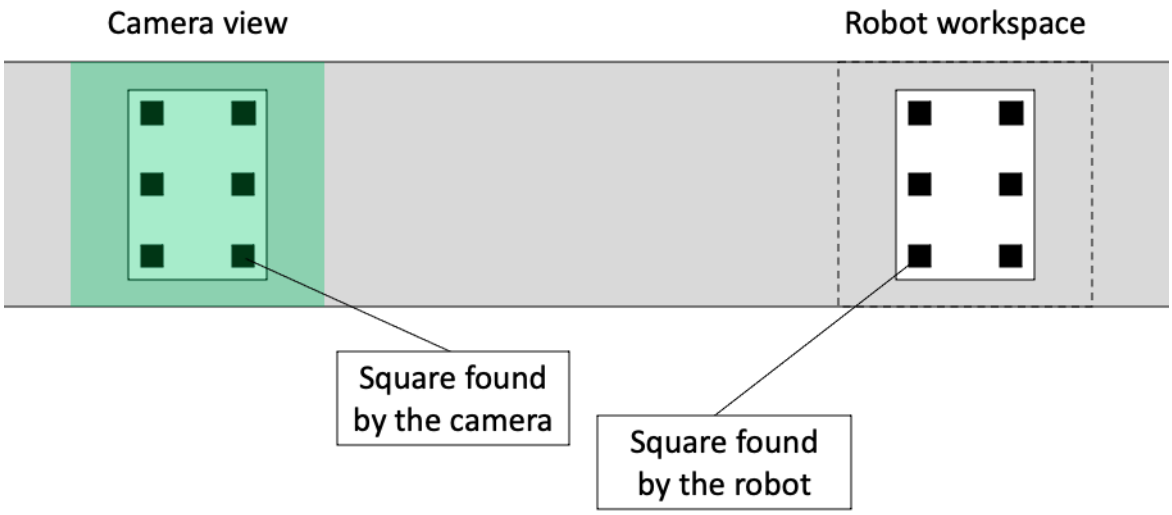


Figure 6.7: This figure displays how the sheet of paper will be placed on the conveyor belt and where its squares are situated.

6.2 Concept selection

In the limitations in Section 1.4, we introduced a few limitations that were put forward by B&R. For example, no expensive laser or cables. Another limitation of this project was the demand for proof of concept. Since we already knew that the jogging concept has been successful [1], and it does not require any new hardware, we chose to focus on implementing this concept and improving it. For the error estimation we therefore chose the concept *Jogging*. The chosen path estimation concept was *Following the object*, since this was believed to be the easiest to implement out of the two concepts for path estimation. Furthermore, we expected the results to be, roughly the same. From this point onwards, *Jogging* and *Following the object* will be used for the calibration process.

7 Calibration method

In Section 6.2, it was mentioned that the concepts *Following the object* and *Jogging* were chosen. In this chapter the implementation of these concepts, and how the final calibration method will work, will be explained.

7.1 Concepts and implementation

The robot, conveyor, and camera are controlled through a B&R PLC, which have four programs, called: Main, CameraControl, Jogging, and Conveyor. All programs are written in Structured Text and are all cyclical. The purpose of Main is to communicate with the rest of the programs, which controls the hardware. In this way, the project is ensured to execute the programs' processes in the correct order. The communications works in the following manner. Main and other programs share global variables, which are used to check whether programs should start or not. Global variables are usually not preferred when programming and there are other options, but they are the quickest to implement and were therefore used. When Main turns a global variable to true, another program is allowed to start. When the program is finished, it turns another global variable to true, to communicate back to main that it is finished. For example, Main turns on the global boolean *Start camera*, and in return, when the camera program is finished, it turns *Camera finished* to true.

When the PLC program is initiated, the camera starts first. Once the first picture is taken, which is the starting point for the paper, the next part of the sequence can start. This includes turning on both the camera's continuous image acquisition and the conveyor belt. Both run at the same time for images of the paper to be taken at different positions. The conveyor belt continues until the paper arrives at the robot's workspace. At that point, the jog function activates, and through a Human Machine Interface (HMI), the operator can jog the robot to the desired points on the paper. The execution sequences can be seen in Figure 7.1 and in Figure 7.2 the communication pathways are illustrated. While at a desired point, the operator saves the coordinates in the global coordinate frame. Finally, after all points are found, which are stored in a global variable of the type array, they are sent from the PLC to a computer where calculations are done using a program written in Python. The transfer is done by the PLC outputting a Comma Separate Variable (CSV) file. The file is then transferred from the PLC to the computer through a File Transfer Protocol server. When the CSV file has been transferred to the computer, one can extract the variables' values from the CSV file, and they can then be used for calculating the transformation matrix in the Python program. The Python program consists of methods for calculating the transformation matrix, and from that, it is possible to derive the subsequent translations and Euler angles. The point of the program is to simplify the process of loading the CSV file and deriving the necessary data for the

calibration. This is the final step of the process.

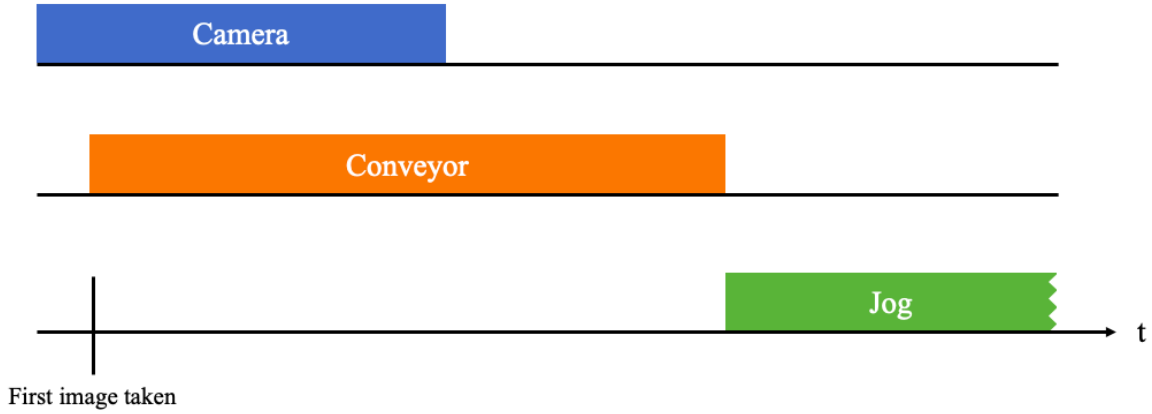


Figure 7.1: A timeline of the PLC programs' execution order.

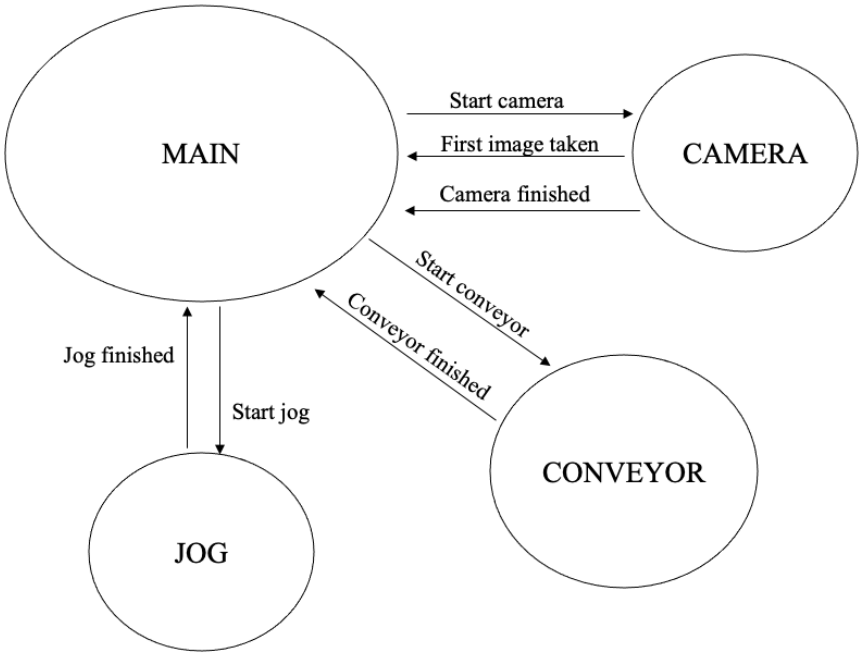


Figure 7.2: A schematic view of the PLC programs.

7.1.1 Camera program

This subsection will provide more information on how the camera program works. As mentioned in Section 7.1, the camera program will begin by taking one image. This is important as this is the starting point of the sheet of paper, that will later travel along the conveyor belt at a distance of 800 mm. When the camera has acquired the image it will perform the image processing and find the six points on the paper. If less than six points are found they will not be stored in the global variable that saves all points. This is so that inaccurate results are not validated later in the process. There are more precautions taken, such as using a Sobel filter on the camera images, a filter that is used for detecting edges. This filter works well in this case since there is a big

contrast between the white and black parts on the paper. Furthermore, settings such as the minimum and maximum area, and rectangularity, of the squares will be set in order to prevent the camera from finding points that are wrong. The camera view over the sheet of paper can be seen in Figure 7.3, where the six squares are marked. The Sobel filter has been applied to this image, and therefore, everything is black except for the edges.

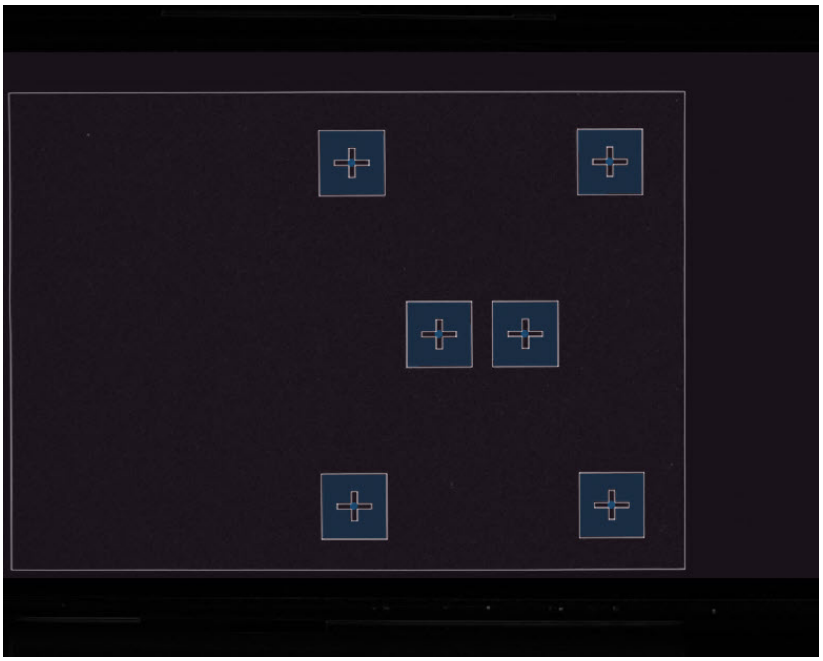


Figure 7.3: The sheet of paper with six points. The camera found the middle points of all six points and marked them with blue dots in the crosses.

The black squares are the target for the camera function, while the crosses are made to assist the jogging. The squares are 4 cm on each side, while the crosses have stripes of 20 mm in length and 2 mm wide.

7.1.2 Convenyor and jogging programs

As mentioned in Section 7.1, after the camera has taken its first image the conveyor belt will start running. It will keep moving until it has arrived at a set distance. At this point, the jog program will be activated and the operator can start jogging the robot and saving the coordinates. The operator will be able to choose the velocity and the distance for each move. The velocity ranges from 1 to 200 mm/s and the distance can either be chosen at 20 or 1 mm. The larger distance is to move between points, while the smaller is to move to the center of a square.

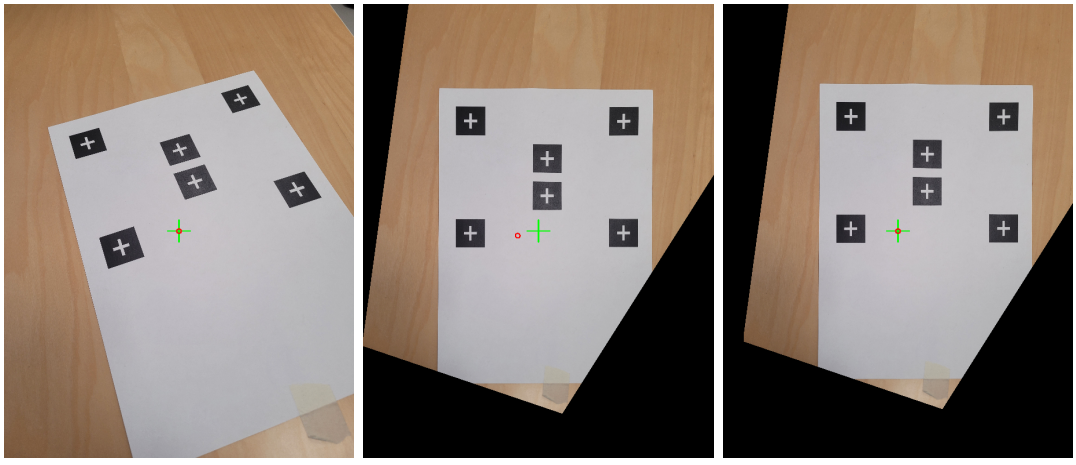
7.1.3 Mathematical program

This subsection will go through the parts of the Python program, which handles the all calculations. There are most notably five important parts of the program.

- Projective transformation.
- Converting pixel coordinates into global coordinates.
- Deriving the homogeneous transformation matrix.
- Deriving the Euler angles and the translations.
- Plotting all points in a 3D graph.

Implementing the projective transformation

A projective transformation is usually used as an image transformation, but due to the inability to extract images during the process, this transformation is only performed on the data. One can imagine this transformation as moving coordinates instead of stretching an image. This projective transformation consists of two separate transformations. The first part moves the data from four corner points into their expected locations. The expected location is a non-rotated rectangle where the sides have the same ratio as the real points from the sheet. With four known points and four destination points a projective transformation matrix is calculated and expressed as the matrix M_1. All the data from the camera is then multiplied by this matrix and creates a new set of data that closer represents the sheet's real size and shape. The next transformation will structure the data for global coordinates by shifting the transformed points so that the previous center point becomes the center point again. This means that the image has the same center point both before and after the transformations and acts as though the camera is looking at the same point but from a different perspective. This second step is done through only translation. The first step is important to place the data on a theoretically perfect XY plane and the second step shifts that plane to place data correctly in front of the camera which can be seen in Figure 7.4.



(a) An example image of the sheet.

(b) The projected example image.

(c) The projected example image with translations.

Figure 7.4: The complete projective transformation image filtering process from an example image (7.4a) into a projected (7.4b) and then a translated image (7.4c). The green cross represents the middle of each image. The red circle indicates a point on the paper that was in the middle in Figure 7.4a

Converting pixel coordinates into global coordinates

The pixel coordinates of the points will vary between 1280 px in the x direction and 1024 px in the y direction as decided by the camera hardware. One should also note that the x-axis is flipped in the camera frame, compared to the global frame. This can be seen in Figure 7.5. Therefore, it is necessary to first flip the x-axis by negating all pixel coordinates' x values. After this procedure, the middle point of the camera view should be the origin, instead of the top left corner. This is done by subtracting half of the pixel length of the image in both the x and y directions. Furthermore, the pixel coordinates will be converted into millimeters. This is done by multiplying each pixel coordinate by a conversion number. The conversion number is derived by dividing the distance between the points in millimeters by the distance in pixels. This results in

$$\mu = \frac{\text{Distance in millimeters}}{\text{Distance in pixel}}. \tag{7.1}$$

The distance in pixels can be derived by taking the distance between the points found in the image. The real points in millimeters on the sheet of paper can be derived by physically measuring them. By applying μ to each camera point, they are converted into millimeters. Finally, the camera is situated somewhere in 3D space. According to the available drawings it is supposed to be placed in (x, y, z) = (543, 101, 738), in millimeters. This is added to all converted points found by the camera. After applying these steps, all camera points should be correctly converted into global points. The correct results should look like Figure 7.6.

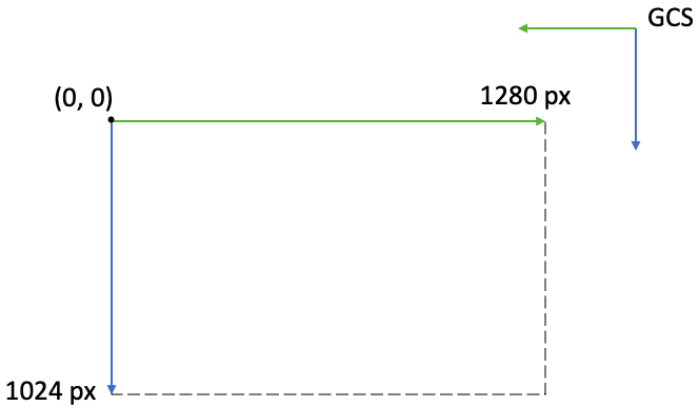


Figure 7.5: Shows the global coordinate system (top right) and the camera's coordinate system (left). They have opposite directions of the x-axis (green).

Finding the homogeneous transformation matrix

Deriving the homogeneous transformation matrix will be done using the same workflow as mentioned in Chapter 4. In Python, the library Numpy has a function called *pinv()* that is used to derive the pseudo inverse. An important detail is that, in order to get the transformation matrix, one will have to compare the jogged and predicted points where the predicted points are situated around the origin. Since the predicted and

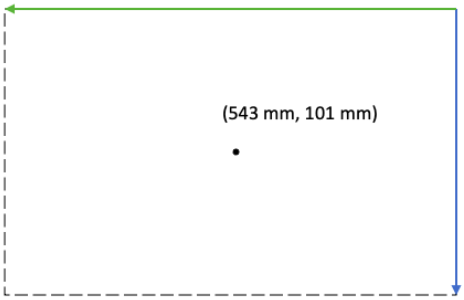


Figure 7.6: Shows the final results, after flipping the x-axis (green), changing the origin, and converting pixel points into global points.

jogged points will be somewhere in the global space they have to be moved to the origin. For this to be possible one has to subtract the centroid of the predicted points to all jogged and predicted points. At this point, the pseudo inverse can be found and the transformation matrix can be derived. After deriving the matrix the rotations and the translations can be extrapolated from it. The translations can be extrapolated from vector **t** in Equation 4.1 while the rotations are extracted from the matrix **R** from the same equation using the Python library Scipy. This library has a function that can remove the scale from the matrix and then express it as pure rotational Euler angles in degrees with the XYZ rotation order [12].

7.2 Data collection

In this section, the data collection from the camera and the robot will be explained in more detail. For each image acquired, the camera will find the black squares if they fit the criteria, as explained in Subsection 7.1.1, and derive their middle points. This will give the points in the camera's coordinate frame as pixels. Furthermore, the coordinates will be saved into the global array that keeps important data that is to be transferred to the Python program. However, the points are in the pixel coordinates and have to be translated into coordinates that are of equal type as the global coordinates.

Within this Python code, the coordinates must also be transformed according to Figure 5.1. The camera input will represent P after being translated from pixels to real positions. The new position P' will be calculated as a translation of P along the path found using the methods in Chapter 3. The distance of this translation is described in the robot code and is received through the same CSV file.

Jogging will be used for gathering the coordinates of the points on the paper in the robot workspace. These points represent P'_T. It is the coordinates at the tip of the TCP that are of interest, since it will touch the center of the squares on the paper to mark the points. For jogging to be carried out successfully and in a secure manner, different velocities will be used. As the TCP nears the desired point, one wants to slow down and jog more carefully, which is done by choosing a smaller step size. The TCP can therefore move 20 mm or 1 mm at a time. After saving all jogged points, they are saved to the global variable that was previously mentioned. Several attempts

will be performed with both the camera and the jogging, in order to investigate noise and accuracy.

7.3 Physical testing and validation

As mentioned in Chapter 2, a physical setup of the system is available, and it will be used for testing the calibration process and gathering data to validate the process.

From the data collection, one can derive several parameters to validate. Of most interest, is the performance of the calibration method, which can be validated based on its deviations. The simplest form of results is how much the result deviates from test to test with regard to averages and standard deviations, especially in terms of translations and Euler angles. Using the average and the standard deviation of these values, one can get an idea of the performance. A high standard deviation will imply that the method is unreliable. What constitutes a high deviation depends on the application of the pick-and-place process, since some applications are more forgiving than others. However, if the method deviates as much as centimeters, that would be problematic. Other important parameters that will be validated are the coordinates of the prediction and the direction of the conveyor belt.

7.3.1 Validating camera accuracy

The camera must find the points accurately to get accurate results. A way of testing this is to sample many images and their respective coordinates. By validating the accuracy of the results, i.e., how much it deviates from sample to sample, one can get an idea of the variance of the accuracy. For this test, the sheet of paper has, instead of six points, only one point. This can be seen in Figure 7.7, where the camera has found that point and its coordinates.

7.3.2 Operator and calibration accuracy

Apart from the camera having to be accurate, the operator also has to be accurate. One problematic part of the jogging process is that the view over the robot's workspace influences the results. The perspective can greatly influence the placement of the TCP. Nevertheless, a test that can be performed is performing the calibration method several times, having the sheet of paper start from the same starting point and stop at the same endpoint. This minimizes errors from the camera since the same images should be acquired for each sample. The idea is that this will lead to most of the errors occurring from jogging inaccurately.

To determine the accuracy of the process, however, one will have to move the paper's starting point for each sample. This test will be performed with and without using the projective transformation so that the difference can be analyzed. Another test that will be performed, is introducing errors of a known distance and investigating how well

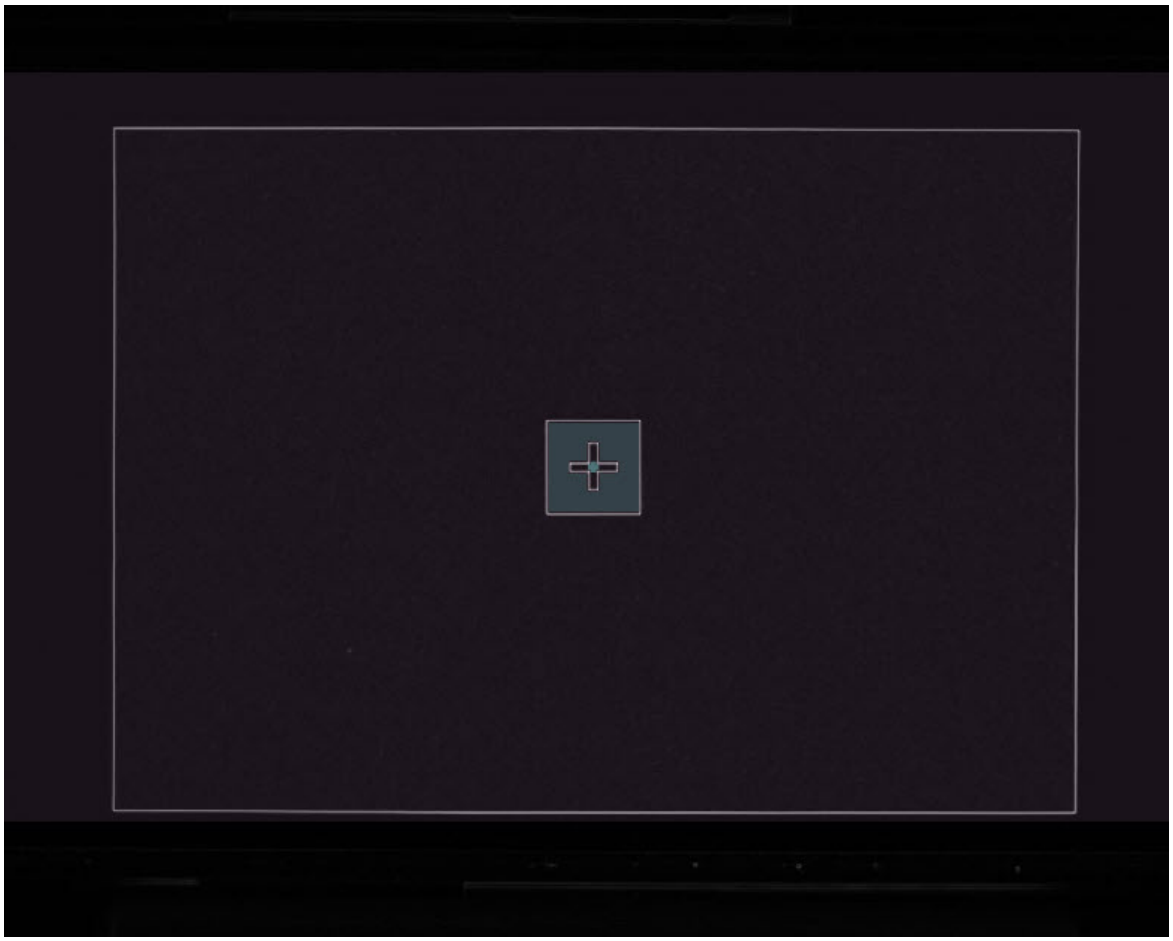


Figure 7.7: An image of the sheet of paper and its middle point, found by the camera. The middle point is marked by a small blue dot in the cross.

the process finds these errors. This will be done by introducing a 50 mm error in the robot configuration, in both x and y directions.

7.3.3 Pointer

To jog the robot safely to the conveyor, safety precautions are needed. The risk is that the TCP will go too far down and into the conveyor belt, destroying both machines. To remedy this, a pointer will be attached to the TCP, which is designed to break at lower forces than the conveyor or the robot. The pointer can be seen in Figure 7.8.

The pointer is a 110 mm long cylinder with a pointed tip for accurate pointing and a base that attaches to the TCP. The material is a standard PLA 3D-print plastic with low elasticity, meaning that the pointer must be thin to break easily. To decrease the toughness of the pointer, it is equipped with holes along its sides to create fragile supports that will bend and break under pressure. The base of the pointer is fitted with four screw holes that match the screw holes on the TCP and are fastened according to Figure 7.9.

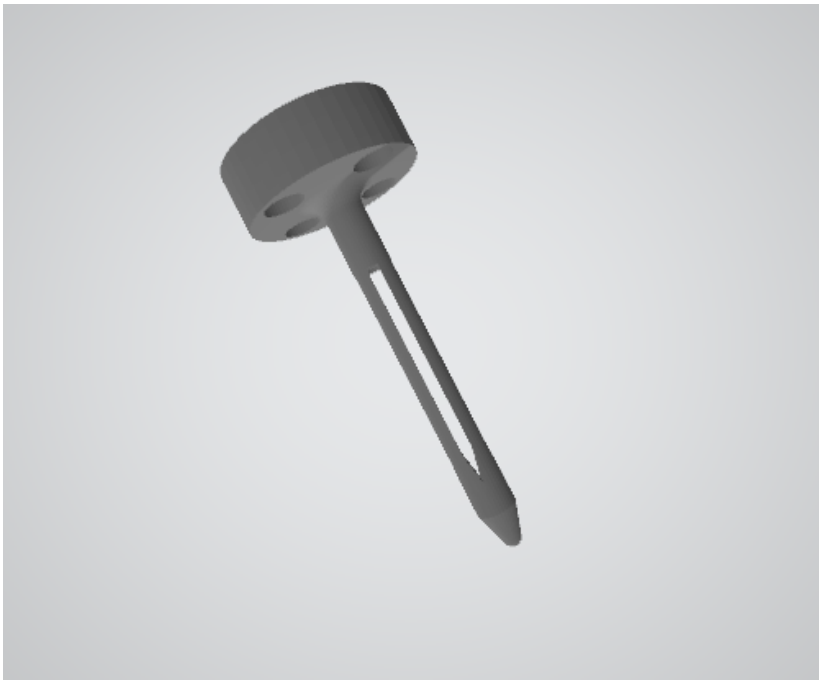


Figure 7.8: 3D version of the TCP pointer.

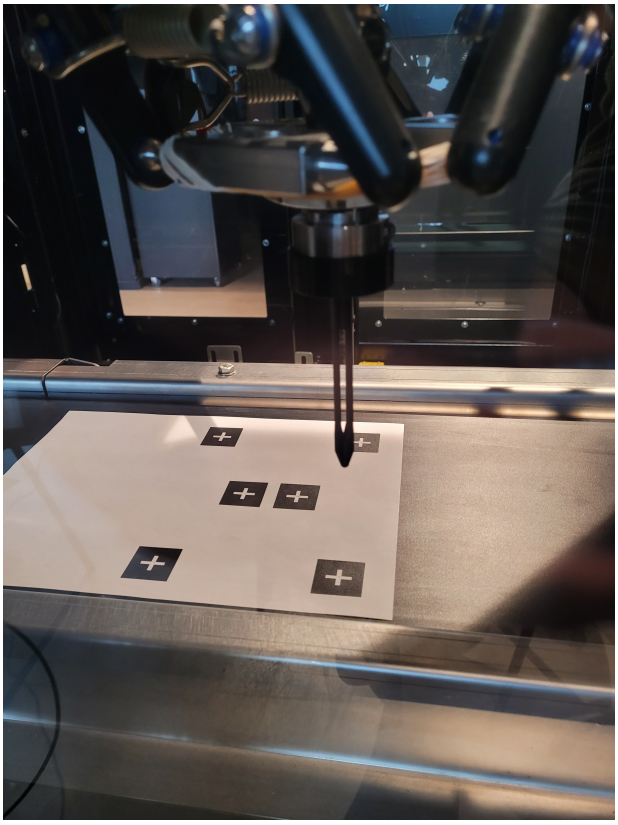


Figure 7.9: Photo of the pointer fastened on the TCP of the pick and place machine. The sheet of paper is underneath the conveyor.

8 Results

8.1 Accuracy of the process

By changing the starting position of the sheet of paper, at each sample, for 30 samples, where each sample took 10-15 minutes to perform, it resulted in Table 8.1. Notice how the mean value resulted in a value $\neq 0$, for all parameters. These are the errors found through the calibration process. Each parameter had its respective σ, which is the standard deviation normalized by $N - 1$, where N is the number of samples.

Parameter	mean value	σ
X (mm)	-4.58	0.72
Y (mm)	18.13	1.10
Z (mm)	-0.41	0.42
α (°)	-0.82	0.13
β (°)	0.01	0.14
γ (°)	0.30	0.12

Table 8.1: The results from 30 samples. Errors in translations (X, Y, Z) and Euler angles (α, β, γ), and their respective standard deviation. The error is between the predicted and the jogged points.

The results in Table 8.1 are derived from the difference between the predicted and the jogged plane. A visual representation of these two planes can be seen in Figure 8.1. The blue plane is the predicted plane and the red is the jogged plane. The jogged plane is tilted around the x-axis, this is also seen in Table 8.1, where $\alpha < 0$.

In Figure 8.2, one can see all the points that were found during the calibration process of the same sample as in Figure 8.1. The blue plane, which is larger than the other planes in Figure 8.2, is made up of all the camera points. From these points, the prediction is made, which results in the smaller blue plane. The smaller blue plane and the red plane are the same planes as in Figure 8.1. The images' dissimilarities result from the differences in the scaling of the axes.

By applying the transformation matrix that was found, the predicted plane is fitted on the jogged plane, which can be seen in Figure 8.3 a). In b), the predicted plane is changed through the translations and Euler angles found for that specific sample.

8.2 Camera precision

In Figure 8.4 a), all camera points are plotted. In b) each centroid is plotted and in c) they are plotted with a fitted black line. Notice the downward trend on the black

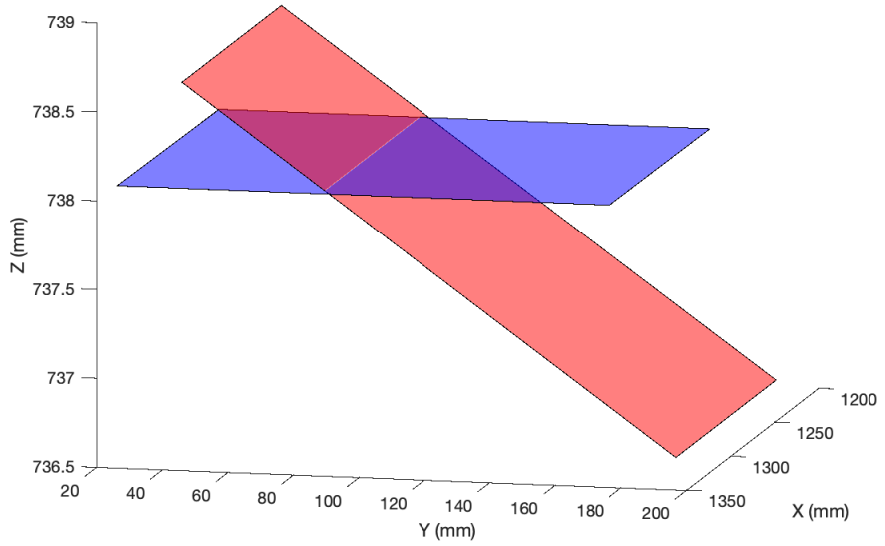


Figure 8.1: A plot of the predicted plane (blue) and the jogged plane (red). This is the result from one of the 30 gathered samples.

line, this is the path in which the paper travels. The angle of the trend is the camera angle, θ, the angle between the camera and the conveyor belt.

From the former 30 samples, θ had the results as shown in Table 8.2.

	mean value	σ	Min	Max
θ (°)	-0.19	0.02	-0.23	-0.14

Table 8.2: Results of the camera angle, i.e., its mean value, standard deviation, min and max values. These results are derived from the same samples as from the results in Table 8.1.

There were some concerns that the variance of θ would result in an unsatisfactory change in the X and Y translations in Table 8.1. Therefore, the correlation between θ and the translations in X and Y were derived. The correlation can be seen in Table 8.3, and the plotted values can be seen in Figure 8.5

$corr(X, \theta)$	$corr(Y, \theta)$
0.01	-0.08

Table 8.3: Correlation between the translations in X and Y in Table 8.1, and θ in Table 8.3.

By performing the camera test mentioned in Subsection 7.3.1, and collecting the data, the following results were derived. With a total of 98 images, their averages and standard deviations in x_c and y_c directions can be found in Table 8.4. The results are also shown in Figure 8.6.

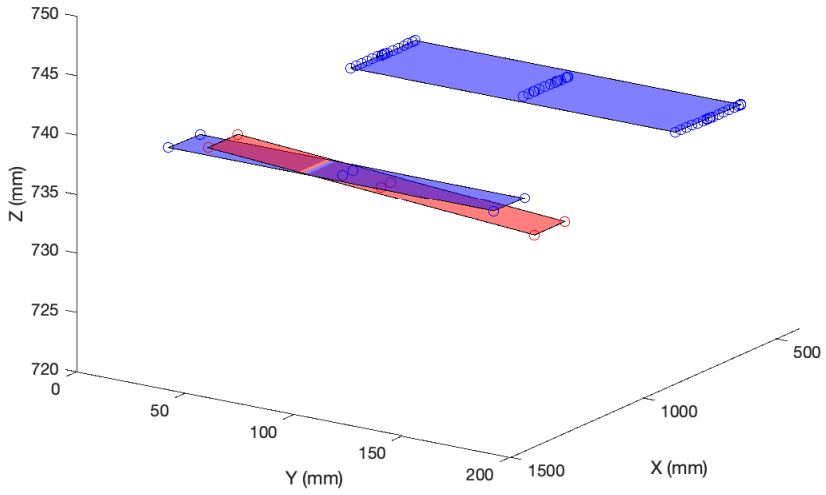


Figure 8.2: The full plot of all points gathered during the calibration process of the same sample as in Figure 8.1. The points on the larger blue plane are points gathered from the camera. The other red and blue planes are the same as in Figure 8.1.

$\bar{x}_c$	$\bar{y}_c$	σ_x	σ_y
630.47	454.97	0.0154	0.0144

Table 8.4: Accuracy of camera in pixels. $\bar{x}_c$ and $\bar{y}_c$ are the averages of the point in pixels in the camera's coordinate system, while σ_x and σ_y are the standard deviations in pixels.

8.3 Operator accuracy

Performing the calibration method, using the same position for the starting point of the paper and for the endpoint, and repeating this, resulted in the following values, see Table 8.5. This test was repeated 7 times. Thereafter, the average, interquartile range (IQR), and the range were derived.

8.3.1 Introducing errors

By introducing errors of 50 mm in the positive x and negative y directions, and then running the calibration process, the following deviations after 3 samples were found. One sample was with an error in x, one in y, and one in both x and y. The 3 samples are compared to the mean value presented in Table 8.1. The deviations, in X and Y, can be seen in Table 8.6. The difference between the mean values and the three samples can be seen in Table 8.7.

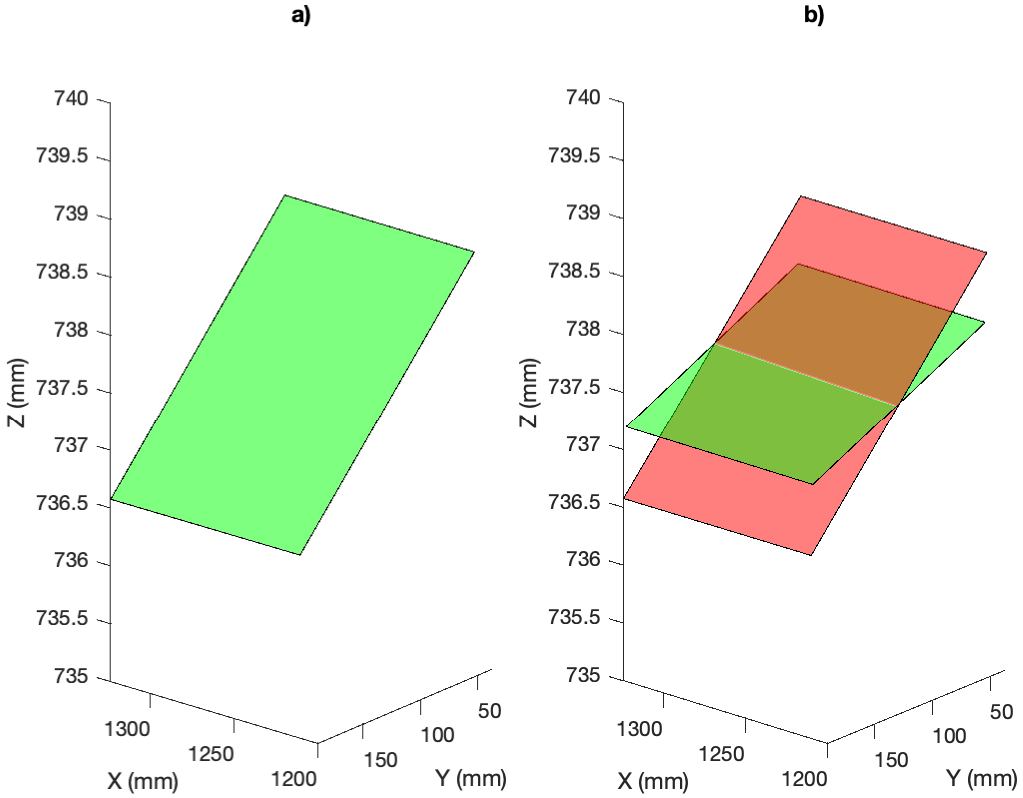


Figure 8.3: Difference between applying the transformation matrix to the predicted plane, compared to applying the translations and Euler angles. In a), the predicted plane (green) is transformed using the transformation matrix. However, in b), the predicted plane (green) is transformed using the translation and Euler angles. The red plane is the jogged plane. Notice how using the transformation matrix fits the predicted plane onto the jogged plane almost perfectly, the green plane in a) and the red plane in b). When using the translations and Euler angles, it results in the green plane in b), which differs more from the jogged plane (red).

The predicted and jogged planes in sample 3, in Table 8.6, are plotted in Figure 8.7. Although similar, these planes differ from the ones in Figure 8.1, because of the introduced 50 mm error.

8.3.2 Projective transform

The results from implementing the projective transform utilize the same input and output from Table 8.1 and Table 8.2 where the camera input is first transformed with a projective transformation.

The following tables are a comparison of Table 8.1 and Table 8.8 as well as a comparison of Table 8.2 and Table 8.9.

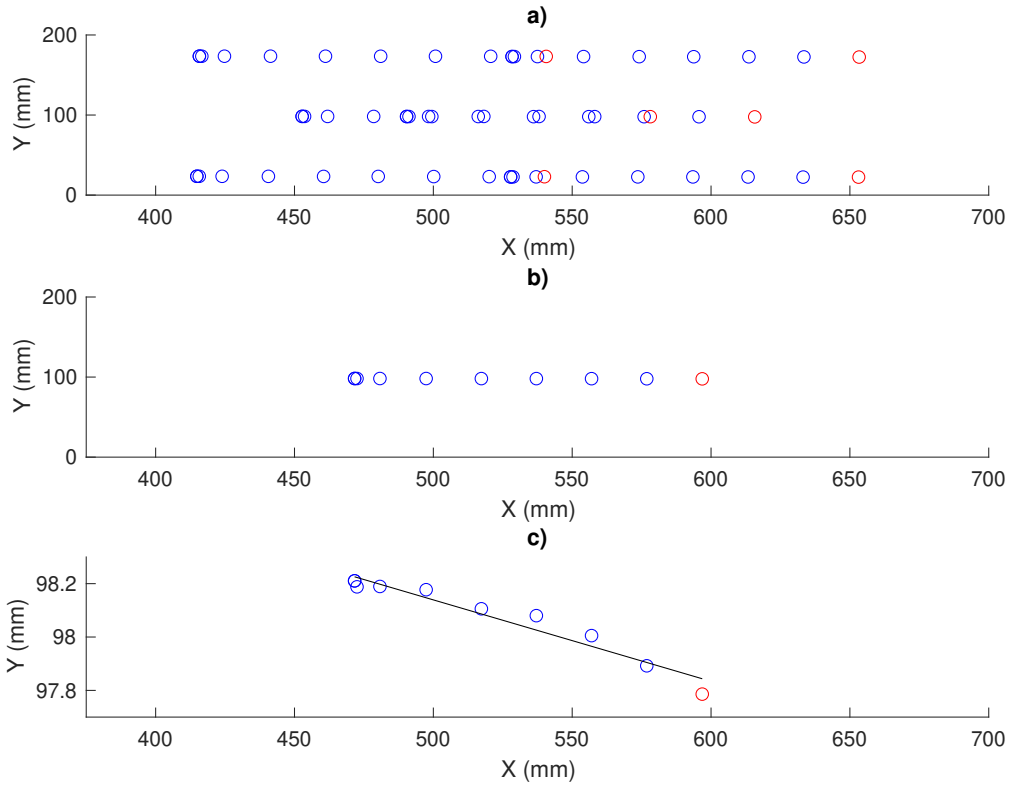


Figure 8.4: During the calibration process, ten images were taken. In a), all points that were found by the camera are plotted, a total of 60. There are six red circles which represent the 6 points found in the last image. In b), the centroid for the 6 points in each image is plotted, and the last centroid is marked in red. In c), the centroids are the same as in b), but with a trend line through the centroids. This line is the same as the path of the paper.

Parameter	mean value	IQR	Range
X (mm)	−4.55	0.16	0.43
Y (mm)	20.07	0.37	1.65
Z (mm)	0.11	0.12	0.34
α (°)	−0.77	0.07	0.38
β (°)	0.34	0.37	0.39
γ (°)	0.31	0.14	0.66

Table 8.5: Results from 7 samples, by performing the test that is presented in Section 7.3.2. The results are presented as the average, interquartile range (IQR), and range in order to accurately represent the results.

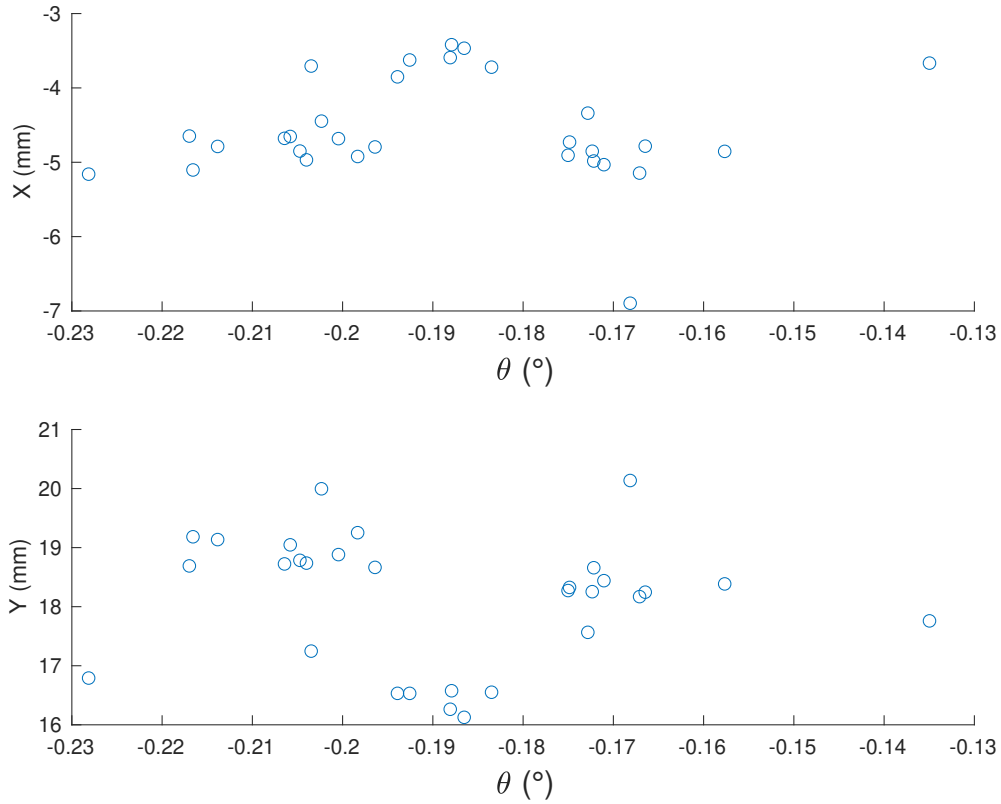


Figure 8.5: Scatter plot of the 30 samples with their relations between the error in translations X and Y, and θ. The errors in translations are on the vertical axes and θ is on the horizontal axis.

Parameter	mean value	Sample 1	Sample 2	Sample 3
X (mm)	−4.58	47.26	−2.82	47.19
Y (mm)	18.13	16.72	−33.75	−33.67

Table 8.6: Mean values in X and Y from Table 8.1, and also, errors in X and Y by applying a 50 mm error in the robot configuration and then running the calibration method. This error was applied in both positive x and negative y directions.

Axis	Sample 1	Sample 2	Sample 3
Δx (mm)	51.84		51.76
Δy (mm)		51.88	51.80

Table 8.7: Difference between the mean values and in the three samples in Table 8.6.

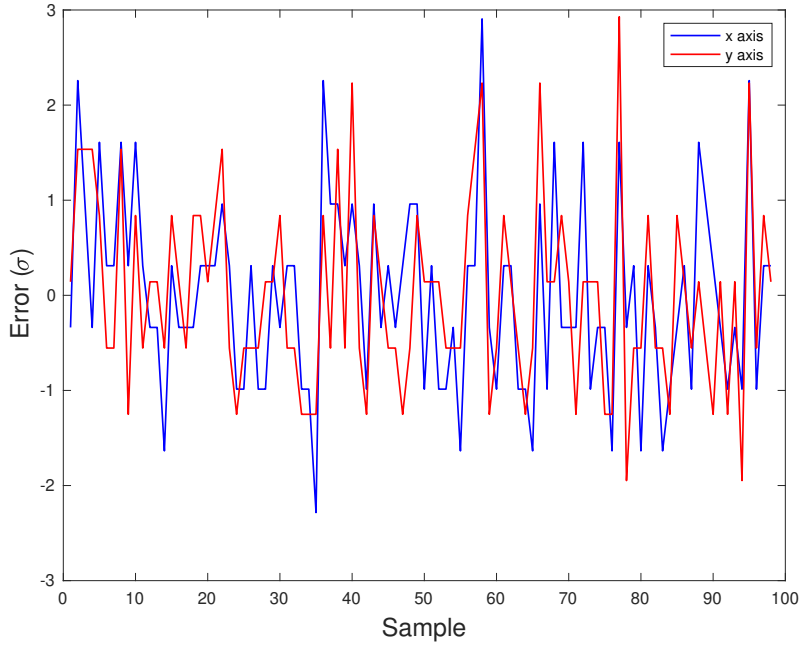


Figure 8.6: Deviation from the normalized values of the pixel positions in the x and y-axes. The vertical axis of the figure represents the number of standard deviations that each sample deviates from the norm. The deviations in the x and y-axes can be found in Table 8.4.

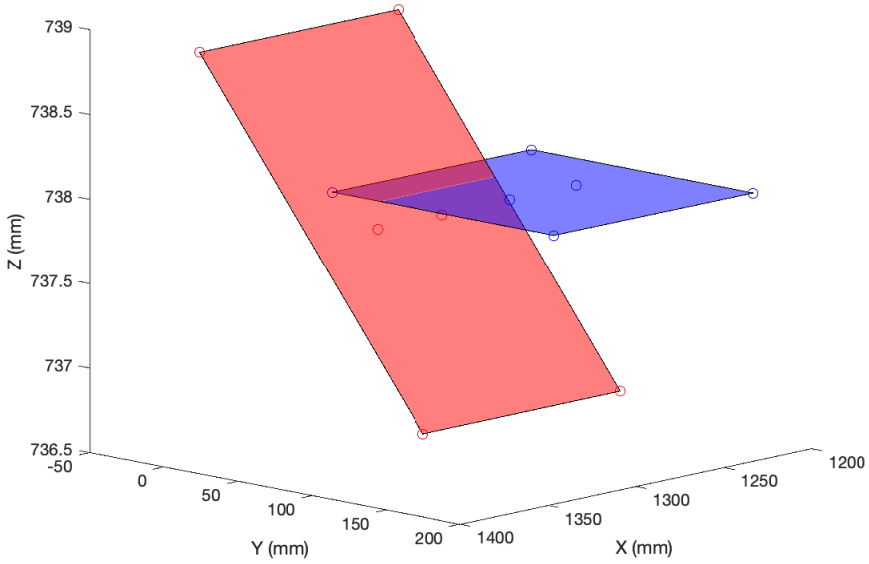


Figure 8.7: Display of sample 3 in Table 8.7, where there is an error of 50 mm in both x and y-directions. Both the predicted plane (red) and jogged plane (red) can be seen. Furthermore, this figure can be compared to Figure 8.1, where there are no errors introduced.

Parameter	mean value	σ
X (mm)	−4.50	0.57
Y (mm)	16.20	6.38
Z (mm)	−0.46	0.42
α (°)	−0.83	0.14
β (°)	−0.01	0.11
γ (°)	0.18	0.44

Table 8.8: The result from 30 samples with projective transformation. Errors in translations (X, Y, Z) and Euler angles (α, β, γ), and their respective standard deviation. The error is between the predicted and the jogged points when including projective transformation.

	mean value	σ	Min	Max
θ (°)	-0.04	0.46	-1.30	0.93

Table 8.9: Results of the camera angle when implementing projective transform, i.e., its mean value, standard deviation, and min, and max values. These results are derived from the same samples as for the results in Table 8.8.

Parameter	mean value	σ
ΔX (mm)	0.02	−0.01
ΔY (mm)	−1.82	5.37
ΔZ (mm)	0.00	0.00
$\Delta\alpha$ (°)	0.00	0.00
$\Delta\beta$ (°)	0.00	0.00
$\Delta\gamma$ (°)	−0.12	0.31

Table 8.10: The difference in result with and without projective transformation. Differences in translations (X, Y, Z) and Euler angles (α, β, γ), and their respective standard deviation.

	mean value	σ	Min	Max
$\Delta\ \theta$ (°)	0.14	0.44	-1.07	1.06

Table 8.11: Difference in results of the camera angle with and without implementing projective transform, i.e., the difference in mean value, standard deviation, min, and max values.

9 Discussion

9.1 Transformation matrix

A transformation matrix is useful due to its ability to concatenate a lot of information into a single matrix. That information is, rotation, translation, and scale. This allows us to extrapolate lots of information from minimal amounts of points. The problem arises when the transformation matrix portrays information that should not be there, in our case, scale should not be present as the same paper is measured. Small scales will lead to larger errors over large distances. The effect of scale is larger the further away from the origin we are, making the process more accurate closer to the chosen origin. The scale is a result of small measurement errors that should not exist, as both the jogging and the camera are measuring the same paper. The scale is not included in our software calibration as only translation and the Euler angles are extrapolated. The difference when including scale can be seen in Figure 8.3.

To derive the transformation matrix the pseudo-inverse was utilized to calculate an over-determined system. Using this implies that no perfect solution exists, as it gives an approximate solution for contradicting equations. The pseudo-inverse solves two important issues. Firstly, it takes equal account of all points improving accuracy, and in doing so, lessens the effect of measurement errors. It is difficult to find the true errors, but if they are assumed to be random, and centered around the true value, then by using many points, the pseudo-inverse finds errors that are close to the true values. This is supported by the low σ values of Table 8.1. It would theoretically be better to include more points on the paper, as this would increase accuracy, however, this would also increase the execution time taken for each test. In a factory setting, it would be better to use more than six points, like in our case, for a process that is only performed once. During this project however, it was more important to do more tests. The pseudo-inverse takes into equal account all points, and this has great importance for accuracy in a broader sense than measurement errors. By approximating all points, we get closer to the true values, where lots of points describe the same thing: rotation, translation, and scale. By using more points we get more information where everything should point towards the same values. This is of course directly impacted by measurement errors, but as mentioned, the pseudo-inverse handles the multitude of information and takes the values that are approximately best for all equations. This handles outliers and provides statistically likely values for all transformations.

Another improvement that could have been made, is to widen the test area. One of the risks of this test is that all values are correct in our sheet, but perhaps incorrect outside of it. The best examples are rotations and scales, that, with small errors, grow larger further away from the pivot point. Our sheet was as large as possible, while still being within the camera view for the entirety of the camera process, i.e., until ten images have been acquired. We are in this aspect hindered by two things, the width of the conveyor belt, and the camera view. In general, one might be hindered by the

workspace of the robot as well, but not in this instance.

9.2 Relative errors and transformations

There is no way to measure the positions of the mechatronic devices in the global coordinate space directly, but through their sensors, one can derive their relative positions in the global coordinate space. Due to this, we may only find relative errors between the camera and the robot. This in practice means that we only alter the configuration for one of them so that it will better fit in with the other one. This is not completely accurate when comparing it to the true scenario, where most likely both of them carry some imperfection with regard to their theoretical position and orientation. However, as long as the two parts are speaking the same language while communicating, it will not cause issues as they are both in agreement on how they relate to each other.

9.3 Accuracy

By reviewing the results, we see that the calibration method resulted in a deviation from the drawings. First of all, in the x and y directions, there was a difference of 4.5 and 18 mm, respectively, in Table 8.1. If our calibration is correct, this would entail that either the camera, robot, or conveyor belt, or all of them, are incorrectly placed compared to the drawings. However, as we will discuss later, there are several sources of errors that can influence the results. Nevertheless, we also found a rotation around the x-axis, since $\alpha < 0$. Practically speaking, this means that one side of the width of the conveyor belt is above the other. This aligns well with what we experienced as we jogged the robot since we had to jog down more on one side than the other. The width of the conveyor belt is 240 mm, and an $\alpha = -0.8^\circ$ over this length would result in a -3.35 mm difference in height. When jogging the difference was 2 mm. However, these measurements were not taken at the edges nor 240 mm distance apart, but 150 mm. The height difference should therefore be lower, and instead $\Delta z = 150 \cdot \sin(-0.8) = -2.09$ mm, and an α of -0.8° is therefore reasonable.

In Table 8.1, we found the total deviations of the system. This test was performed a total of 30 times, to get reliable results. A high σ would be expected of an unreliable calibration process. However, our results were rather stable, i.e., low standard deviations in all translations, of at most 1.11 mm in Y. A standard deviation of 1.11 mm entails that, 95% of the time, one will achieve an accuracy of ± 2.2 mm. This precision will for some industrial applications not be precise enough. However, for rougher industrial applications, such as the packaging industry or warehouse automation, the tolerances are within millimeters and can therefore be considered a good fit.

In Figure 8.3 a), we can see that the transformation matrix that was found, accurately fitted the predicted plane onto the jogged plane, as the green plane in a) is almost identical to the red plane in b). However, when deriving the translations and Euler angles from the transformation, some of the scale is lost. This is why the transformed

plane (green) in b) is not perfectly fitted onto the jogged plane (red). However, at the edges, there is only an error of roughly 0.5 mm in Z. On the entire length Y, 150 mm, that is not much. Although it looks like a big difference in the plot, due to the scaling of the z-axis. To move forward we disregard scale and derive the Euler angles to input these into the robot configuration in Automation Studio. The scale is known but we also know that is shouldn't exist and so we do not implement it in the configuration.

With our results in accuracy, we can make a distinction between whether or not the results are accurate, and through this, if the accuracy is sufficient enough for a pick-and-place process. As mentioned, we are likely to fall within 2.2 mm from the average, our rotations are reasonable and the results are better than the predicted plane. This is true when comparing Figure 8.1 and Figure 8.3, as in the latter figure, the transformed plane is closer to the jogged plane than the predicted plane is in the former figure. However, the error in scale and variance in our results add uncertainties, but the results are an improvement. Scale adds more issues outside of our sheet. Therefore, the result is more accurate in the middle of where the sheet of paper was measured. There are suggestions for improving this in Subsection 10.2, but for many processes, this will be sufficient, at least within the sheet of paper's area.

9.3.1 Camera program

The accuracy of the camera was high, which can be seen in Table 8.4. However, one should note that this test was done when the conveyor belt was stationary, and not while it was moving, which influences the results. The camera program is essential to the calibration since it defines the starting point of the paper, and at what angle the paper travels. The camera angle θ, had an average of -0.19° and did not deviate much during each sample, which can be seen in Table 8.2. Since this angle influences the prediction of where the paper will end up, it is important that it is correct. The maximum deviation in θ, from its mean value, was -0.05°. The paper travels 800 mm, this means that the change in X and Y is $\Delta X = 800 - 800 \cdot \cos -0.05$ and $\Delta Y = 800 \cdot \sin -0.05$, which is ≈ 0 mm and -0.70 mm. An incorrect θ therefore has a larger effect on the accuracy in Y than in X. By deriving the correlation between the angle and the translations, in X and Y, we can state the effect that our θ had on our results. In Table 8.3, we found that the correlation between θ and the translations in X and Y were low, 0.01 and -0.08, respectively. This result indicates that θ has a very low correlation with the translations, and therefore can be seen as having a random effect on the errors. Furthermore, this is what we would expect to happen. If there is an error, then there should not be a correlation between θ and the translations, since that would entail that there is something systemically wrong in the calibration process. We can also see in Figure 8.5, that there is no visible trend in the data, which supports the low correlations found in Table 8.3.

Projective Transformation

The projective transformation had unsatisfactory effects on the resulting transformations as the deviations increased. In Table 8.10 and Table 8.11 the impact from

the projective transformation can be seen as an increase in standard deviations for Y and X translation and γ rotation. This indicates that the accuracy is lower when using projective transform. The projective transform had a variety of effects, where some values stayed relatively the same, while other values varied greatly. The largest deviation was in Y, where the standard deviation was increased when compared to Table 8.1. An increase of 5 mm in the standard deviation is a sign of poor performance. The average in Y was also altered by 2 mm. Some values had no impact from the projective transformation, both the Z translation, α angle, and β angle had no impact. This is because those values are influenced by changes in the measurements of Z. The projective transformation only had effects on X and Y translations, since the camera took 2D images, and thus had no effect on Z. The final angle γ had a relatively large increase compared to the other angles, and the same goes for the camera angle θ, where the mean differentiated -0.12° for γ and 0.14° for θ. The standard deviation increased for both angles: 0.31° and 0.44°, respectively. To summarize, the projective transformation implemented affected the Y translation, γ and θ, the most. There is almost no difference in values that impact height, such as Z translation and the angles α and β. The final value of X translations is impacted less than a millimeter.

One thing about the results that sticks out, is that the values γ and θ are impacted by the same magnitude. This is likely due to both rotating around the z-axis, and being roughly the same size and opposite, implying that γ compensates for θ. θ is also increased, and this is probably due to that the projective transformations prioritize fixing the sheet of paper to be perfectly aligned with the X and Y axes. This will cause rotations in the camera angle θ depending on how the sheet is oriented. The system is robust enough to compensate for this with our γ angle, but the angle of the paper introduces a lot more uncertainty to our θ calculations, which introduces higher standard deviations.

With an increase in θ, the paper would be sent in another direction. θ is very close to 0, but the distance the paper travels is quite large. Thus, this will have, relative to the found errors, a large impact on the errors, especially in Y. This can be seen in the results, as the Y translation's standard deviations are increased. The average of Y is also altered more than X, which we would expect from this behavior. Something unexpected is the impact of projective transformations on the X translation, where the standard deviation is slightly lower. However, the difference is small, which indicates that the impact of projective transformation on X translations, is of less importance.

All the values impacting height values remain relatively the same in both cases. This is expected as the camera has no height data. If there had been any differences in height, it would indicate that the projective transformation does not function properly. In our case, the camera was already manually tuned to some extent and the largest difference was in θ.

9.3.2 Operator

In Table 8.5, the operator accuracy can be seen. It is called operator accuracy since this test is meant to alleviate errors that are not influenced by the operator. By letting the sheet of paper start at the same starting position at each sample, the camera should,

roughly, take the same images and find the same points, and therefore reduce the impact of the camera errors on the results. The results show that there are deviations in X and Y, but very little in Z. This means that, compared to the drawings, the the physical setup is offset by those translations. However, the interquartile range and range were rather large in Y. This is believed to be because of the poor visual perspective along the y-axis, which will be discussed more in Section 9.4. The reason for presenting the range and interquartile range was that it is a more appropriate method for presenting the variance in the data for lower amounts of samples. Since only seven samples were taken, deriving the standard deviation would lead to misleading results. Nevertheless, the range is 1.65 mm in Y, and notice how the deviations in translations in Y are roughly four times as large as the ones in X. This is most likely due to the problematic perspective over the jogging. To combat this issue, one would either have to incorporate a better view over the jogging, or, replace the human with additional sensors, like the other concepts discussed in Section 6.1.2.

9.3.3 Error adaptation

By introducing errors to the system, we wanted to see how well the system adapts to those errors. This was done by changing the placement of the robot base in Automation Studio. We added 50 mm to the x and y directions and expected to find this error when running the calibration process. As can be seen in Table 8.7, both in the x and y axes, these errors were found, with a 1.88 mm inaccuracy at most. This test demonstrates that, although not perfectly accurate, the calibration can pick up on errors between the physical system and its drawings. If a value far from 50 mm is discovered, the calibration process would not hold and therefore be useless as it means the process could not find large errors. For sample 2 in Table 8.6, Y was found to be -33.75 and not -50 mm. This is because of the already existing error of, on average, 18.13 mm. Therefore, if the system had been compensated for the existing error, we would find that the new error would be close to -50 mm. Instead, since the existing error has not been compensated for, we include the mean value and calculate the difference. This is the result we see in Table 8.7.

9.4 Sources of errors

9.4.1 Jogging

The jogging process has a few sources of errors. One source is discrete jogging, as the robot program can only jog at a minimum of one-millimeter accuracy. This forces the operator to make judgment calls if one-millimeter accuracy is insufficient. A solution is to decrease each move from 1 mm to 0.5 mm. There are times when a movement of 0.5 mm would be preferred over 1 mm, thus increasing the precision of the process. Another source is the perspective on the jogging. In our setup, a barricade protects the operator from the machine, which prevents the operator from getting a full view of the TCP. This mostly affected jogging in the y-axis, since that axis is parallel to the operator's perspective. Therefore, placing the TCP accurately in the y direction

is more difficult, than in the x direction. This notion is supported by the results in Table 8.5, where the values in Y deviate more than the ones in X. Since the sheet of paper started and stopped at the same coordinates in each sample, there should be a minimal effect by the camera on the results, and most of the sources of errors should be caused by the jogging part of the process. This, therefore, supports the idea that the perspective affects the precision, in our case, this was most noticeable in Y. This, in turn, creates longer decision times where the operator may second guess if their decision is accurate. Furthermore, this increases the tediousness and boredom, which fuels more inaccuracy. The process therefore has to be simple enough so that an operator can make accurate readings while jogging.

9.4.2 Conveyor

The belt on the conveyor has a concave shape, where the edges of the width are slightly higher than in the middle of the belt. Contrary to our assumption that the belt is perfectly flat. This is not an issue as the belt is soft and can be pressed down using the TCP until it is flat. Also, the conveyor belt has varying support underneath, due to alternating spaces between rolling pins. Theoretically, this should impact the height values. However, this was not something we noticed, as the tension in the belt was high and the heights measured were constant.

9.4.3 Hardware

Another source of inaccuracy is the precision of the hardware. As seen in Table 8.4, the camera precision was very accurate. The σ was around 15 milli pixels. To get an idea of the magnitude of the accuracy, with a conversion rate of $\mu = 0.3$, which is a reasonable conversion number for our process, that would entail a precision of 3 μm. This is very accurate compared to the jogging procedure which was, as mentioned, done with a millimeter accuracy. However, the camera accuracy was measured while the paper was stationary. How large of an effect movement has on the accuracy, was not measured in this project. However, when viewing Figure 8.4 c), one would expect the centroids to be on a straight line since the conveyor is assumed to be straight. As we can see, this is not the case. This means that some error in the image processing has been introduced, which is larger than the error measured in Table 8.4. This is likely because the images are acquired while the paper is moving. Furthermore, the accuracy of the delta robot was 0.1 mm. These are the most important limitations in the hardware, which will limit the accuracy of the process. To improve the system it would be most appropriate to introduce a smaller step size than 1 millimeter in the jogging procedure, but also, to improve the concept *Following the object* by acquiring more images. Only ten images were used for analyzing the camera angle, which we found to vary by 0.75 mm in Y from the average, and a simple improvement would therefore be to take more images. Unfortunately, this would slow down the process since one would have to decrease the speed of the conveyor belt while acquiring images. On the other hand, this increase in time is still relatively short compared to the jogging sequence. Furthermore, the process would also benefit from this, because the paper having a lower velocity would increase accuracy.

The real position of the machine and camera is somewhat unknown. There is a theoretical setup in Robot Studio that was followed precisely, but when manually calibrating the setup during installation, there may have been some unknown deviations introduced that were not documented.

10 Conclusions and future work

10.1 Conclusion

This thesis has presented and implemented a method that could reduce the need for manually tuning a pick-and-place process. Through the integration of software algorithms, that take advantage of available sensors, the errors within the system were identified. The process was demonstrated to be fast and simple to use, and since no additional sensors were integrated, it was cheap to implement.

The experimental results show that the proposed method lies within a margin of a few millimeters, which is acceptable in several industries. This outcome highlights the potential of this method, in addition to its simplicity and fast execution time.

Further development of this method can go down two paths. Either alleviating the need for jogging by including new sensors, or, making adjustments in the jogging process.

In conclusion, the proposed method offers a simple, cost-effective, and fast calibration process, presenting a potential alternative to manual tuning. This thesis, therefore, represents a promising step towards increasing productivity in the pick-and-place process, by reducing the need for manual calibration.

10.2 Future work

10.2.1 Sensors

Implementing sensors instead of jogging will increase the level of autonomy in the solution. This is in large the purpose behind this thesis and a valuable goal for B&R. The difficulty of implementing this is somewhat economically driven, as the sensors ideally would only be used once at the initial setup. Therefore, sensors should rarely require alterations in other design aspects, such as allowing cords to be drawn inside the machine workspace. This would instead require wireless communication which is not supported by B&R PLCs. Also, the sensor in itself has to be designed and produced. This labor proved to be outside the scope of this project as the time constrain forced us to prioritise other things, but the conclusions drawn from this work may continue in a sensory development by B&R. Smaller electronic sensors are largely cheaper than cameras which may prove sufficient to justify their low amount of use per part. Our tests were held back in numbers due to the amount of time it took to complete one test, between 10-15 minutes. By implementing sensors one could perform tests automatically, providing samples while the operator focuses on other tasks.

10.2.2 Mathematical operations

To implement this process fully on a B&R PLC, there have to be more options for calculations within structured text or in C. It was theoretically possible to import C-libraries to apply linear algebra functions, but this proved difficult for us to import to our project because of constraints that Automation Studio has set on libraries. Nevertheless, if one were to implement more mathematical functions in Automation Studio, it would reduce the need for human interaction by removing the use of the Python program that has manual data imports.

As the lack of scale is known, it is also possible to alter our data to fit the expected answer of only translation and rotation. By for example normalizing distances and creating orthogonal vectors a cleaner representation can be created where the data behaves as we expect it to. However, this has the issue of being biased towards the expected result and not what the data is describing.

10.2.3 Jogging measurements

In our process, jogging was moved in steps of one and twenty millimeters. A relatively easy solution to increase jogging accuracy is to decrease the step size and utilize the HMI to switch back and forth between step sizes. The reason the step was relatively large, was to make the robot faster to move, but this came at the expense of measurement errors. By implementing smaller step sizes, the operator would solely be hindered by their eye measurements and not by the step size. In combination with this, it would be beneficial to decrease the width of the crosses on the paper, providing an increased visual aid for the operator. In addition, one could implement a laser for tracking the position of the TCP. As mentioned, the perspective of the TCP can influence how well it is placed, especially in the y direction. One could place an embedded laser in the 3D-printed pointer to alleviate this issue. An example of this solution can be seen in Figure 10.1. This would prevent the pointer from blocking the view in the y direction. One could move until the laser is in the middle of the cross, and then move down to the correct height.

Furthermore, it would also be beneficial to increase the number of points that were measured while jogging, since this would decrease the variance of the results. This was not implemented since we had to perform many tests and wanted to minimize the lead time. Also, measuring at different places on the conveyor, instead of only one, would be beneficial. In doing so, the measurements would cover more of the conveyor belt, which increases accuracy along the conveyor belt. This would impact β and γ angles the most, as they would be calculated with larger levers.

Another notable aspect of the jogging process is how time-consuming it was. Moving with a 20 mm step size between points at most a 150 mm distance apart, took a long time. What could have been done better is, since we know the distance between points on the paper, have pre-set distances that the operator could choose to move in between points, and therefore not have to move in 20 mm steps.

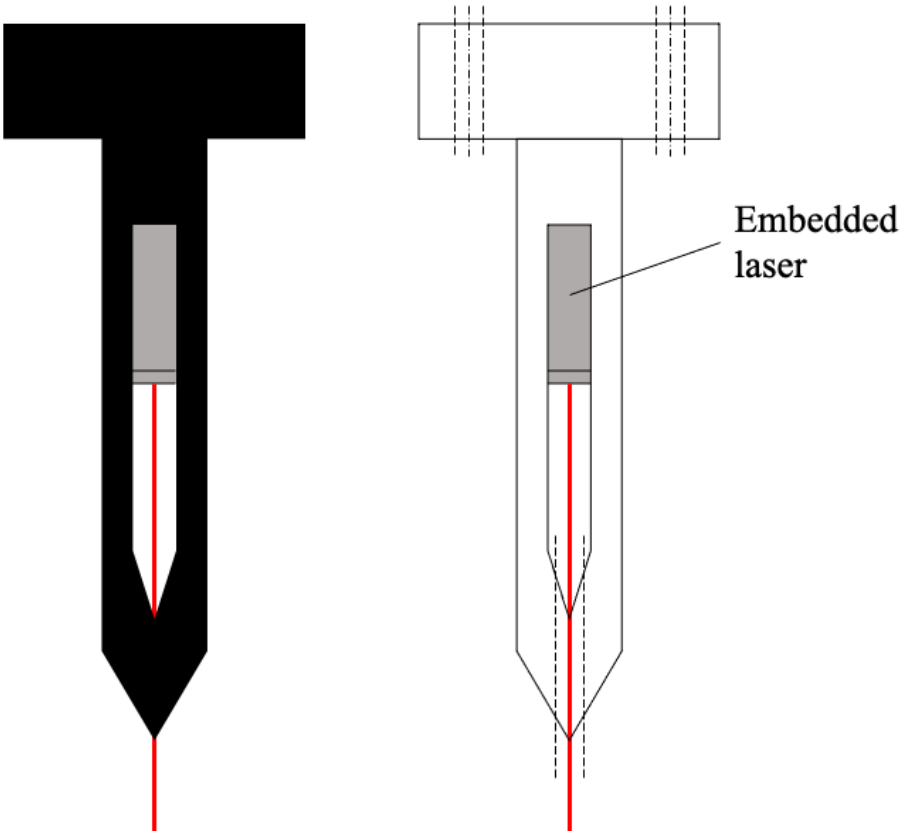


Figure 10.1: An example of how to add an embedded laser into the 3D pointer.

10.2.4 Camera

Our process used the concept *Following the object* for the path estimation. If *Following the conveyor belt* was used instead, one would only have to take one picture. To be able to take ten images while the conveyor belt is running, in the case of *Following the object*, the paper has to be rather small. Otherwise, the paper will move out of the camera view before acquiring all ten images. However, by using *Following the conveyor belt* only one image has to be acquired and one can therefore design a bigger paper, which was previously described as better at maintaining scale values. This is due to the points covering a larger area, which lessens the effect of errors.

Another area that could have been improved upon, is the pixel conversion. At this stage, the pixel conversion is derived based on the first image. Since the projective transformation was not successfully implemented, the pixel conversion number μ will differ depending on where in the image the paper is situated. Therefore, the first image's conversion number does not represent the conversion number throughout the entire image. Thus, a simple solution is to derive the conversion number for each image, instead of basing it on only the first image. Or, if the projective transformation is successfully implemented, then it is more acceptable to base it on one image.

10.2.5 Projective transformation

Currently, the projective transformation had unsatisfactory results, where it worsened the standard deviations for Y translations. This is because the projective transformation increased the variance of the resulting θ from the path estimation. A better-developed projective transform should work more in tandem with the path estimation. An example would be integrating θ as a rotation in the projective transform, meaning that the camera is perfectly oriented along the conveyor and the prediction P' in Figure 5.1 would always go straight in the x-axis.

10.2.6 Sheet design

The sheet has to be redesigned if some of the mentioned improvements are to be implemented. For example, since the concept *Following the conveyor belt* allows for a larger sheet, the sheet's size can be increased. An operator may want to measure more points and invest more time into the calibration. This would therefore call for the sheet of paper to be marked with more black squares. Also, as mentioned, thinner crosses would improve aim as the center point is narrower, this would therefore also call for a change of the sheet of paper.

It may also a good idea to use something thicker with more weight than a sheet of paper. This could solve issues with the belt's concave shape, as it would flatten it out. It would also not be as soft to press into if there was an even pressure from the plate. For example, a Medium Density Fibreboard (MDF) that consists of compressed wood fiber would be a good choice. It is cheap and flat with some weight to it.

References

[1] Deyong Shang, Yuwei Wang, Zhiyuan Yang, Junjie Wang and Yue Liu. "Study on Comprehensive Calibration and Image Sieving for Coal-Gangue Separation Parallel Robot". In: *Applied Sciences* 10.20 (2020). URL: https://www.mdpi.com/2076-3417/10/20/7059.

[2] B&R. *B&R has simplified the development of pick-and-place applications*. 2020. URL: https://www.br-automation.com/sv/foeretaget/press-room/more-productive-with-ready-made-software-components-14-09-2020/ (visited on 15/05/2024).

[3] Mike Cable. "Calibration: A Technician's Guide". In: (2005).

[4] Codian. *D4-1100-HD021-RH090*. URL: https://codian-robotics.com/blog/producten/d4-1100-hd021-rh090/ (visited on 16/05/2024).

[5] B&R Industrial automation. *8LSA35.DB030S000-3*. URL: https://www.br-automation.com/sv/produkter/8lsa35db030s000-3/ (visited on 16/05/2024).

[6] Robert Collins. *Lecture 16: Planar Homographies*. 2007. URL: https://www.cse.psu.edu/~rtc12/CSE486/lecture16.pdf (visited on 16/05/2024).

[7] Kerstin Vännman and Adam Jonsson. *Matematisk Statistik*. 3rd ed. Lund, Sweden: Studentlitteratur, 2021.

[8] Leonid Freidovich. *Control Methods For Robotic Applications*. Umeå University, 2017.

[9] Anders Holst and Victor Ulfnarovski. *Matrix Theory*. 1st ed. Lund, Sweden: Studentlitteratur, 2017.

[10] Gregory G Slabaugh. "Computing Euler angles from a rotation matrix". In: *Retrieved on August* 6.2000 (1999), pp. 39–63.

[11] Lionel Brits. *Euler Angles*. 2008. URL: https://commons.wikimedia.org/wiki/File:Eulerangles.svg (visited on 15/05/2024).

[12] Scipy. *Scipy: Rotation.as_euler*. 2024. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.transform.Rotation.as_euler.html#scipy.spatial.transform.Rotation.as_euler (visited on 10/05/2024).