CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3

Faculty of Electrical Engineering
Department of Cybernetics

Bachelor's Thesis

Calibration of the Manipulator Position in the Robotic Cell

Filip Gregor

May 2023

Supervisor: Ing. Vladimír Smutný, Ph.D.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Gregor Filip** Personal ID number: **491902**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Calibration of the Manipulator Position in the Robotic Cell

Bachelor’s thesis title in Czech:

Kalibrace polohy robotu v buňce

Guidelines:

1. Get familiar with the arrangement and parameters of the robotic cell in the Rohde&Schwarz plant.
2. Research existing method of camera and robot calibration and relative calibration of camera and robot.
3. Propose the method to calibrate robot position relative to the robotic cell using gripper mounted camera.
4. Implement, test, and document the proposed method.
5. Evaluate the results, make conclusions.

Bibliography / sources:

[1] HELLER, Jan. Global Optimization Techniques in Camera-Robot Calibration. Prague, 2015. Doctoral Thesis. Center for Machine Perception Department of Cybernetics Faculty of Electrical Engineering Czech Technical University in Prague. Vedoucí práce Ing. Tomáš Pajdla, Ph.D.
[2] DANIILIDIS, Konstantinos. Hand-Eye Calibration Using Dual Quaternions. The International Journal of Robotics Research. 1999, 18(3), 286-298. ISSN 0278-3649. Dostupné z: doi:10.1177/02783649922066213
[3] ENEBUSE, Ikenna, Mathias FOO, Babul Salam Ksm Kader IBRAHIM, Hafiz AHMED, Fhon SUPMAK a Odongo Steven EYOBU. A Comparative Review of Hand-Eye Calibration Techniques for Vision Guided Robots. IEEE Access. 2021, 9, 113143-113155. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2021.3104514.
[4] ZHUANG, Hanqi, Zvi S. ROTH, Xuan XU a Kuanchih WANG. Camera calibration issues in robot calibration with eye-on-hand configuration. Robotics and Computer-Integrated Manufacturing. 1993, 10(6), 401-412. ISSN 07365845. Dostupné z: doi:10.1016/0736-5845(93)90003-3
[5] Bradski, G. & Kaehler, A., 2008. Learning OpenCV: Computer vision with the OpenCV library, " O'Reilly Media, Inc."

Name and workplace of bachelor’s thesis supervisor:

Ing. Vladimír Smutný, Ph.D. Robotic Perception CIIRC

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **11.05.2022** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **19.02.2024**

Ing. Vladimír Smutný, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt	Student's signature

Acknowledgement / Declaration

I want to express my deepest thanks to Ing. Vladimír Smutný, Ph.D., for all the time, consultation, and help in creating this bachelor's thesis.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 26. 5. 2023

.....

Abstrakt / Abstract

Tato práce se zabývá kalibrováním polohy průmyslového manipulátoru v pracovní buňce. Systém bude využívat kameru pevně připojenou k chapadlu manipulátoru. V práci je zvolen vhodný druh kalibračních značek a postup snímání kalibračních dat. Jsou zvoleny vhodné kalibrační algoritmy pro hledání jednotlivých transformací. Tyto algoritmy jsou naimplementovány a popsány. Nakonec je provedeno ověření funkčnosti na robotu A465 a měření přesnosti. U naměřených dat je provedena diskuze výsledků.

Klíčová slova: OpenCV, kalibrace kamery, kalibrace oko-ruka, kalibrace z báze robotu do světového souřadnicového systému

This work deals with calibrating an industrial manipulator position in a work cell. The system will use a camera attached to the manipulator end-effector. The thesis selects a suitable type of calibration marker and proposes the procedure for collecting calibration data. Appropriate calibration algorithms are chosen for the estimation of individual transformations. These algorithms are implemented and described in detail. The functionality is verified on the A465 robot, and the accuracy is evaluated. The obtained data are discussed, and the results are analyzed.

Keywords: OpenCV, Camera calibration, Hand-Eye calibration, Robot base to world calibration

Contents /

Tables / Figures

Chapter 1
Introduction

Industrial manipulators are a standard part of production processes today, thanks to their ability to perform repetitive tasks quickly and accurately. In order to achieve high accuracy in the performed tasks, it is necessary to calibrate the robots during their deployment and subsequent operation. One of the most popular approach to robot calibration is calibration using cameras and calibration markers.

This work aims to design and test the calibration methods of a robotic manipulator for the needs of the Rohde & Schwarz production plant. The first task is to propose a method of calibrating the camera and the Hand-eye calibration when putting the robot into operation. The second task is to design and test the Robot-World calibration method for the periodic calibration of the robot position in the work cell and the possibility of verifying the correctly entered production task by the production operator.

A camera mounted on the arm of the manipulator and calibration markers will be used for these tasks. The OpenCV library's tools will be used to search for markers in images and subsequent calculations. The result of the work will be three separate programs. The first program will calculate the camera's internal parameters from the captured data and the transformation from the camera to the robot gripper. The second program calculates the transformation of the robot coordinate system to the world coordinate system from the captured markers in the robot's workspace. A third program, according to a marker placed in the work cell, can check whether the correct production task has been selected by the production operator.

1.1 Description of the problem

The Rohde & Schwarz Vimperk production plant focuses on the production of innovative products in the field of measuring, broadcasting, and media technology [1]. These products are manufactured using a hybrid production process that incorporates both manual and automated techniques on a medium scale. Industrial manipulators are rarely used due to high acquisition costs and lower workloads, which typically result in long-term return on investment.

There are options for increasing the utilization of the industrial manipulator by moving it between multiple workstations. Rohde & Schwarz decided to place an industrial manipulator on a mobile robot that would move between individual work cells. Moving the robotic manipulator caused an error in its position relative to the work cell. For this reason, it is necessary to find the exact position of the manipulator in relation to the work cell after each relocation of the robot. An automatic method of finding the manipulator's position in the work cell must be created for this task. This bachelor's thesis will deal with this problem.

Chapter 2
Proposed solution

This chapter focuses on the equipment, data collection, and algorithms used to solve the given task. First, the individual transformations and coordinate systems used in this work will be described. Then the markers' types, strengths, and weaknesses will be discussed. Subsequently, the principles of data collection and the conditions under which it is carried out will be described. In the end, algorithms designed for calculating individual transformations will be presented.

2.1 Description of transformations and coordinate systems

This work uses several coordinate systems and transformation matrices to determine the position and orientation of the robotic manipulator, camera, and other elements. These coordinate systems are connected by transformation matrices that enable the conversion between individual coordinate systems. This list contains all coordinate systems used in this work:

- BRF: The base robot frame coordinate system is connected to the robot's base, and each robot manipulator defines it in its program.
- BG: The gripper coordinate system is defined as a system connected to the flange at the end of the robotic manipulator.
- BC: The camera coordinate system is defined as a system located at the camera's position in space, with the z-axis pointing in the direction of the camera's view.
- BM: The marker base is defined as the coordinate system located at the position of the marker in space. The position of the coordinate system is determined depending on the specific type of marker.
- BW: The world base coordinate system is defined as a coordinate system located at some defined point. The work cell designer determines this coordinate system.

The following list shows all the transformations used in this work and how to obtain a specific transformation:

- T_{BG}^{BRF}: The transformation matrix from the gripper to the robot base is obtained by calculating the forward kinematic task using known robot joint coordinates.
- T_{BG}^{BC}: The transformation matrix from the griper to the camera is obtained by calculating the hand-eye calibration using the Static-methods program.
- T_{BC}^{BM}: The transformation matrix from the camera to the marker is obtained using the `solvePnP()` function from the OpenCV library [2].
- T_{BM}^{BW}: The transformation matrix from the marker coordinate system to the world coordinate system is calculated using the known position of the marker in the world coordinate system.
- T_{BW}^{BRF}: The transformation matrix from the world coordinate system to the robot base will be calculated using the Robot-world program.

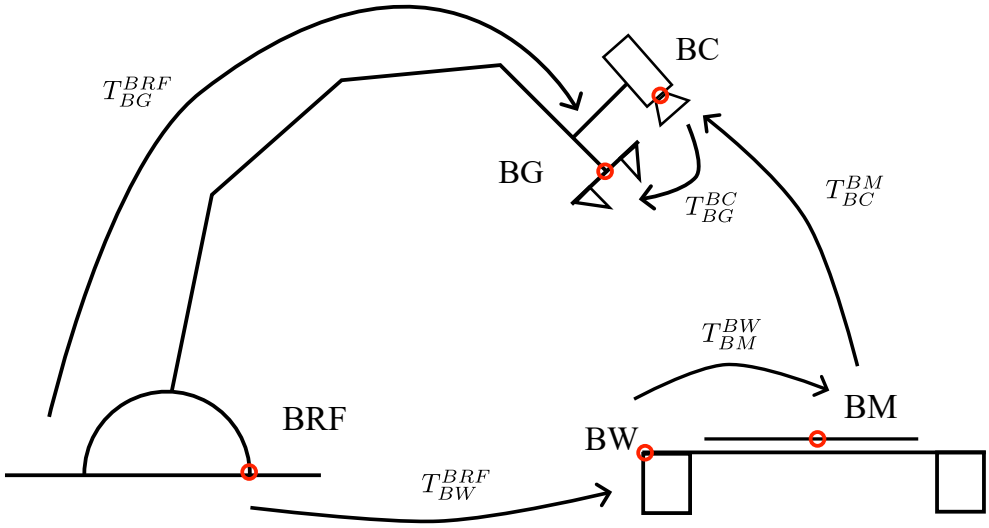

Figure 2.1. Schematic of the robot with coordinate systems and transformations.

2.2 Used equipment

2.2.1 A465 robotic manipulator

The A465 robotic manipulator is an industrial manipulator for handling light objects manufactured by CRS. It is an industrial manipulator with six axes of freedom. This manipulator is located in the CIRC CTU building and was used in this work to test the functionality of individual programs. The manipulator is not controlled by the original control unit but by the Mars 8 control unit [3]. Primary data about the robot are shown in the Table 2.1[4].

Parameters	Value
Repeatability	±0.05 mm
Reach	711 mm
Payload	2 kg

Table 2.1. A465 robotic manipulator parameters [4].

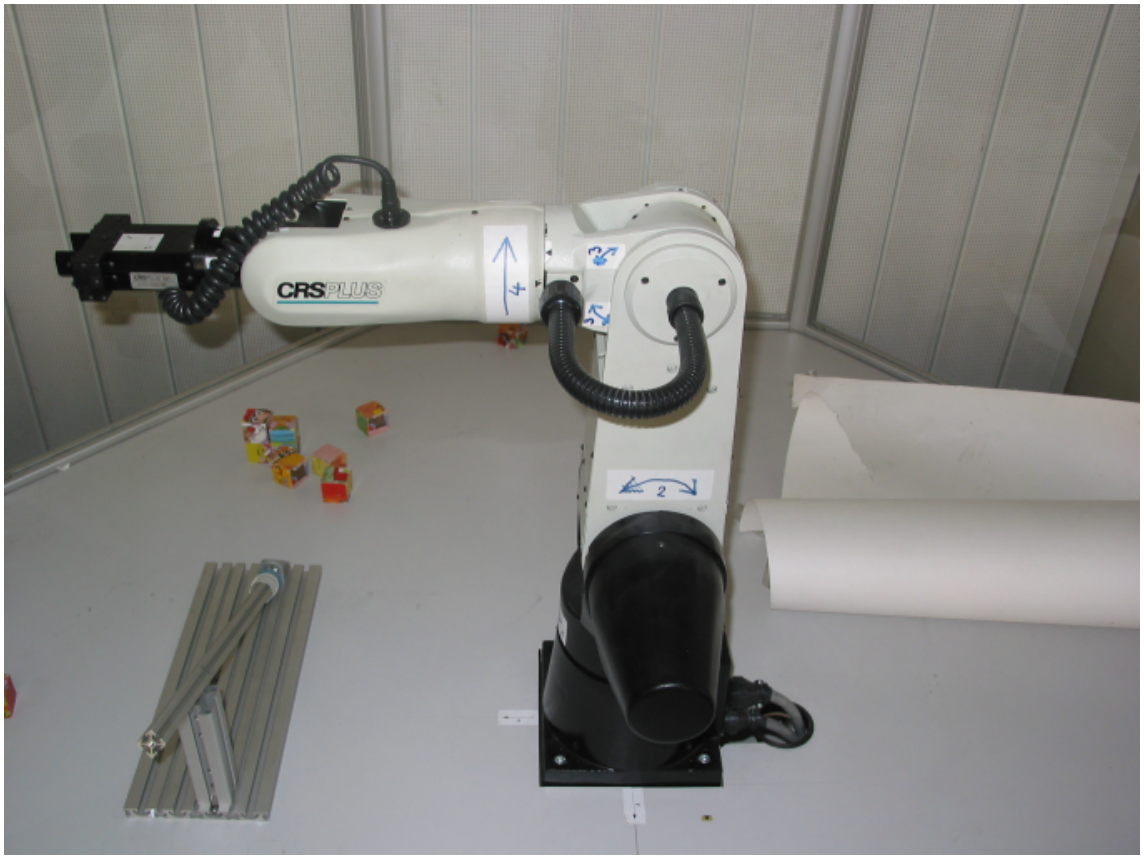

Figure 2.2. Photo of the A465 robotic manipulator in the work cell [5].

2.2.2 UR10 e robotic manipulator

The UR10e robotic manipulator is a collaborative industrial manipulator manufactured by Universal Robots. It is an industrial manipulator with six degrees of freedom. This robotic manipulator is used in the Rohde & Schwarz manufacturing plant, and this work is designed to be deployed on this type of robot. Primary data about the robot are listed in the Table 2.2[6].

Parameters	Value
Repeatability	±0.05 mm
Reach	1 300 mm
Payload	12.5 kg

Table 2.2. UR10 e robotic manipulator parameters [6].

Figure 2.3. Photo of the UR10 e robotic manipulator [7].

2.2.3 Camera and lens

A Basler camera and lens were used to collect calibration data at the Rohde & Schwarz plant. Specifically, the daA3840-45um camera model and the Basler Lens C125-0618-5M-P lens with a focal length of 6mm were used. The technical specifications of both components can be found in the attached Table 2.3.

Parameters	Value
Camera resolution	8.3 MP
Pixel Size	2 µm x 2 µm
Focal Length	6 mm

Table 2.3. Camera and lens parameters [8][9].

2.3 Calibration targets

Calibration markers intended for machine vision calibration are high-contrast black-and-white markers. These markers are designed for easy recognition using machine vision algorithms. Their primary purpose is to quickly and accurately determine the position of the given marker in the image. Another task is to determine the reference size for other objects in the image. Some markers also have a coded number in the images, which allows automatic marker identification. We can divide these markers into several types.

2.3.1 Calibration marker

Calibration markers are circular or square marks containing one or more reference points. There are many types of tags, such as ArUco tags, ApriTags, or CCTags. The main difference between the markers is their accuracy during calibration and the ability to store additional information. A typical representative of a marker that can store information is the ArUco marker [10].

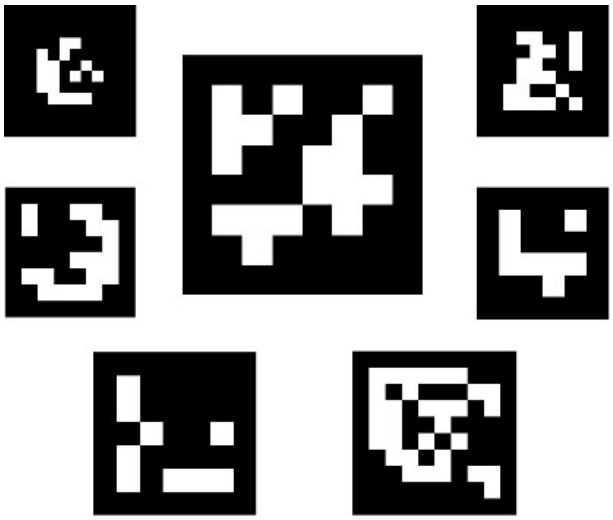

Figure 2.4. Example of ArUco markers [11].

2.3.2 Calibration boards

Calibration plates can be described as a pre-defined grid of markers. Its main advantage is to provide multiple reference points for one image. The most commonly used calibration boards are Checkerboard, Circle board, Asymmetric circle board, and ChArUco board. The Circle board and The Asymmetric circle board are grids made of dots. We are looking for the calibration points in the middle of the dots. The circle board is rotationally invariant when viewed from multiple cameras. Therefore an Asymmetric circular board is used more often, which has already solved this problem. The main disadvantage is that the entire grid has to be visible in the image. The Checkerboard and ChArUco board make a checkerboard pattern. The sought calibration points are on the vertices of the squares. The disadvantage of the Checkerboard grid, as with the dot patterns, is that in order to use the pattern successfully, we need to see the entire calibration pattern in the picture. This problem is solved by the ChArUco board, which places smaller ArUco markers in the white fields, thanks to which calibration can be carried out even with a partially visible board [12].

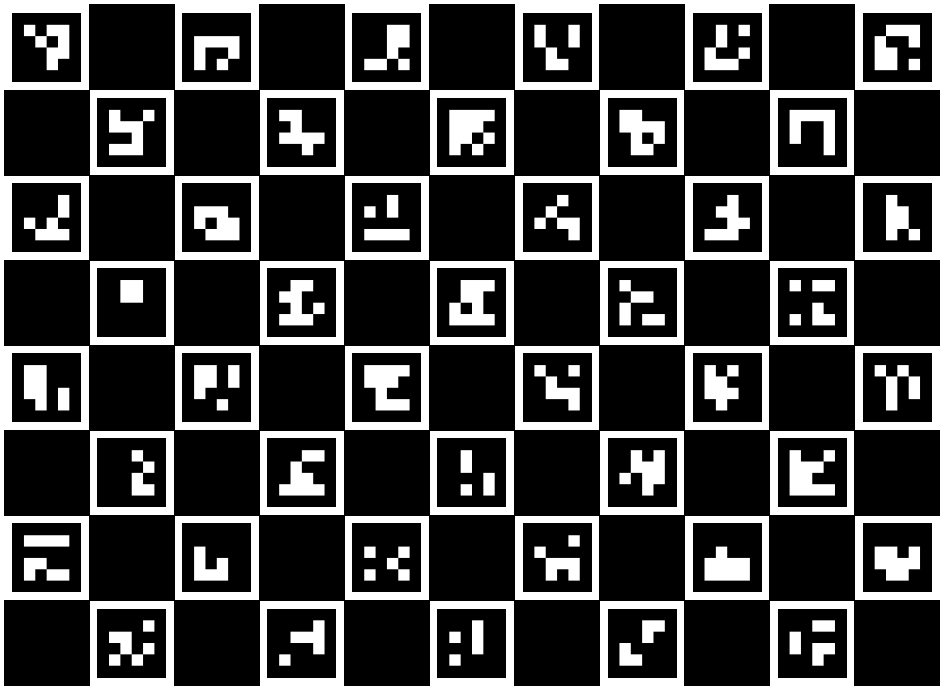

Figure 2.5. Example of ChArUco board [13].

2.3.3 Used markers for this project

For the realization of this work, we chose two types of markers. Camera calibration and Hand-Eye calibration will use the ChArUco board. The main reason for choosing this solution is the need for at least six reference points for a robust Hand-Eye calibration result [14]. For this reason, it is more appropriate to use calibration boards than calibration markers. Another reason to use the ChArUco board is the fixed camera parameters set by the client. The limiting parameter is the camera's field of view, which limits how many reference points can be seen at once. To choose the size of the squares, choosing a short edge of the square for more markers per image is ideal, but this edge is limited in recognizing ArUco markers. The ideal value for the specified assembly is 2.5 cm.

We chose the ArUco markers for the Robot-World calibration. The main reason is the limited space in the working area of the robot around the device, which does not allow the use of calibration boards. Another reason is the pre-defined detection functions in OpenCV for their detection and the possibility of loading a specific tag ID.

2.4 Calibration positions

Collecting calibration data is essential when using a camera attached to the end of a robotic manipulator. We use calibration data to calculate the necessary transformations to control the robotic manipulator. We will guide the robotic manipulator to predetermined positions for collecting images containing calibration markers. Data collection

must meet strict conditions for the successful and accurate execution of all necessary calibrations.

2.4.1 External conditions for calibration

The primary condition determining the quality of captured data is light. A sufficiently robust and consistent light source across the entire calibration board is ideal. The reason for a powerful light source is the possibility of using a smaller aperture. Another condition for correct marker detection is that the light does not reflect from the surface of the calibration board so as not to limit the possibility of detecting markers. The important condition for correctly captured data is that the position of the calibration board must not be changed during data capture. The last rule is that the calibrated camera must not be modified in any way between calibration positions [15].

2.4.2 Rules and Procedures for Accurate Calibration Data Collection

For accurate and robust hand-eye and camera calibration, taking a sufficient number of calibration images is crucial. We need at least ten images, although the exact number can vary depending on the number of calibration points seen in individual images. To ensure that the calibration data is evenly distributed, the individual positions must differ from each other by at least 30 degrees in at least two axes of rotation. This rule will help ensure that the calibration data is accurate across the full range of motion. Another essential rule to follow is to use the position found from the data read from the values of the individual joints of the robotic manipulator to obtain data about the gripper's position. However, to accurately read the values of the joints, it is necessary first to wait for them to settle in the position of the robotic manipulator [15].

2.4.3 Collection of calibration data

Positions for camera and hand-eye calibration were chosen by collecting points on the upper half of the sphere where the center of the sphere lies in the center of the calibration board. We will intersect this sphere with three planes parallel to the calibration board's plane. These planes will be above each other, and the distance of the individual planes from the calibration board is set so that the segment from the center of the calibration board to the intersection of the sphere with the plane forms the following angles 30, 55, and 75 degrees. Subsequently, the camera is always turned towards the center of the sphere. The side section of the plane and the sphere can be seen in the following diagram Figure 2.6. We will divide the circle obtained by the intersection of the sphere and the surface into eight segments of equal length on which we will perform individual measurements. We perform the last measurement on the sphere directly above the center of the sphere. In this way, we obtain 19 calibration positions. We calculate the individual positions using the following equations:

$$x = h\cos\left(a\frac{360^\circ}{n}\right) + S_x, \tag{1}$$

$$y = h\sin\left(a\frac{360^\circ}{n}\right) + S_y, \tag{2}$$

$$z = h\sin\left(\theta\right) + S_z, \tag{3}$$

$$\alpha = 90^\circ - \theta, \tag{4}$$

$$\beta = -a\frac{360^\circ}{n}, \tag{5}$$

$$\gamma = a30^\circ. \tag{6}$$

In these equations, a is from $\{1, .., 8\}$ and describes how many points we look for on individual circles. The variable n indicates the number of points on the circle, in our case, eight positions. The variable h indicates the sphere's radius, and the variable θ indicates the angle between the calibration board and the segment with the center of the calibration board to the camera position.

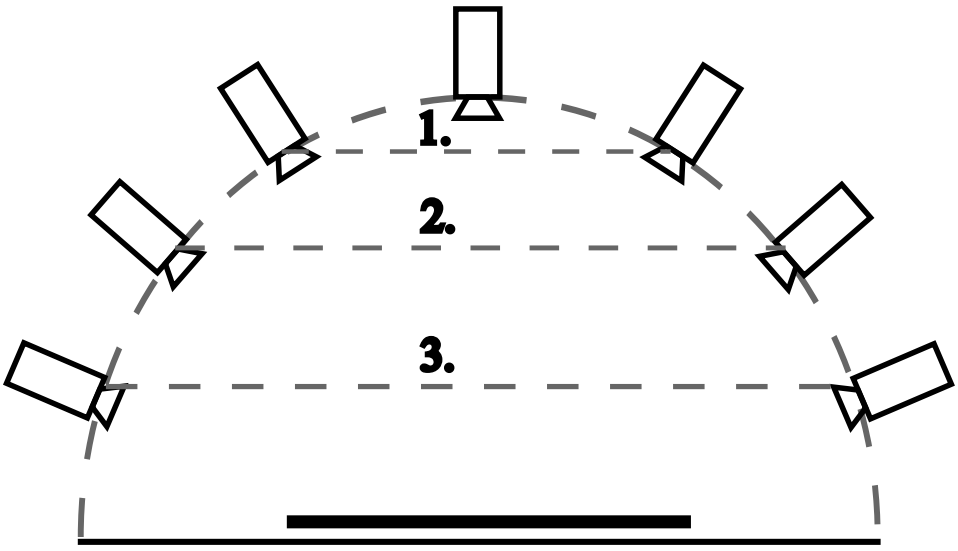

Figure 2.6. Vertical cross-section of data collection positions.

For robot-world calibration, we will guide the robot to three calibration positions. These positions will be placed above the marker at distances corresponding to the maximum sharpness of the camera.

2.5 Calibration

The calibration of robotic manipulators is a critical task in robotics, as it ensures accuracy and precision during their operation. The upcoming section will specifically focus on calibrating the manipulator base position in the work cell using cameras. There are two possible configurations for the relative location of the manipulator and the camera: the camera can either be situated at the end of the manipulator or on a separate platform attached to the world coordinate system. This work will only deal with the camera located at the end effector of the manipulator.

2.5.1 Camera calibration

Camera calibration deals with finding the internal and external parameters of the camera. These relationships are essential for connecting the two-dimensional world in the image with the three-dimensional world in front of the camera. We will use the pinhole

camera model to describe the camera and then construct the equations. This model is shown in diagram Figure 2.7. This model linearly approximates the behavior of a real camera. The following equation defines the relationship between points in the camera $[u \quad v]^T$ and points in space $[x \quad y \quad z]^T$ Equation (7)[14][16].

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K [R \quad t] \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{7}$$

R denotes the rotation matrix from the world coordinate system to the camera coordinate system, and t is the translation vector from the world coordinate system to the camera coordinate system. Furthermore, $[u \quad v]^T$ indicate the point position in the camera, and $[x \quad y \quad z]^T$ determine the point position in the world coordinate system. The variable s denotes the scaling factor, and the matrix K denotes the camera's internal parameters.

The matrix K includes the internal parameters of the camera. These internal parameters are defined as follows:

- f_u, f_v: These parameters indicate the focal length in pixels.
- x_0, y_0: These parameters are the coordinates of the principal point.
- s: This parameter indicates the skew of the image.

$$K = \begin{bmatrix} f_u & s & x_0 \\ 0 & f_v & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

This equation only works if the camera has a distortion-free lens. Real lenses have distortion, which we can approximate by radial and tangential distortion. For this reason, we also need to find the external camera parameters. The algorithm from the article [17], which is implemented in the OpenCV library, was used to calculate these parameters.

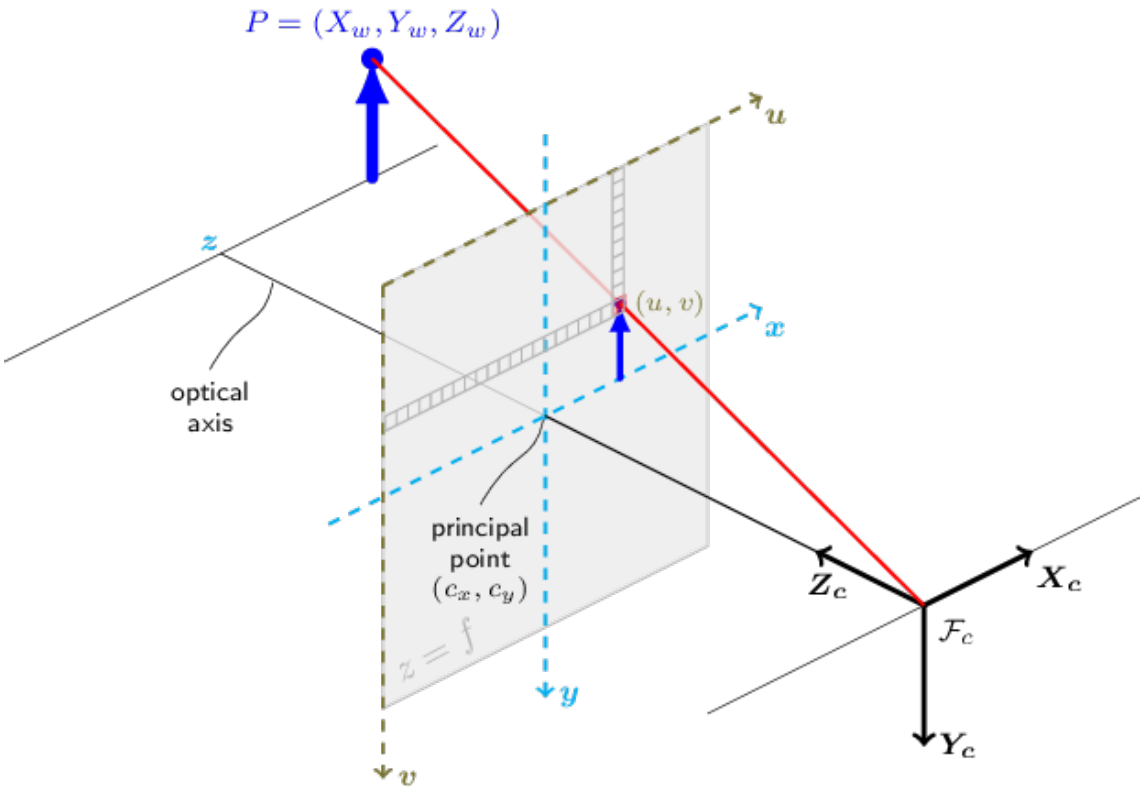

Figure 2.7. Schematic of the image in the camera with the camera's coordinate system [18].

2.5.2 Hand-Eye calibration

Hand-eye calibration aims to find the transformation matrix from the camera coordinates system to the gripper coordinates system. This transformation is used in a task where a robotic manipulator lifts an object detected by a camera. To calculate this transformation, it will need a transformation from the gripper coordinate system to the robot base coordinate system. We will calculate this transformation using the results of the forward kinematic task provided by the robot software. We will use the following conversion to convert the rotation from the Euler angles in the zyx orientation to the rotation Matrix (9). The second transformation that we will need for the calculation is from the calibration board coordinate system to the camera coordinate system. We get this transformation when calibrating the camera.

$$\begin{bmatrix} \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\beta)\sin(\gamma) - \cos(\gamma)\sin(\alpha) & \cos(\alpha)\cos(\gamma)\sin(\beta) - \sin(\alpha)\sin(\gamma) \\ \cos(\beta)\sin(\alpha) & \cos(\alpha)\cos(\gamma) + \sin(\alpha)\sin(\beta)\sin(\gamma) & \cos(\gamma)\sin(\alpha)\sin(\beta) - \cos(\alpha)\sin(\gamma) \\ -\sin(\beta) & \cos(\beta)\sin(\gamma) & \cos(\beta)\cos(\gamma) \end{bmatrix} \tag{9}$$

To calculate the hand-eye calibration, we will use the measured transformations from which we will make the following Equation (10). These equations correspond to the transformations marked in the following diagram Figure 2.1. Subsequently, we can substitute this and obtain Equation (11), where i denotes the i-th measured data. This adjustment gives us a system of linear equations, which we can use to find the solution with standards methods. The solution methods can be distinguished into several types: with a iterative solution first of rotation and then translation, simultaneous estimation

of both values, or optimized simultaneity solution of both. From the possible methods, we chose the Dual quaternions method for this work due to the highest measured accuracy in the articles [19][12].

$$(T_{BRF}^{BG2})^{-1} T_{BRF}^{BG1} T_{BG}^{BC} = T_{BG}^{BC} T_{BC2}^{BM} (T_{BC1}^{BM})^{-1} \tag{10}$$

$$A_i X = X B_i \tag{11}$$

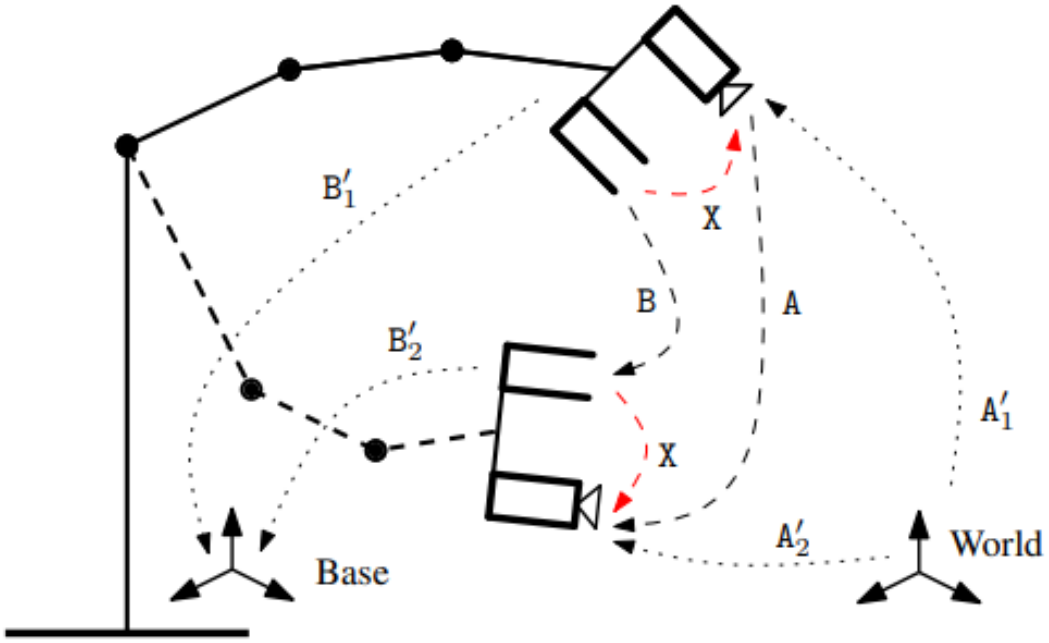

Figure 2.8. Schematic of hand-eye calibration [16].

2.5.3 Robot-World calibration

The robot-world calibration deals with finding a transformation from the robot's base coordinate system to the world's coordinate system of our choice. We use this transformation to guide the manipulator to a given point specified in world coordinates. To build an optimization task, we must first get the transformation from the robot base coordinates to the gripper coordinates. We obtain this transformation matrix in the same way as in hand-eye calibration task. Next, we will need the transformation from the camera to the gripper, which we calculated using hand-eye calibration. Subsequently, we must find a transformation between the world coordinate system and the camera coordinate system.

To obtain the following transformation matrices, we will use Aruko markers, which we will place around the perimeter of the device as marked on the diagram Figure 2.9. We find these markers in individual images and find the transformation from the coordinate system of the marker to the camera's coordinate system.

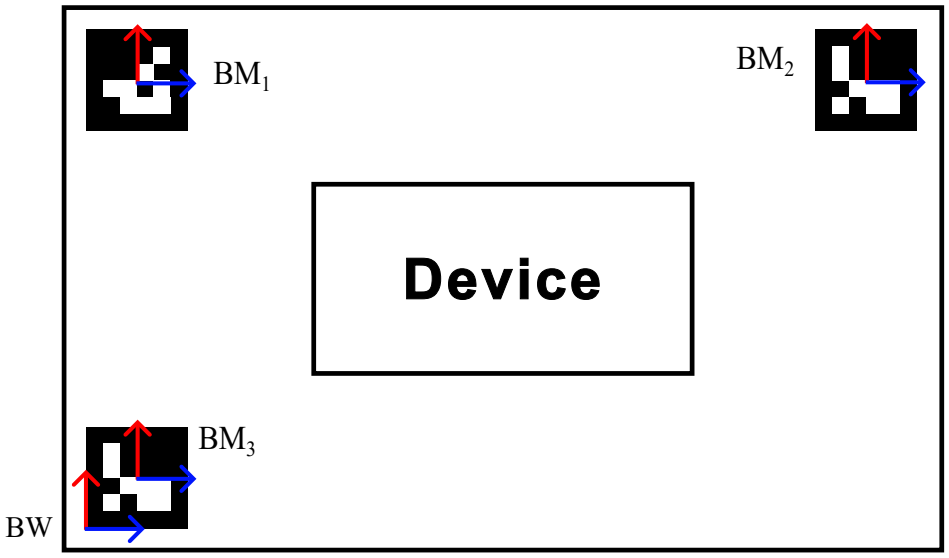

Figure 2.9. Diagram of the placement of aruco markers on the device.

To calculate the given task, we will use a modified method used in the article [20]. We will optimize the distance of the points specified in the coordinate system of the robot base. We get four corner points from each marker.

In OpenCV, the coordinate system of the marker is located in the middle of the marker. The coordinates of the corners of the marker in the coordinate system of the marker are $[\pm 0.5l; \pm 0.5l]$, where l is the length of the edge of the marker. At the same time, we know the coordinates of the edge of the marker in the world coordinate system. Using these two pieces of information and the known transformation matrices, we can convert these coordinates into the world coordinate system using the Equation (12) and Equation (13). Subsequently, we will define an optimization task to find the optimal value of the transformation from the robot's base to the coordinate system of the world.

$$h(\vec{x_{p_{ij}}}) = P_{BRF_{ij}} = T_{BW}^{BRF} P_{BW_{ij}} \tag{12}$$

$$h(\vec{x_{c_{ij}}}) = P_{BRF_{ij}} = T_{BG}^{BRF} (T_{BG}^{BC})^{-1} (T_{BC}^{BM})^{-1} P_{BM_{ij}} \tag{13}$$

$$\min \sum_{i=1}^{3} \sum_{j=1}^{4} \| \vec{x_{c_{ij}}} - \vec{x_{p_{ij}}} \|^2 \tag{14}$$

$P_{BW_{ij}}$ denotes the corner of marker i and the corner j specified in the world coordinate system. $P_{BM_{ij}}$ denotes the corner of marker i and corner j specified in the coordinate system of marker i. The $h()$ function adds 1 to the end of a 3D vector and thus creates a vector in 4D.

For calculating the optimal value, it is necessary to use the initial estimate of the transformation from the robot's base to the world's coordinate system. We will calculate the estimated transformation using the Equation (15). For this estimate, we will use one of the images used for the optimization calculation. We obtain the transformation

matrices from the robot base to the gripper, from the camera to the gripper, and from the marker to the camera in the same way as for the optimization calculation. The only missing transformation matrix is from the world coordinate system to the marker coordinate system. We know about this transformation that it is only a translation in the x-axis and y-axis. For the calculation, we will use the known coordinates of the center of the marker in world coordinates and insert them into the following matrix.

$$\begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{BW}^{BRF} = T_{BG}^{BRF}(T_{BG}^{BC})^{-1}(T_{BC}^{BM})^{-1}(T_{BM}^{BW})^{-1} \tag{15}$$

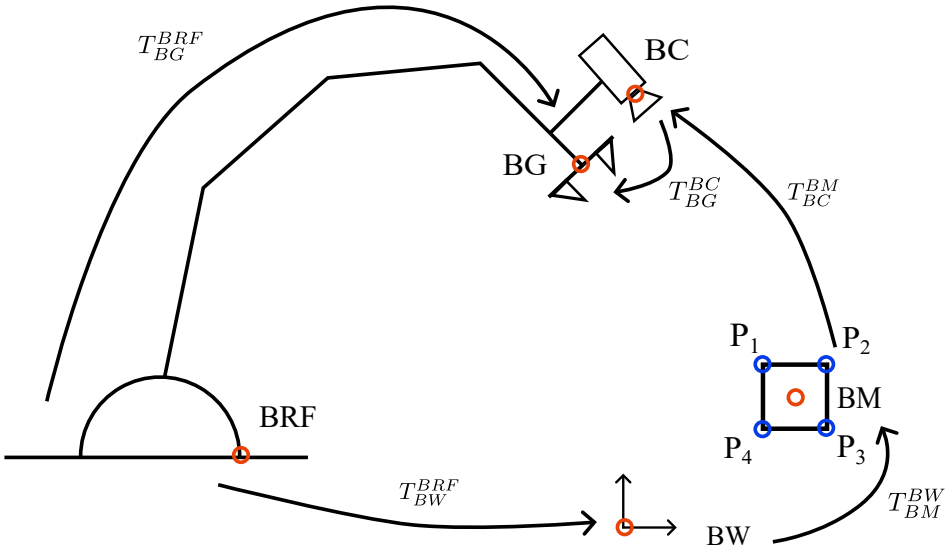

Figure 2.10. Schematic of the robot for calibration Robot-World.

Chapter 3
Implementation

The proposed solution must fulfill different required tasks at different times and with different frequency. Therefore, we divide the problem we are solving into three programs, each performing the appropriate actions at the necessary moment.

The first program is called the Internal robot calibration program, which is used to initialize the robot before deployment in production or after maintenance. It performs camera and hand-eye calibration using a ChArUco board calibration board. The camera calibration finds the camera's internal parameters, and hand calibration Finds the transformation from the camera coordinate system to the gripper coordinate system.

The second program, the External robot calibrations program, is responsible for calibrating the transformation from the base of the robot manipulator to the world coordinate system after moving to the work cell. This program contains an implemented robot-world calibration algorithm described in Subsection 2.5.3. ArUco markers placed on the work surface are used for position calibration.

The last program is called the Entry check program, which verifies if the work task for the current cell is correctly set for the central control system. This program uses the ArUco markers normally used for calibration in the External robot calibrations program and other ArUco markers located in the given cell. The entry check program takes information from these markers as input data and compares them to the expected values for that cell.

This Table 3.1 clearly describes the context in which it should be used correctly.

Program	When the program is used.
`Internal robot calibration program`	This program is used before deploying the robot in production or after performing camera maintenance.
`External robot calibrations program`	This program is used every time the robot is brought to the workplace.
`Entry check program`	This program is used every time a production program is entered.

Table 3.1. Table describing the use of individual programs.

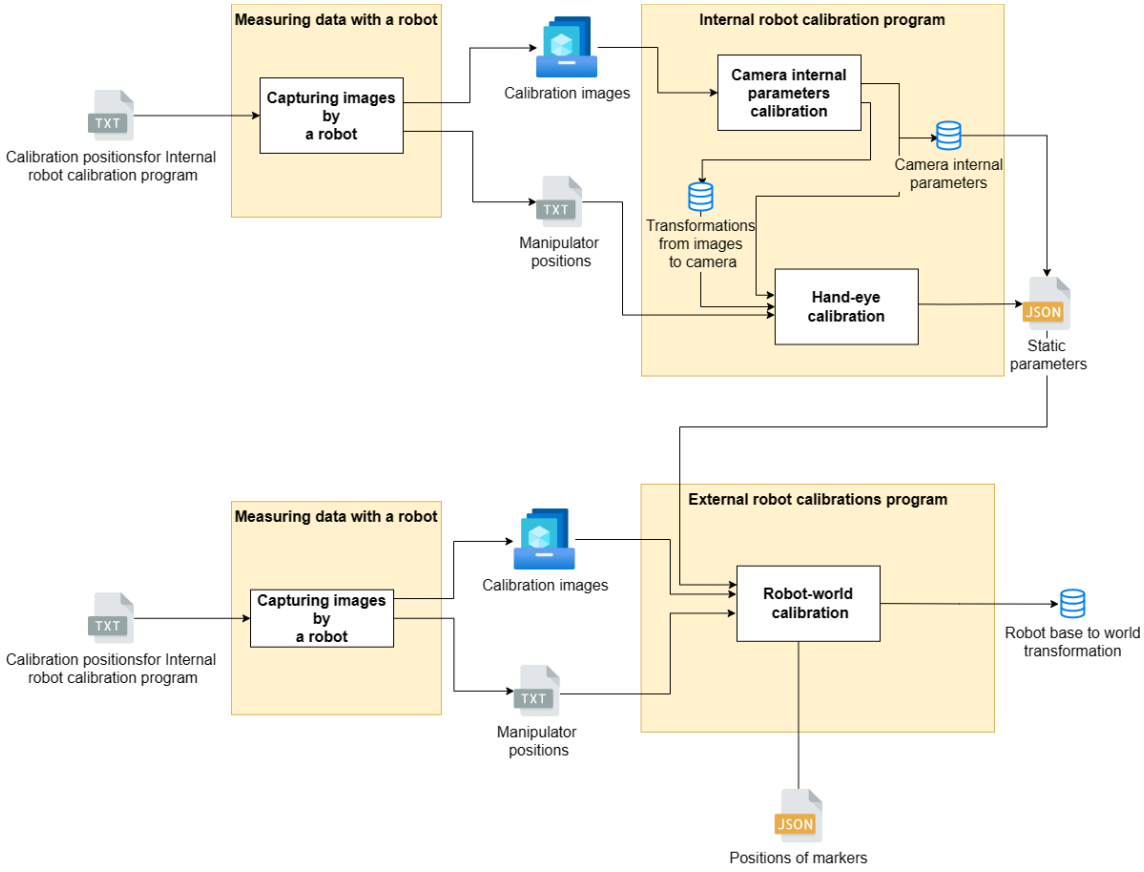

Figure 3.1. Schematic of inputs and outputs of individual programs.

3.1 Internal robot calibration program

Internal robot calibration program is responsible for camera calibration and hand-eye calibration. This module is used to initialize the parameters after the first start of the robotic manipulator or after maintenance is performed on the camera. We first need to place a calibration board in front of the robotic manipulator to use it.

Before starting the module, it is necessary to edit the file `constanc.py`. This file contains all the constants used in the following module. The constants that must be set before starting are those related to the selected calibration pattern and the file's name containing the calculated calibration data. The constants description is shown in the Table 3.2.

The module needs two input parameters to run. The first input parameter is path-to-images. The second input parameter is path-to-positions. In this parameter, we set the path to the TXT file, which contains the individual calibration positions obtained from the joint coordinates of the robot. These positions must be entered in the format, one position per line. Furthermore, the data is compared one after the other: $x\ y\ z\ Yaw\ Pitch\ Roll$. Positions are given in millimeters and angles in degrees, with only a space between each data.

Constant	Description	Default settings
CHARUCOBOARD_ROWCOUNT	Number of charucoboard rows.	14
CHARUCOBOARD_COLCOUNT	Number of charucoboard columns.	10
ARUCO_DICT	Dictionary used to generate ArUco markers.	DICT_4X4_1000
SQUARE_LENGTH	The size of the checkerboard square in meters.	0.025
MAKER_LENGTH	The size of the marker in meters.	0.015
NAME_STATIC_PARAMS	The name of the file to save the calibration results.	`static_params.json`

Table 3.2. Table containing default parameter settings and their description.

```
567.987517264901 0.0380750719599591 177.850935037031 -179.985603946761 79.6364998474734 -7.99038001528443
561.645121005374 40.1110374612002 177.844127991112 -162.017751591796 79.6393248819885 -11.0136747171203
543.193075007797 76.3499437044015 177.859918875963 -144.016272933373 79.6382089853997 97.9864000311224
514.46679633841 105.152428732159 177.867015215686 -126.000371587056 79.6422998712996 -44.001240224773
478.260957397527 123.595779804352 177.855607005465 -107.997010715275 79.639521233004 -52.9974362666184
438.113657236571 129.955786352302 177.876778739181 -90.0141505926892 79.6384377491283 -153.010451372343
397.956187791057 123.647175719875 177.857350935696 -72.0154405751661 79.6423543255504 -146.00993051155
361.719255571049 105.239292105624 177.898010096372 -53.9951153747909 79.6422535927328 -141.989848976156
361.357051488027 -104.980451800734 177.853084417398 54.0408182010247 79.6436493621113 115.035265650555
397.558919222748 -123.580720144252 177.914135626187 71.9660160720917 79.6408095860251 66.9704685405777
437.686490583638 -130.040140401394 177.91865158623 89.9814134794366 79.6438865594938 -26.0152208887572
477.870772759013 -123.783610670992 177.91957480626 107.958373998137 79.6447072301022 167.963400267733
514.121488491758 -105.414454193871 177.909248865504 125.987263655197 79.6412969242937 165.99134774442
```

Figure 3.2. Example of output data from the robot in a format to use for calibration.

The output of Internal robot calibration program is a JSON file containing the calculated calibrations. The data in this file is stored using a JSON object. This object is defined in the format {key: data}, similar to a dictionary in Python [21]. The output file contains four keys, thanks to which we can obtain camera calibration and hand-eye calibration data. The list of keys and data formats is described in the Table 3.3.

Key	Description	Specifications
`camera_mat`	Contains internal camera parameters in matrix format.	3×3 matrix K
`distortion_vec`	Contains external camera parameters in vector format.	vector with six constants $\{k_1, ..., k_6\}$
`rotation_mat`	The rotation matrix of the transformation from camera to grip.	3×3 matrix R
`translation_vec`	The translation vector from camera to grip.	3×1 vector T

Table 3.3. A table containing the keys and description of the data.

3.2 External robot calibrations program

This program expects four input arguments. The first argument is a file name of a JSON file containing the results of the `Internal robot calibration` program. The second argument is the filename of the calibration image file. The third parameter is the filename of a TXT file containing the positions of the manipulator in x y z Yaw $Pitch$ $Roll$ orientation, just like in the first program. The last parameter is a JSON dictionary containing the positions of the corners of individual markers in the world coordinate system, while the order is clockwise from the upper left corner. The marker ID is the key used to access individual markers.

For the correct functioning of the program, it is necessary to set the constants in the `constants.py` file. The constants are described in the Table 3.4. Other possible modifications in this module include changing the keywords for the file containing the camera calibration and the hand-eye calibration results. These parameters are preset for compatibility with the `internal_robot_calibration` program, and it is unnecessary to change them if the program uses the basic settings. All these parameters are listed in this Table 3.3.

Constant	Description	Default settings
ARUCO_DICT	Dictionary used to generate ArUco markers.	DICT_4X4_1000
NUM_OF_POSE	Number of markers used for calibration.	3
MAKER_LENGTH	The size of the marker in meters.	0.04

Table 3.4. Table containing default parameter settings and their description.

The return value is a NumPy matrix [22] containing the transformation from the robot base coordinate system to the world coordinate system. The program also includes implemented checks for user inputs, which are described in the Table 3.5.

Error	Description
1	Wrong number of images or wrong path.
2	Wrong path to static parameters.
3	Wrong size of marker.
4	Wrong path to markers positions.
5	Wrong path to data.

Table 3.5. The table contains the error number and description of the error.

3.3 Entry check program

This program is a supporting module for the operator input control application in manufacturing. Its job is to evaluate a camera image containing a tag, find that tag, and return its ID. The program is run with one parameter: the path to the image. The output of this function is the ArUco marker ID, or -1 if the marker was not found.

Chapter 4
Experimental results

This chapter focuses on verifying the functionality of programs and the accuracy of calibrating the robot base's position in the work cell. The measured results will be presented, and there will be discussions about the accuracy, potential causes, and ways to improve the outcomes.

4.1 Camera calibration result

Reprojection error is the most important parameter for determining the quality of camera calibration. This error tells us the difference in the object's position in the image, which we estimate based on the known size of the object compared to the position of the object in the actual image of the calibrated camera. Another critical parameter is error distribution in individual parts of the image and whether the error is more significant in just one direction or is distributed evenly over the entire image. We determine the reprojection error by calculating the root mean square error using the Equation (1). p_p is the position of the point calculated from the model, p_c is the position found in the image, and n is the number of points in our case equal to four. To evaluate the error distribution, we write the error value for a specific marker in the calibration image. We draw the direction of the error as an arrow pointing from the place where we expected the point to the place where the point is in the image. For better visibility, we enlarged the arrow 400 times [23].

$$e = \sqrt{\frac{\sum_{j=1}^{n} \|p_p - p_c\|^2}{n}} \tag{1}$$

We created function `test-camera-calib()` to verify the calibration. The inputs to this function are images intended for calibration and images intended for evaluation. First, we calculate the camera parameters using the `inner-param-calib()` function. This function is described in the calibration chapter. This calibration will give us the camera parameters. Those parameters will be tested for accuracy. Subsequently, we find ArUco markers on the validation images using the `detectMarkers()` function. This function gives us their position in the image. Subsequently, a marker-to-camera transformation is calculated for each marker using the `estimatePoseSingleMarkers()` function. Thanks to the known dimensions of the ArUco markers and the location of the coordinate system, we can calculate where these marks would be in the image if the camera had the parameters we managed to calculate. We perform this calculation using the `projectPoints()` function. Since we know the location of the points in the real image and the points in the camera image that we calculated, we can calculate the reprojection error.

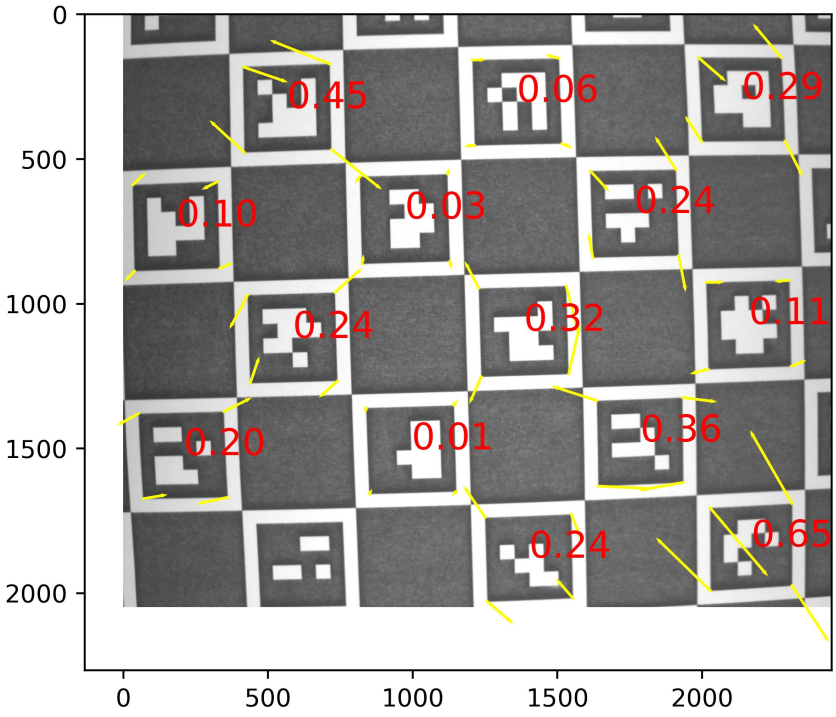

Figure 4.1. Image with reproduction error value and error directions.

4.1.1 Experiment

For testing, we take eight images in four different tilts of the manipulator, two images each. The angle between the camera and the calibration board for image acquisition is 0.6 rad, 0.8 rad, 1.39 rad, and 1.222 rad. Furthermore, we take the first image with an x-axis tilt and the second with a y-axis tilt. We evaluated images with the `test-camera-calib()` function. The maximum value, minimum value, median, and mean can be found in Table 4.1. It can be seen from the table that the maximum error value is not more than 1 pixel. From the averages, we can calculate the total average value of the reprojection error of the measured data, which corresponds to 0.32 pix.

Image number	Max RMS	Min RMS	Median RMS	Mean RMS
1.	0.42	0.11	0.21	0.23
2.	0.98	0.12	0.34	0.44
3.	0.72	0.12	0.29	0.34
4.	0.67	0.08	0.36	0.35
5.	0.80	0.16	0.41	0.40
6.	0.48	0.13	0.24	0.29
7.	0.55	0.08	0.27	0.30
8.	0.74	0.06	0.25	0.23

Table 4.1. Table of RMS values from individual images. All values are in pixels.

4.2 Hand-eye validation

In this section, we will focus on verifying the accuracy of the hand-eye calibration results by comparing them to the expected values calculated using DH notation.

4.2.1 Estimation of hand-eye transformation

We estimate the hand-eye transformation value to compare the results of the hand-eye transformation value computed by the program. We will use Denavit–Hartenberg notation to calculate the estimate. First, we must determine the gripper's coordinate system orientations and camera coordinate systems orientation. We obtain these coordinate systems from the following Figure 4.2 and Figure 2.7. We will measure translations on individual axes on a real robot. Thanks to these data, we can calculate the parameters of the two DH notations listed in the Table 4.2. Value marked with ? is the optical center position that cannot be measured with a ruler and is therefore marked as unknown in this value estimate. Finally, we calculate the resulting hand-eye transformation Matrix (2).

θ	z	x	α
$0°$	0.045	?	$-90°$
$-90°$	0.015	0	$-90°$

Table 4.2. Table containing parameters of DH notations.

$$\begin{bmatrix} 0 & 0 & 1 & ? \\ 0 & -1 & 0 & 0.015 \\ 1 & 0 & 0 & 0.045 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

4.2.2 Experiment

As part of the experiment, images were captured. Subsequently, a computation of transformation from the camera coordinate system to the gripper coordinate system was performed using the function `internal_robot_calibration()`. The result of this transformation is presented in the following Table (3):

$$\begin{bmatrix} 0.1028 & 0.1571 & 0.9822 & 0.0251 \\ 0.1918 & -0.9720 & 0.1353 & 0.0161 \\ 0.9760 & 0.1744 & -0.1301 & 0.0448 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Comparing the results, we found that the transformation estimate corresponds to the calculated transformation. At the same time, we observed that the unknown value falls within the possible interval of values.

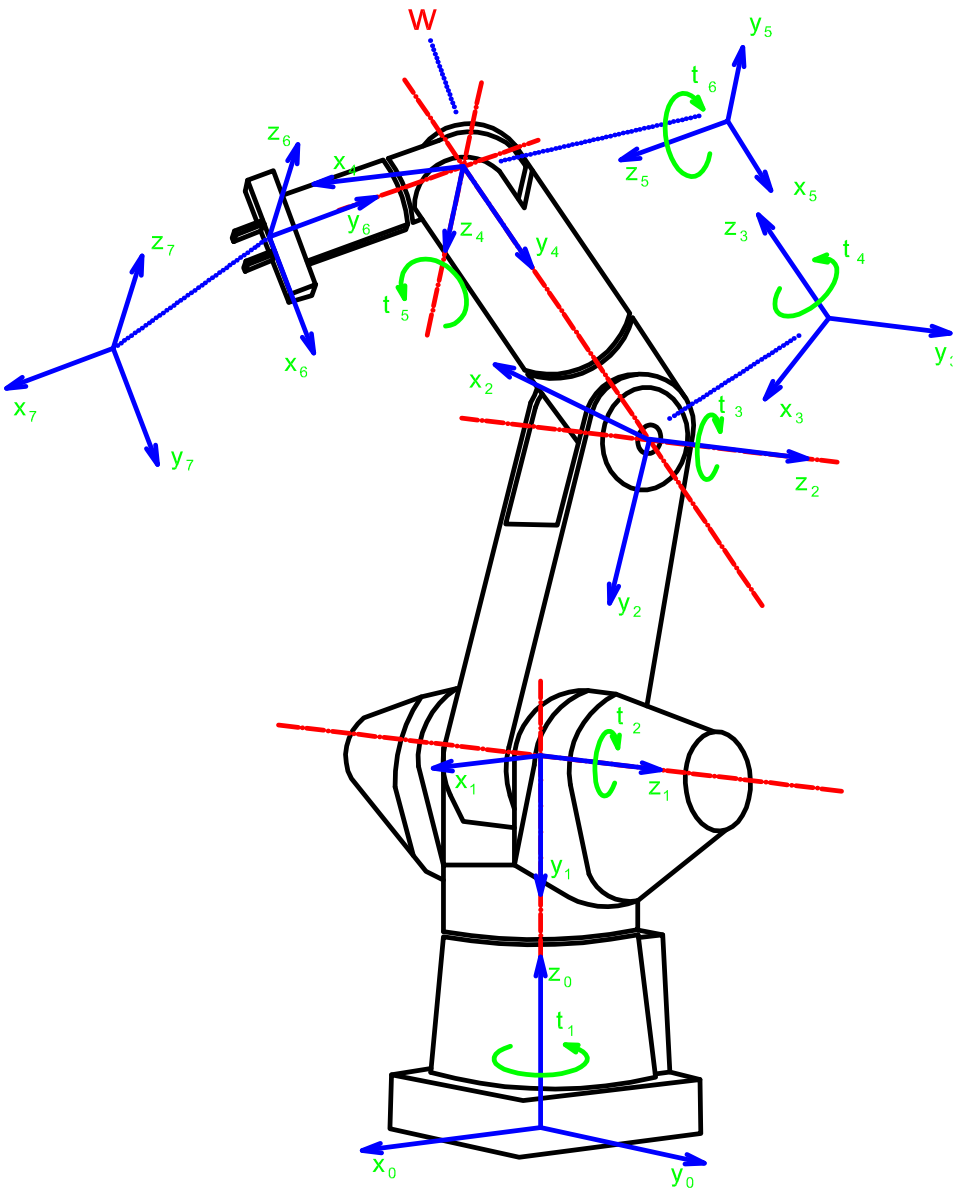

Figure 4.2. Robotic manipulator A465 with marked joint coordinate systems [24].

4.3 Robot-world result

In this section, we will deal with the execution of the robot-world calibration experiment and evaluate the obtained results. First, the transformation estimation method for checking the results will be described, followed by the experiment and its results.

4.3.1 Estimation of robot-world transformation

We estimate the value of this transformation to compare the obtained results of the robot-world transformation. We will use Denavit–Hartenberg notation to calculate

the estimate. For this calculation, we need to find out the orientation of the world coordinate system. We get it from the diagram Figure 4.3, where the coordinate system is marked in the lower left of the diagram. The red arrow indicates the orientation of the x-axis, the green arrow the orientation of the y-axis, and the z-axis extends outward from the board. Next, we need to get the orientation of the coordinate system of the robot base, which is marked in the Figure 4.2. Subsequently, we will measure individual translations in the real world. Thanks to these data, we can calculate the parameters of the two DH notations, which are entered in this Table 4.3.

θ	z	x	α
$0°$	-0.05	0.35	$0°$
$-90°$	0	-0.15	$0°$

Table 4.3. Table containing parameters of DH notations.

From the parameters of the DH notation, we will construct the corresponding matrices. By multiplying these matrices, we get the Matrix (4), which estimates the value that the External robot calibrations program should calculate for us. We will then measure the data to calculate the robot-world calibration and use the External robot calibrations program to obtain the transformation Matrix (5).

$$\begin{bmatrix} 0 & 1 & 0 & 0.34 \\ -1 & 0 & 0 & 0.16 \\ 0 & 0 & 1 & -0.05 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} -0.0629 & 0.9980 & -0.0041 & 0.3444 \\ -0.9979 & -0.0628 & 0.0065 & 0.1809 \\ 0.0062 & 0.0045 & 0.9999 & -0.0088 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

By comparing these two matrices, we see that the value obtained by the calibration program is real. The most significant error is in the z-axis, where it is not very clear from the picture where exactly this coordinate system is located. We will subsequently use this matrix calculated by the program for testing.

4.3.2 Experiment

The following experiment was performed to evaluate the robot-world calibration. Four points on the test board were selected, and their position in the world coordinate system was measured. The position of these points can be seen in the Figure 4.3. Subsequently, we multiply these points with the transformation from the robot base coordinate system to the world coordinate system according to the Equation (6), where i denotes the i-th measured position. This way, we get those points in the coordinate system of the robot base. We then let the robotic manipulator grasp the box and guide it to the calculated coordinates. Now we measure the distance from the center of the box figure to the point we determined and measure the error in the individual axes. Absolute errors in individual axes and individual points are listed in the Table 4.4.

$$P_{BRF_i} = T_{BW}^{BRF} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{6}$$

Measurement position	X-axis error	Y-axis error	Z-axis error
—	mm	mm	mm
1.	6.5	8.5	7
2.	5.5	9.5	6
3.	7.5	11.5	8
4.	3.5	6.5	6

Table 4.4. Table of errors in absolute value in individual axes of robot-world calibration.

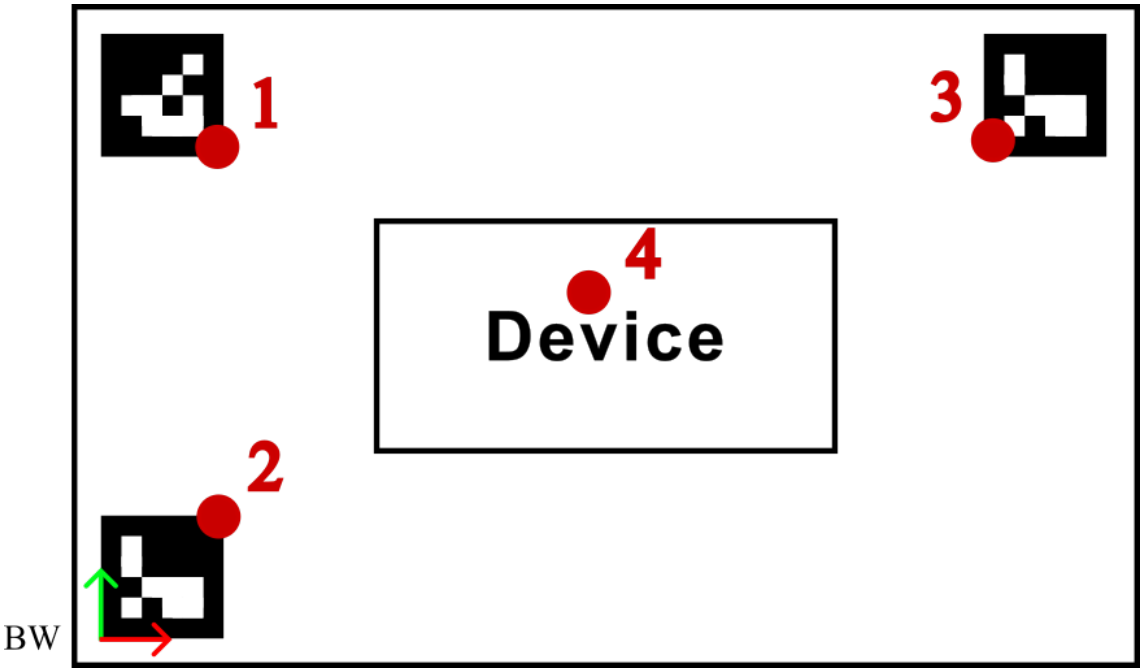

Figure 4.3. Diagram of test positions on the device.

4.3.3 Evaluation of results

The results show that the calibration process leads to meaningful results. At the same time, it can be seen that the results are better in the middle of the workspace than at the edges of the workspace. Unfortunately, the obtained calibration results do not meet expectations, and there is still room for improvement in terms of accuracy.

I think that the error in the calibration is caused by the conditions under which the experiment was carried out. This error could be influenced by two factors: the markers' quality and factors associated with the manipulator. The main factors affecting the manipulator's accuracy are the manipulator's age, the handling of the given manipulator, and the loss of calibration when using a different control unit. The second major factor is the markers used, which were printed on the paper. A more accurate manipulator and better manufactured marker quality should improve calibration accuracy.

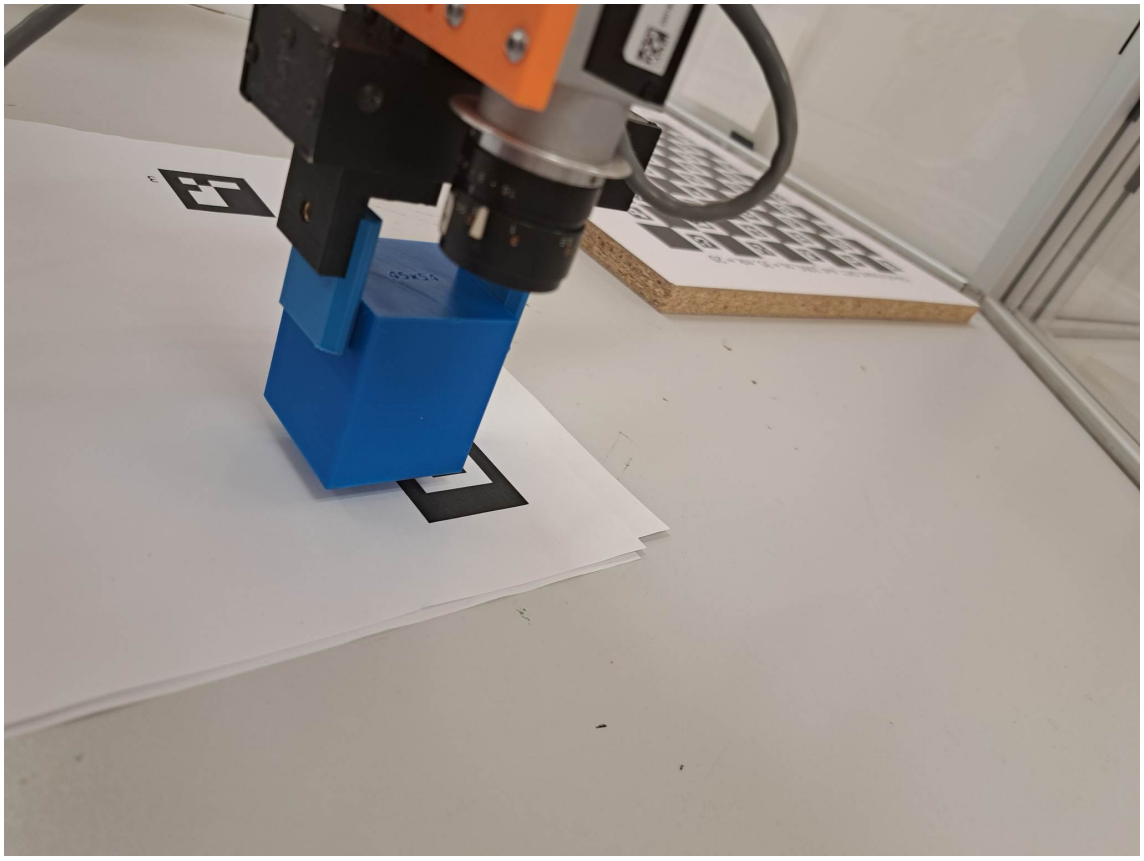

Figure 4.4. A robotic manipulator placing a cube on a test site to check the accuracy.

4.4 Entry check program validation

The Entry check program is used to verify that the production operator entered the production task correctly. In order to verify its functionality, similar conditions were created, like during regular operation in production plants. These tests aimed to determine whether the program works correctly and can be used in industrial production.

4.4.1 Experiment

To verify the functionality of the program. Scanning was performed with a robotic manipulator, during which 40 images were captured. These images consisted of 4 types, with each type containing ten images. Specifically, these were images with ArUco ID 1, ArUco ID 2, ArUco ID 3, and images containing other objects for control measurement purposes. Subsequently, these images were gradually marked by a program that determined the identification of ArUco markers or the absence of these markers in the images. The following images are an example of marking Figure 4.5 and Figure 4.6. Of the 40 marked images, all were evaluated correctly, which leads us to conclude that this function is suitable for checking the correct program entry by the production operator.

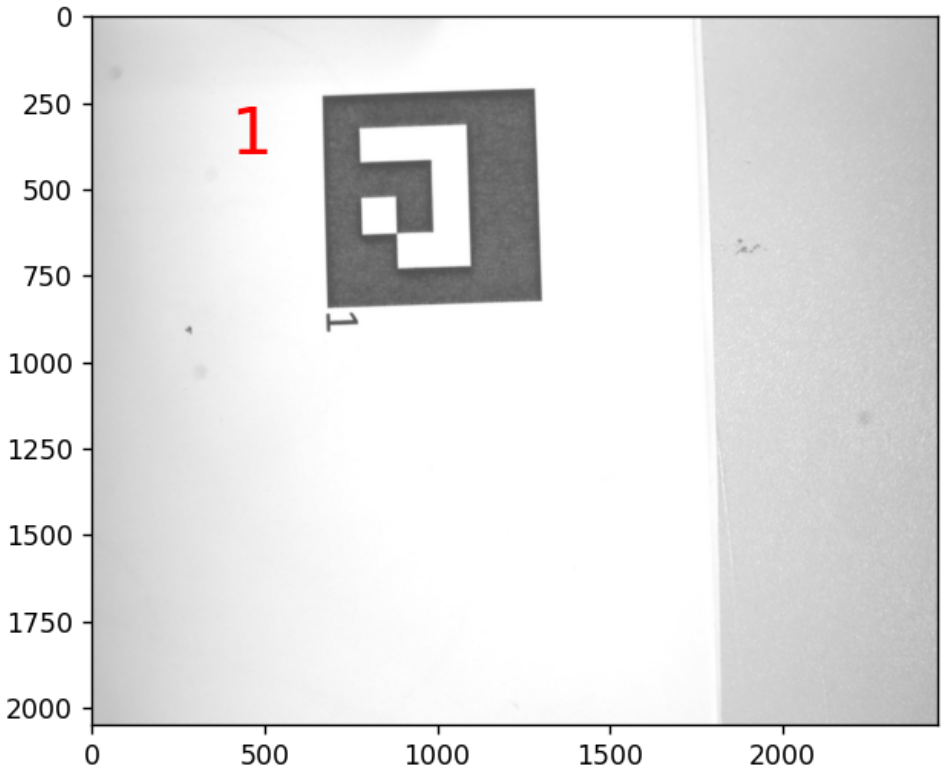

Figure 4.5. Image with detected marker with ID 1.

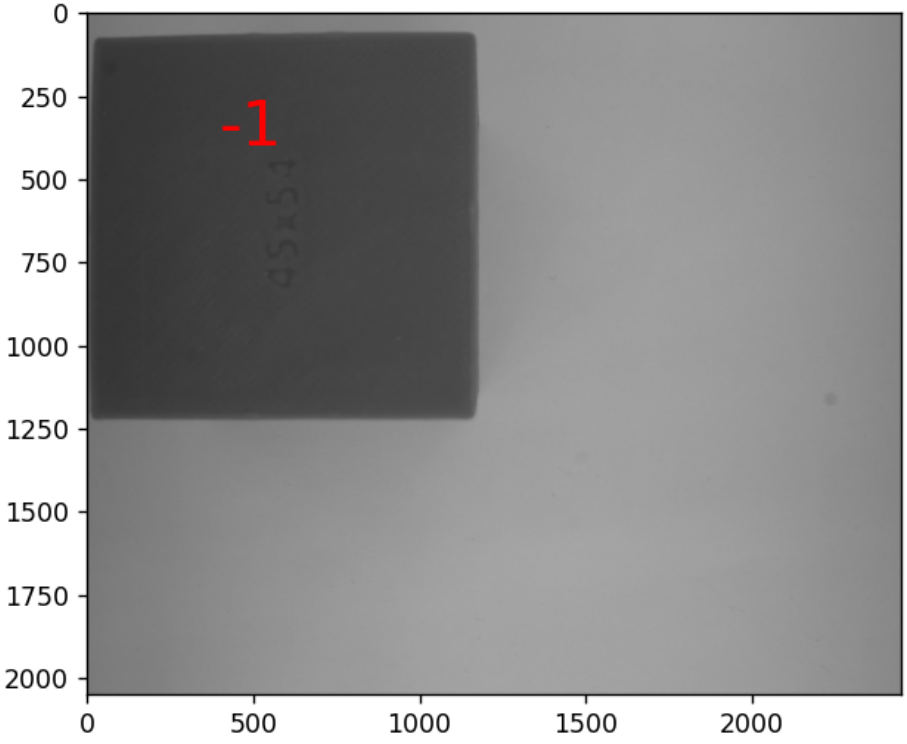

Figure 4.6. Image with no detected marker.

Chapter 5
Conclusions

As part of my bachelor's thesis, I learned about the production cell and the UR10e robotic manipulator in the Rohde & Schwarz production plant. The main goal of my work was to propose an automatic calibration of the transformation from an industrial manipulator base to a world coordinate system after moving to a robotic workplace.

I conducted thorough research on the available calibration markers and analyzed different solutions that correspond to the customer's specific needs. During the research, I learned in detail the algorithms used to calibrate the position of the industrial manipulator in the production environment. I primarily focused on robot-world, hand-eye, and camera calibration methods. These methods are tools for correctly finding the manipulator's position in the work cell and achieving the required accuracy. Based on the results of my research, I evaluated the most suitable solution that meets the customer's specific requirements. I have implemented this proposed method and added a program correctness check function at Rohde & Schwarz's request.

In order to verify the functionality, reliability, and ability to effectively manage the required tasks in a real operating environment, I tested the proposed solution on the A465 industrial manipulator from CRS. This testing confirmed that the proposed solution is able to work successfully in practical situations and achieve the desired results. Testing also made it possible to identify shortcomings for future improvements and to achieve optimal performance of the manipulator in real operation.

5.1 Possible improvements

This bachelor's thesis focuses on creating programs for the automatic calibration of industrial manipulators. The main goal of the work was to create these programs and test their functionality. For future use, there are opportunities to improve the results and operation of the programs. The main possibilities for improvement include using better calibration targets and loading the positions of individual calibration markers from the central database of the manufacturing plant for easy data maintenance and modification.

5.1.1 Calibration targets improvements

In this bachelor's thesis, we only use markers printed on office paper with a laser printer. To increase accuracy, it is better to use markers made by other processes. Here are some production method options:

- Markers printed on foil glued to a solid flat surface and foil overlaid to reduce light reflections into the camera.
- Markers are burned by laser into a metal or plastic plate.

References

[1] *Rohde & Schwarz Vimperk*. 2023.
https://www.rohde-schwarz.com/cz/about-czech-republic/plants/vimperk/o-nas/prehled/o-nas_253302.html.

[2] *OpenCV documentation*.
https://docs.opencv.org/4.x/.

[3] Pavel Píša. *Uživatelský manuál k jednotce MARS 8*.
https://cmp.felk.cvut.cz/~pisa/mars8/mars8_man_cz.pdf.

[4] *A465 Robot*.
http://cmp.felk.cvut.cz/cmp/courses/ROB/labsmaterial/CRS/a465.pdf.

[5] *Robot CRS v předklonu*.
http://cmp.felk.cvut.cz/cmp/courses/ROB/labsmaterial/CRS/CRSpredklon.png.

[6] *THE UR10e*.
https://www.universal-robots.com/products/ur10-robot/.

[7] *UR10 e*.
https://cowelder.com/wp-content/uploads/2021/12/UR10e1.png.

[8] *DaA3840-45um*.
https://www.baslerweb.com/en/products/cameras/area-scan-cameras/dart/daa3840-45um-no-mount/#accessories.

[9] *Basler Lens C125-0618-5M-P f6mm*.
https://www.baslerweb.com/en/products/lenses/fixed-focal-lenses/basler-lens-c125-0618-5m-p-f6mm/.

[10] Michail Kalaitzakis, Brennan Cain, Sabrina Carroll, Anand Ambrosi, Camden Whitehead, and Nikolaos Vitzilaios. Fiducial Markers for Pose Estimation. 2021, 101 (4), DOI 10.1007/s10846-020-01307-9.

[11] *Example of markers images*.
https://docs.opencv.org/4.x/markers.jpg.

[12] Ikenna Enebuse, Mathias Foo, Babul Salam Ksm Kader Ibrahim, Hafiz Ahmed, Fhon Supmak, and Odongo Steven Eyobu. A Comparative Review of Hand-Eye Calibration Techniques for Vision Guided Robots. *IEEE Access*. 2021, 9 113143-113155. DOI 10.1109/ACCESS.2021.3104514.

[13] *ChArUco board*. 2023.
https://calib.io/pages/camera-calibration-pattern-generator?gad=1&gclid=CjwKCAjw36GjBhAkEiwAKwIWyWayh0F5lrbA4D4aP5qE64rsoUn4ntwxAl_9mwag6_XLZHSrI94yOxoCfRkQAvD_BwE.

[14] *Camera Calibration and 3D Reconstruction*.
https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html.

[15] *HALCON/HDevelop*.
https://www.mvtec.com/fileadmin/Redaktion/mvtec.com/products/halcon/documentation/reference/reference_hdevelop.pdf.

[16] Jan Heller. *Global Optimization Techniques in Camera-Robot Calibratio*. 2015.

[17] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 22 (11), 1330-1334. DOI 10.1109/34.888718.

[18] *Pinhole camera model*.
https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html.

[19] Konstantinos Daniilidis. Hand-Eye Calibration Using Dual Quaternions. *The International Journal of Robotics Research*. 1999, 18 (3), 286-298. DOI 10.1177/02783649922066213.

[20] Yang Li, Junlei Hu, Baoxin Tao, Dedong Yu, Yihan Shen, Shengchi Fan, Yiqun Wu, and Xiaojun Chen. Automatic robot-world calibration in an optical-navigated surgical robot system and its application for oral implant placement. *International Journal of Computer Assisted Radiology and Surgery*. 2020, 15 (10), 1685-1692. DOI 10.1007/s11548-020-02232-w.

[21] *Introducing JSON*.
https://www.json.org/json-en.html.

[22] *NumPy Documentation*. 2022.
https://numpy.org/doc/.

[23] *RMS*.
https://calib.io/blogs/knowledge-base/understanding-reprojection-errors.

[24] *Robot A465*.
http://cmp.felk.cvut.cz/cmp/courses/ROB/labsmaterial/CRS/robot_CRS_s_osama_2.pdf.

[25] *Sci-py documentation*. 2023.
https://docs.scipy.org/doc/scipy/.

[26] *Glob - Unix style pathname pattern expansion*. 2001-2023.
https://docs.python.org/3/library/glob.html.

[27] *Json - JSON encoder and decoder*. 2001-2023.
https://docs.python.org/3/library/json.html.

[28] *Math - Mathematical functions*. 2001-2023.
https://docs.python.org/3/library/math.html.

Appendix A
Used libraries

According to the addendum in the contract, I state a list of libraries.

- OpenCV [2]
- NumPy [22]
- SciPy [25]
- glob [26]
- json [27]
- math [28]