Bachelor's Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Autonomous Eye-In-Hand Pick & Place System for an Industrial Robot

Martin Mikšík

Supervisor: Ing. Pavel Burget, Ph.D.
Field of study: Cybernetics and Robotics
May 2023

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Mikšík Martin** Personal ID number: **492358**

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

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Autonomous eye-in-hand pick & place system for an industrial robot

Bachelor’s thesis title in Czech:

Autonomní systém sbírání součástek pomocí kamery na rameni průmyslového robota

Guidelines:

The purpose of this thesis is to design and implement a robotic workplace equipped with a camera to pick parts from an external shelf and put them onto a conveyor shuttle. The position of the shelf is not strictly given and thus callibration of the robot base must be performed using the camera. Furthermore, an autonomous recognition of the parts in the shelf will be performed whereas the parts will be represented by various electronic boards. The robot will be equipped with a standard interface based on OPC UA to interact with the rest of the assembly line the robot is part of.

1. Get acquainted with the callibration methods of a robot baseframe using camera.
2. Design and implement a setup with a camera mounted at a robot flange to callibrate the robot baseframe with respect to a shelf with parts.
3. Design and implement an algorithm to pick electronic boards placed randomly in the shelf and to place them on a defined position on a conveyor shuttle.
4. Design a data model for OPC UA and implement it in the robot to integrate the robot to the assembly line and to allow it to interact with an above laying manufacturing execution system.

Bibliography / sources:

[1] Wang Qi, Fu Li, and Liu Zhenzhong, Review on camera calibration, 06/2010, pp. 3354 – 3358. DOI: 10.1109/CCDC.2010.5498574
[2] Richard Szeliski, Computer vision algorithms and applications, 2011. Springer. DOI: 10.1007/978-1-84882-935-0

Name and workplace of bachelor’s thesis supervisor:

Ing. Pavel Burget, Ph.D. Testbed CIIRC

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

Date of bachelor’s thesis assignment: **30.01.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until:
by the end of summer semester 2023/2024

Ing. Pavel Burget, Ph.D.	prof. Ing. Michael Šebek, DrSc.	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

Acknowledgements

I could not have completed this work without the unwavering support and guidance of my supervisor and director of Testbed for Industry 4.0, Ing. Pavel Burget, Ph.D., thank You.

A debt of gratitude is also owed to the colleagues behind the Flexible assembly line project and researchers at Czech Institute of Informatics, Robotics, and Cybernetics. Thank you for believing in me as head of the project.

I would also like to give special thanks to my family as a whole for their continuous support and understanding when undertaking my studies and research. Thank you.

Declaration

I declare that the presented work was developed independently, and that I have listed all the sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of the university thesis.

In Prague, May 26, 2023

Martin Mikšík

Abstract

Although current flexible assembly lines offer improved adaptability in manufacturing, the need for time-consuming reprogramming of manipulator target poses is needed to accommodate new assembly operations. This presents significant logistical and financial challenges, as it often requires stopping the entire manufacturing process, but also poses safety hazards as humans teach new robot positions within the robot's workspace.

In this study, a method deploying automatic intrinsic and extrinsic calibration, autonomous workspace scanning and assembly parts picking by an Eye-in-Hand approach is introduced, eliminating the need for physical human intervention in a flexible assembly line innovation process, such as a new product to assemble, and removing the requirement for predefined coordinates of the to-be-picked parts, further simplifying assembly lines pre-processing methodologies. To showcase the versatility and applicability, the method was implemented and tested on a concurrently running experimental flexible assembly line while developing a new product for assembly.
To demonstrate the obsolescence of predefined coordinate part placements, autonomous warehouse vehicles were utilized to deliver new parts to the robot in a randomized order, placement and quantity. Our experiments revealed high accuracy in autonomous scanning and picking operations, even when dealing with parts of various shapes and sizes. Additionally, our method utilizes controller-based Flange-Base transformations, therefore allowing the method to be applied to a wide variety of robots that are capable of utilizing computer vision throughout their entire workspace. Moreover, the camera mounted on the robot's arm provides constant visual feedback to the operator, as well as operation status for a digital twin feedback.

The experiments demonstrated precision within a millimeter range and highlighted the capability of our technique to advance the field of industrial autonomous robotics in assembly operations.

Keywords: Camera calibration, Eye-in-Hand, coordinate system transformation, homogeneous transformations, industrial flexible assembly line, OpenCV, Profinet, OPC UA, KUKA KRL, CIIRC

Supervisor: Ing. Pavel Burget, Ph.D.

Abstrakt

Ačkoli současné flexibilní montážní linky nabízejí lepší přizpůsobivost ve výrobě, stále přetrvává nutnost časově náročného přeprogramování cílových poloh robotických manipulátorů pro možnost přizpůsobení novým montážním operacím. To představuje značné logistické a finanční problémy, jelikož často vyžadují zastavení celého výrobního procesu, také ale představují bezpečnostní riziko, neboť lidé učí nové pozice robota v jeho pracovním prostoru.

V této bakalářské práci představujeme metodu zavádějící automatickou intrinsickou a extrinsickou kalibraci, autonomní skenování pracovního prostoru a sbírání montážních dílů (Bin picking) pomocí kamerového systému založeném na principu Eye-in-Hand. Tímto přístupem se eliminuje potřeba fyzického zásahu člověka v procesu inovace flexibilní montážní linky a odstraňuje požadavky na předem definované souřadnice vybíraných dílů, což dále zjednodušuje metodiky předzpracování montážních linek. Abychom ukázali univerzálnost a použitelnost, implementovali a otestovali jsme naši metodu na současně běžící experimentální flexibilní montážní lince při vývoji nového výrobku pro montáž.

Abychom demonstrovali nadbytečnost předem definovaných souřadnicových rozmístění dílů, využili jsme autonomní skladová vozidla (AGV), která robotovi dodávají nové díly v náhodném pořadí, rozmístění a množství. Naše experimenty odhalily vysokou přesnost autonomního skenování a sbírání součástek, a to i při práci s díly různých tvarů a velikostí. Naše metoda navíc využívá Flange-Base transformace založené na řídicí jednotce daného robota, čímž je umožněno použití metody na široké škále systémů, které jsou tak schopné využívat počítačové vidění v celém svém pracovním prostoru. Kamera umístěná na rameni robota může navíc poskytovat neustálou vizuální zpětnou vazbu operátorovi a provozní stav pro digitální dvojče.

Experimenty prokázaly přesnost v milimetrovém rozsahu a zdůraznily schopnost této metodologie rozvinout oblast průmyslové autonomní robotiky v montážních operacích.

Klíčová slova: Kalibrace kamery, Eye-in-Hand, transformace souřadnicových systémů, homogenní transformace, průmyslová flexibilní výrobní linka, OpenCV, Profinet, OPC UA, KUKA KRL, CIIRC

Překlad názvu: Autonomní Systém Sbírání Součástek Pomocí Kamery Na Rameni Průmyslového Robota

Contents

Figures

Tables

Chapter 1

Introduction

Assembly lines have been a staple of industrial manufacturing for over a century, streamlining the production process and increasing efficiency. However, traditional assembly lines are rigid and inflexible, making it challenging to adapt to changing market demands and product developments. In today's rapidly advancing technological landscape, it is increasingly critical for European industries to stay competitive by implementing more flexible and adaptable production methods. Here, the concept of a flexible assembly line comes in, offering a new approach to industrial manufacturing that allows greater adaptability and efficiency.

The Czech Institute of Informatics, Robotics, and Cybernetics (CIIRC) is currently conducting research on the development of an experimental flexible assembly line (later referred to as the Flexible assembly line) at its Testbed for Industry 4.0 facility. This project aims to incorporate cutting-edge technologies such as robotics, manufacturing execution systems, multi-agent planning, and autonomous guided vehicle (AGV) fleet cooperation to enhance the efficiency and adaptability of the Flexible assembly line operations. The research is being carried out in collaboration with prominent industry partners to ensure its practical relevance and applicability in real-world scenarios.

The current state-of-the-art in flexible assembly lines allows for a dynamic reconfiguration of controllers and an implementation of adaptable communication protocols to accommodate changes in factory infrastructure [ABV20]. However, it is limited by the lack of computer vision applications and still relies on the constraint that inbound assembly parts must be placed in predetermined and unchangeable coordinates, which can be time-consuming to program. Additionally, the slightest movement of components can cause a collision in the production system, resulting in costly downtime and repairs.

The focus of this thesis (later referred to as the thesis, the work, the method, or the project) is to explore the implementation of a computer vision Eye-in-Hand approach for the Flexible assembly line.
This strategy will give a robotic arm the ability of visual perception in its entire workspace and can result in improved manufacturing adaptability and cost effectiveness, as inbound parts can be placed relatively anywhere in the workspace, and the localization of such parts can be scanned and calibrated in real-time. This technology can be seen as giving "new eyes and brain" to a robot, allowing it to perceive and understand its surroundings, and is representing an essential advancement in the field, opening up possibilities for future research and development.

Figure 1.1: Frans Timmermans at the Flexible Assembly line project in Testbed for Industry 4.0 facility. Picture by VIC [VIC22]

1.1 Introduction to the Broader Context

In addition to the focus on implementing a computer vision Eye-in-Hand approach for the Flexible assembly line, several paragraphs in this work are dedicated to the development of the project's sub-components to ensure a comprehensive understanding for the reader. These include the modeling, testing, and deployment of a new pneumatic gripper and its control unit, an automatic gripper changer for all robots on the assembly line, remote access to the Eye-in-Hand camera and validation system, visualization of the robot's data feedback, and button-driven control for manual operations.

As the Team Leader of the Flexible assembly line, the main priority is to develop solutions that will not only operationalize the methodology described in this work but also facilitate new solutions for Testbed for Industry 4.0. This involves coordinating development across teams, such as in the creation of a complex 3D model of a fully driveable RC car and all the components needed for its robotic assembly on the Flexible assembly line and adjacent stations. The scope of the ongoing processes then ranges from coordinating the development of tool exchanger identification PCBs, the creation and mounting of part holders and workspace tables, to innovating the process of teaching robots' workspaces with a new calibration probe, as well as coordination of the implementation of AGV mission control and Manufacturing Execution System (MES) behavior, which will overtake the overall assembly lines' planning. These coordinations are essential for a coherent deployment of the project, as well as the continuous improvement of the Flexible assembly line.

Therefore, this work will also explore the development of a new product that can be built on the Flexible assembly line, an RC car consisting of a motor, wheels, several PCBs, and chassis with integrated differential and suspension. These components (Also further referred to as "the parts") will be later autonomously picked for assembly. The primary focus of this work is on the design of the autonomous Pick & Place system, which will be deployed and coordinated with partners of the Czech-German Research and Innovation Centre on Advanced Industrial Production (RICAIP) to demonstrate the adaptability of the line and cooperation of European innovation centers. The goal of this thesis is to make a meaningful and impactful contribution.

It is also important to emphasize that many components of this work had to be developed through iterative processes of testing and refining, which required a significant amount of research, trial and error, and the abandonment of certain ideas and implementations. Therefore, not all of the tried hardware and software solutions are explicitly described in the thesis.

Chapter 2

The Flexible assembly line: An Overview and Development of a New Product

As stated in [MB23], compared to traditional assembly lines, which are typically designed for a high-volume production of a single product, flexible assembly lines offer several advantages. They are more adaptable to changing market demands, allowing manufacturers to respond quickly to shifts in consumer preferences without having to tear down the previous state of the assembly line. A comprehensive review of the research work done in the area of flexible robotic assembly control systems was presented in [CFH92] and the validation of the operational flexibility of robotic smart systems in Industry 4.0 was summarized in [SAHM22].
In this study, project development was carried out on the experimental Flexible assembly line (see Fig. 2.1) which employs a multitude of industrial and collaborative robots, an in-house, custom-built manufacturing execution system (MES) as described in [NDKV22], [WVN+19b], [WVN+21], [JKO18], [WVN+19a], including implementation of LIDAR workspace safety, monorail shuttle distribution, multi-agent execution planner, and inbound AGV communication [Dou22], [NVW20].

KUKA Cybertech KR8 R1620 (subsequently referred to as Cybertech, the robot, or R20) was utilized for the project development (see Fig. 2.2), playing a vital role in the assembly process. The robot is responsible for distributing incoming parts from the warehouse or preceding assembly line to the monorail stations for subsequent assembly operations.

The versatility of the Flexible assembly line lies in its ability to adapt to unexpected situations, such as a malfunction of one of the robots, by

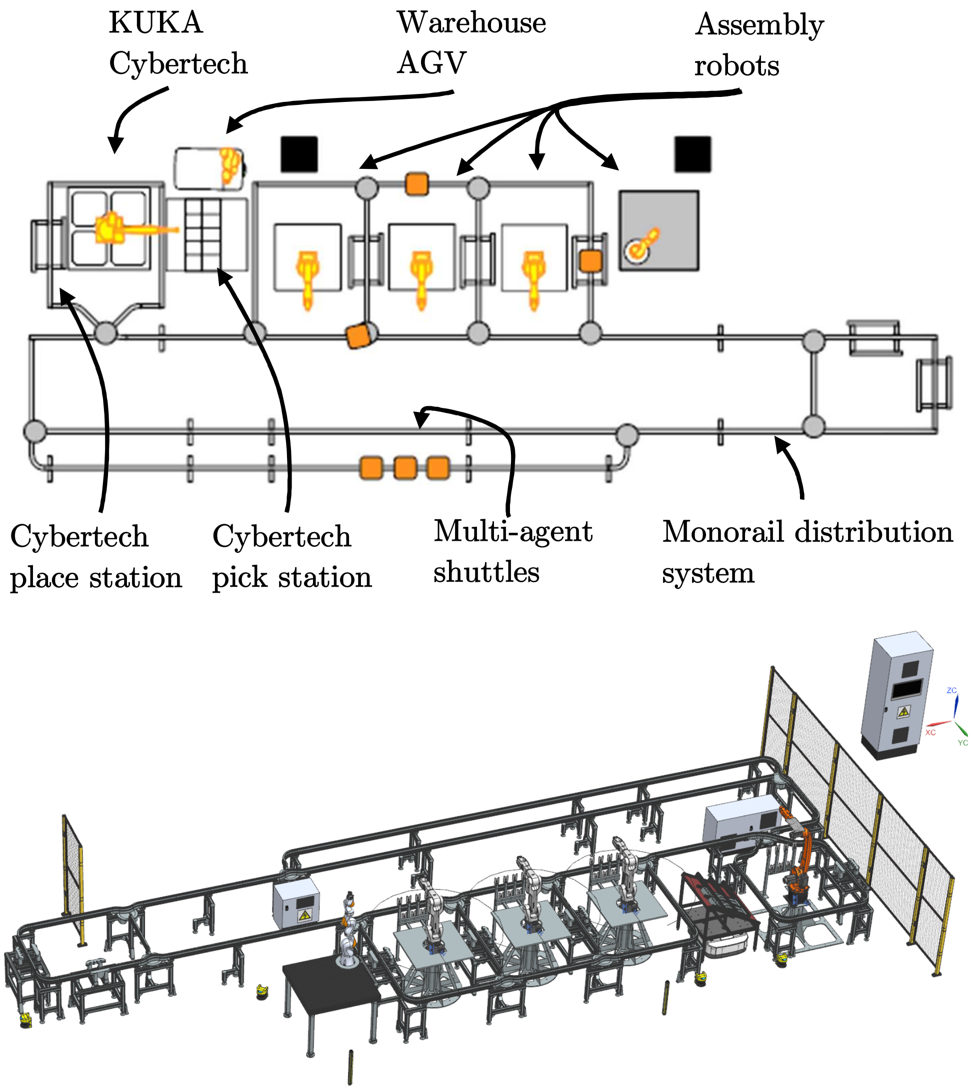


Figure 2.1: The Flexible assembly line model

replanning and continuing the assembly without the affected robot. Additionally, the assembly line can accommodate multiple product developments and assemblies simultaneously.

To demonstrate the obsolescence of predefined coordinate part placements by computer vision adaptation on Cybertech (as stated in Chapter 1), as well as evince the above-described adaptability of the Flexible assembly line, an entirely new product for assembly will be developed (see Fig. 2.4 and Section 1.1), tested and deployed, with retained cross-compatibility to previous assembly line states.

In order to enable the Cybertech robot to autonomously pick parts of varying shapes, sizes, and orientations (see Fig. 2.3), including those that are yet to be defined in the simultaneous product development, a future-proof solution

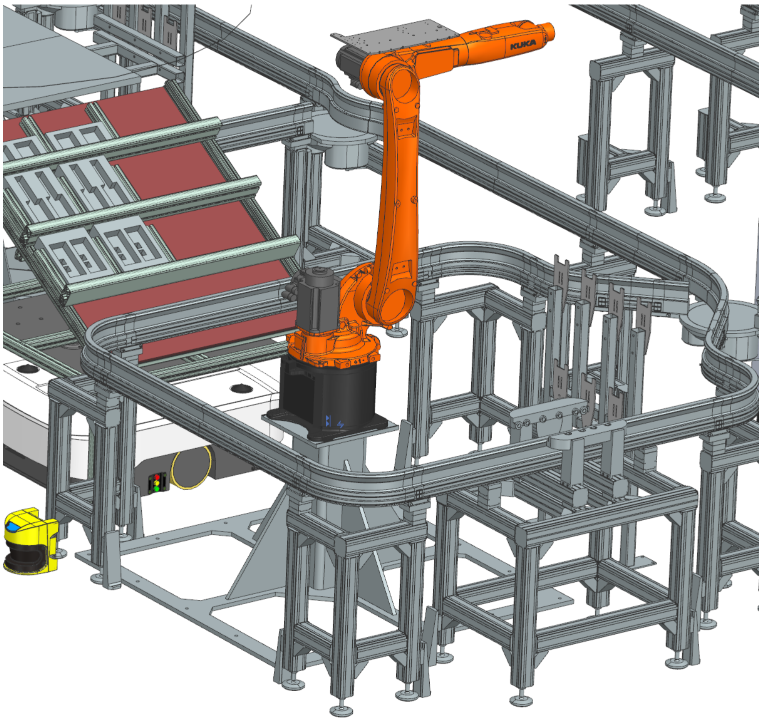


Figure 2.2: KUKA Cybertech Model

for adding new parts is required. While a Neural Network (NN) solution (such as in [KSKP21]) would require retraining for each newly added part, the use of ArUco tags [Bra00] was considered a more viable methodology.

Figure 2.3: Various to-be-picked parts

Since the assembly product is entirely manufactured in-house (see Fig. 2.4), it is possible to print identification tags on all parts during the manufacturing process. As a result, adding a new part to the Cybertech mission script will be accomplished in a matter of seconds, without any additional complications. Beyond the scope of this thesis, the ultimate objective is an integration of the ABB assembly line (seen in [Jí22]), Delta assembly line (seen in [Vor22]), and the Flexible assembly line (seen in Fig. 2.1) to enable a consecutive and autonomous production, with the vision-enabled Cybertech playing a key role in distributing parts of various shapes, sizes, placements, and rotations from an inbound AGV to the Flexible assembly line distribution shuttles.

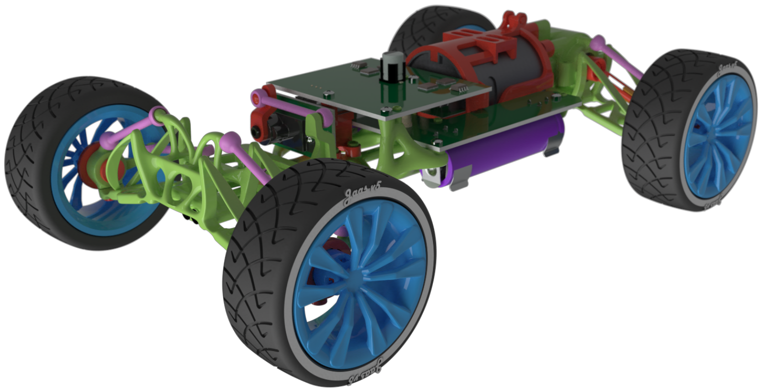


Figure 2.4: Finalized RC car as an assembly product

2.1 Summary of the Works

To summarize, the project will employ already integrated communication protocols, such as Profinet [PRO23], IO-Link [IL23] and OPC UA [Cor23]. The methodology and mission control will be carried out in accordance with the needs of MES, inbound AGVs, meet the requirements of adjacent assembly lines, devices, and distribution shuttles [MB23] [NDKV22], [WVN+19b], [WVN+21], [JKO18], [WVN+19a], [Jí22], [Vor22], [Sta23].
KUKA Robot Language [KRL23] will be used for direct robot control and controller-based transformations within the Workvisual interface [AG23], Python language for mission control [VRDJ95], open source libraries for pose estimation and part localization [Bra00], Basler libraries for camera communication client development [Bas23], and TIA Portal [Sie23] for hardware configuration and mapping.

The following discussions or implementations will be presented in this work:

1. Analysis of computer vision options
2. Camera selection
3. Camera communication client
4. Camera semi-automatic calibration
5. Vacuum gripper development & Valve control
6. Gripper tool changer
7. Eye-in-hand system & transformations
8. Graphical User interface & Human-Machine interface
9. Mission Control
10. Eye-to-Hand validation system
11. Real-life deployment
12. Summary of findings and future work

Chapter 3

An Analysis of Computer Vision Options and Camera Selection

This chapter will analyze and compare the various types of perception systems, evaluate their strengths and weaknesses, make a decision on the system to be used, and proceed with the construction of the selected system.

3.1 Vision options and requirements

It is crucial that the selection criteria are thoroughly researched, and that the preferred options and specifications of the computer vision systems are established in the early stages of the project to avoid potentially costly and time-consuming modifications later on. Change of system after its initial installation due to unsatisfactory imaging results or unmet project requirements can result in significant technical challenges such as the need to reinstall proprietary camera software, rewire cables, and modify the mounting mechanism. In extreme cases, such as switching from mono-vision to stereo-vision, the entire project perspective may need to be reevaluated and a substantial number of algorithms may need to be revised.

Therefore, a discussion and decision on the available system options will be made with the aim of constructing a universal and future-proof vision system.

3.1.1 Vision Placement

The positioning of the camera with respect to the robotic arm has a direct impact on the accuracy, efficiency and applicability of the system's perception capabilities and decision-making performance. The three widely used options for camera placement are Eye-in-hand, Eye-to-hand and Upward-looking (see Fig. 3.1), each offering different advantages and disadvantages.

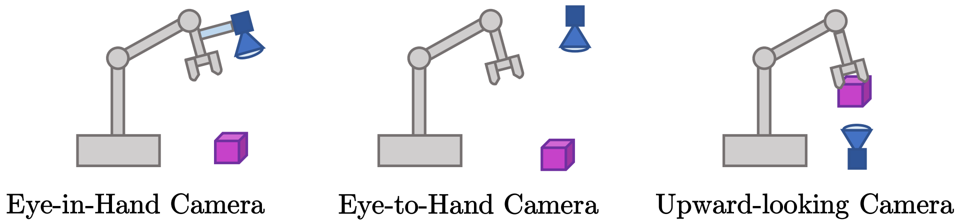


Figure 3.1: Spatial relationship options

1. **Eye-in-Hand** places the camera directly on the end of the robotic arm, providing the most direct view of the environment within the robot's full workspace.
2. **Eye-to-Hand** approach mounts a camera separately and oversees the robot's ROI[1], where the picking procedure is executed.
3. **Upward-looking** mounts the camera separately and inspects a small portion of the environment, often used to scan already picked or processed parts.

Although the to-be-picked parts and manufacturing process might not be known in this project stage due to the still ongoing RC car development, the fundamental concept is clear; Multiple objects of unknown structure, in unknown locations, and with dimensions in the range of centimeters are intended to be picked by an industrial robotic arm. Therefore, some key prerequisites can already be recognized, such as the need for scanning, part classification, pose estimation, and picking transformations.
The Upward-looking option can, therefore, be ruled out, as the method assumes that an object is already picked and awaits further perception-based inspection.
The Eye-to-Hand method offers several benefits, such as the possibility of independent robot movements and camera frame capturing, leading to a significant reduction in the operation cycle time. The coordinate frame transformation of the relationship between the camera and robot base is

[1] Region of interest

static, thus simplifying the project's implementation. The primary drawback of this approach is its potential insufficiency for densely populated assembly lines due to the possible interference between the vision system and the robotic arm, leading to obstruction issues. Additionally, the robot's ability to perform vision-based operations is limited to the camera's field of view, thereby restricting its adaptability and applicability in certain scenarios. Considering the objective of this work, which is to address the dynamic changes in the inbound AGV location and ensure applicability to other robots with varying workspace integration, it is apparent that this method may not be suitable.

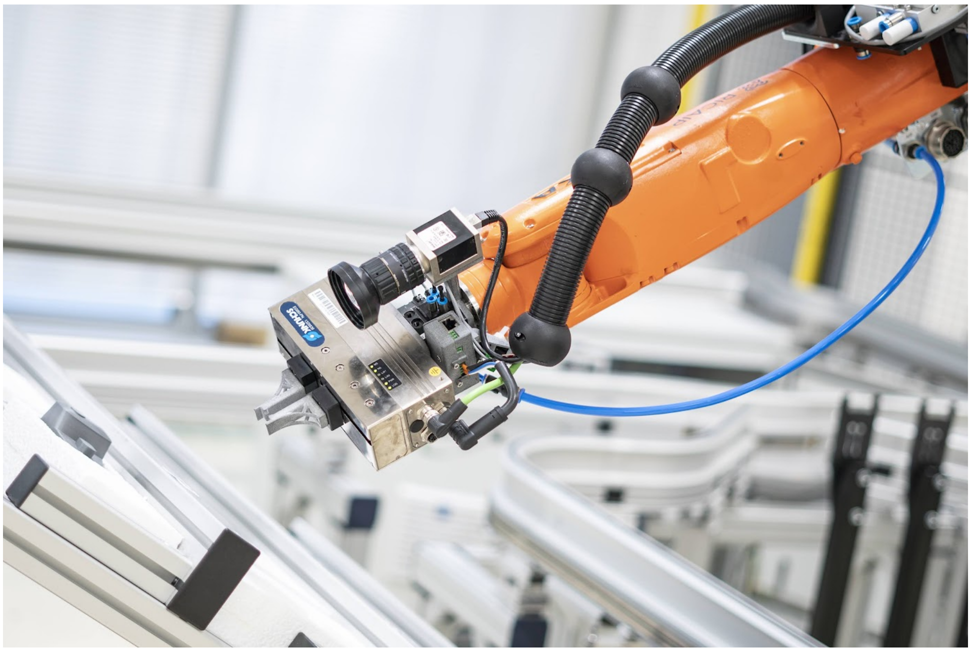


Figure 3.2: Cybertech Eye-in-Hand testing

The Eye-in-Hand method provides increased flexibility by allowing the perception ability to be maintained throughout the robot's operating range. Thus, the same system can be employed in case of any changes in assembly methods or the need to use vision on another assembly station within the robot's reach. However, this method does require the robot to stop and take a picture, making it more challenging to engineer, mainly due to the scarcity of available software solutions and study material on the topics of dynamic Eye-in-Hand transformations, as the available literature (such as [WHS23]) primarily focuses on methods and algorithms to solve hand-eye calibration problems, rather than ready-to-use software solutions. The thesis aims to achieve generality and flexibility, in which case the benefits of this approach outweigh the drawbacks, making it suitable for adoption. As such, the Eye-in-Hand method will be employed.

3.1.2 Depth information

Since tasks involving 3D objects will be performed by the robot, it is essential to consider stereo vision, which provides the ability to perceive depth information.
A stereo-vision technique engages two cameras (described in [Sze11]), which are positioned parallel to each other and separated by a known distance along the baseline b (see Fig. 3.3). Given that the focal lengths f are equivalent[2], and Z is in the direction of the optical axis, the left and right projections u_L and u_R of the point P from the left and right cameras can be found, respectively. Then, computation of the disparity d is possible. The disparity is inversely proportional to the depth and depth information can enable robots to understand the 3D structure of the environment, hence allowing the bin-picking grasping of objects of unknown structure.

$$u_L = f\frac{x_A}{z_A} \qquad u_R = f\frac{x_A - b}{z_A} \tag{3.1}$$

$$d = (u_L - u_R) = f\frac{b}{Z_A} \tag{3.2}$$

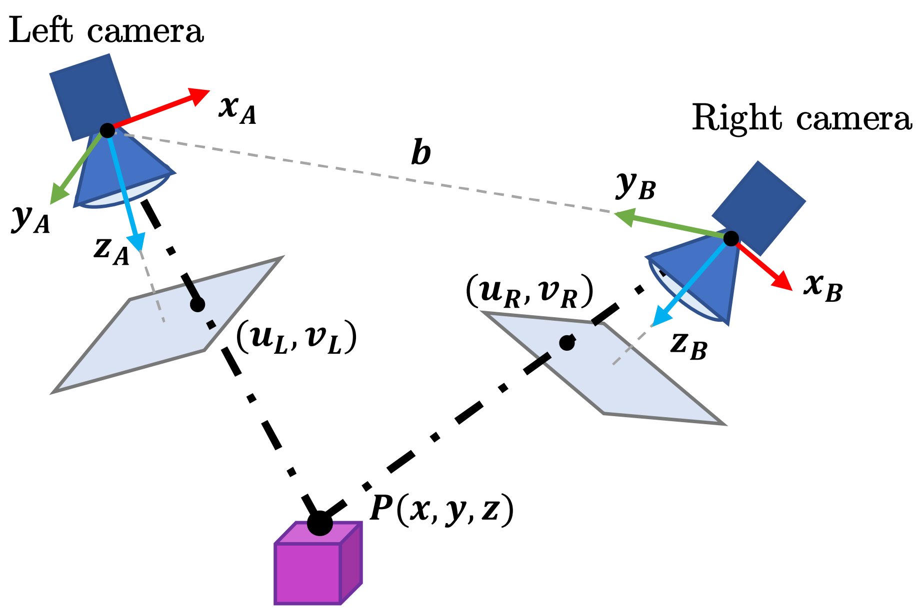


Figure 3.3: Simplified epipolar geometry

Using this capability, differentiation can be made between objects that are located far away from those that are in close proximity. To accurately

[2] Further information, material, and conventions will be described in Section 5.1

perceive depth, however, it is needed to perform stereo calibration, camera rectification, deep knowledge of epipolar geometry, and the development of robust feature-matching algorithms are needed.

Figure 3.4: Disparity map inaccuracy[MFR+22]

A project report [MFR+22] describes the inaccuracy and difficulties of such an approach. Amongst other resolutions, the report has come to a conclusion of severe uncertainty of depth estimation in stereo perception for small and shallow objects. In our scenario, picking printed circuit boards (seen in Fig. 2.3) that are only a few millimeters thick would pose a significant challenge, even for a stereo system that is calibrated to a high degree of accuracy.
Furthermore, a robust high-end industrial stereo vision system often costs in the range of thousands of euros and requires the use of licensed proprietary software. Therefore, a monocular system is chosen for this project and depth information can be retrieved by ArUco tags [Bra00] and multiple-view pose estimation.

3.2 Eye-in-Hand Vision System Requirements

With the knowledge that a monocular Eye-in-Hand system is used, a further definition of the system parameters must be settled[Sas23]:

- Field of View
- Spectroscopy (BW/color)
- Frame rate
- Depth of Focus
- Image resolution & pixel size
- Distortion and calibration complexity

- Connectivity options

In our application, there is a need for variability in scanning inbound parts in unknown locations, thus a wide-angle lens is deemed suitable to provide a broader range of view. For ease of human-machine interaction and graphical user interface implementation, a color (RGB) feed is required instead of black and white images, although this does not affect the computation pipeline.
A high depth of focus is required to ensure the variety of proximities of the parts, as their depth cannot be assumed.
Aperture settings, which determine how much light hits the sensor, also affect the depth of focus. To accommodate both sufficient light transmission and high depth of focus, a sensor with larger pixel dimensions is needed.
To obtain a clear image of small parts that are far away, a high-resolution sensor is also required, which may result in a reduced frame rate of only a few frames per second. As previously stated, the Eye-in-Hand method requires the robot to stop to take a picture, and hence the reduced frame rate is not an issue.
To accommodate a large sensor and a wide field of view, a large lens is needed, which can introduce unwanted distortion and "fish eye" effects (as described in [HGJD08]). Given proper execution, meeting all the requirements and eliminating any undesired distortions through the calibration process can be achieved.

3.2.1 Decision on the Vision System Used

In summary, the specific requirements for the vision system chosen for the project have been discussed in previous sections. The chosen system must be able to effectively scan inbound parts in unknown locations, and therefore it needs to meet certain criteria, including:

- A wide-angle lens to provide a large field of view for scanning
- RGB feed for HMI & GUI visualizations
- High depth of focus to accommodate the varying part proximity
- Large pixel dimensions to allow for enough light transmission
- High-resolution sensor to ensure clear images of small objects
- Frame rate does not affect the project methodology
- Calibration of potential "fish-eye" effect

Consequently, after evaluating different options, the Basler acA4112-8gc IP camera coupled with the Basler C23-0824-5M-P wide-angle lens is selected as the most suitable choice for our needs. This camera offers a resolution of 4096 px x 3000 px, a high-quality 1.1" RGB Sony sensor, an 8 mm focal length, and a variable aperture ranging from F2.4 to F16, making it capable of fulfilling all the necessary requirements.

Figure 3.5: The chosen vision system

Chapter 4

The Project Prerequisites: Communication, Automation and Gripper Design

The success of the Project's goals heavily depends on the ability to establish reliable communication between different components of the system, the development of automation tools, valve control mechanisms, and user interfaces, as depicted in Figure 4.1. Furthermore, the design of a suitable tool plays a pivotal role in the project's overall success, as it is responsible for an accurate and reliable grasping of objects.
This chapter, therefore, highlights such implementations and serves as a prerequisite for future Project development.

The material presented in this chapter, as well as some of the implementations of hardware configurations, were made possible with the invaluable assistance of colleagues and are described for a coherent understanding of the Project development only.

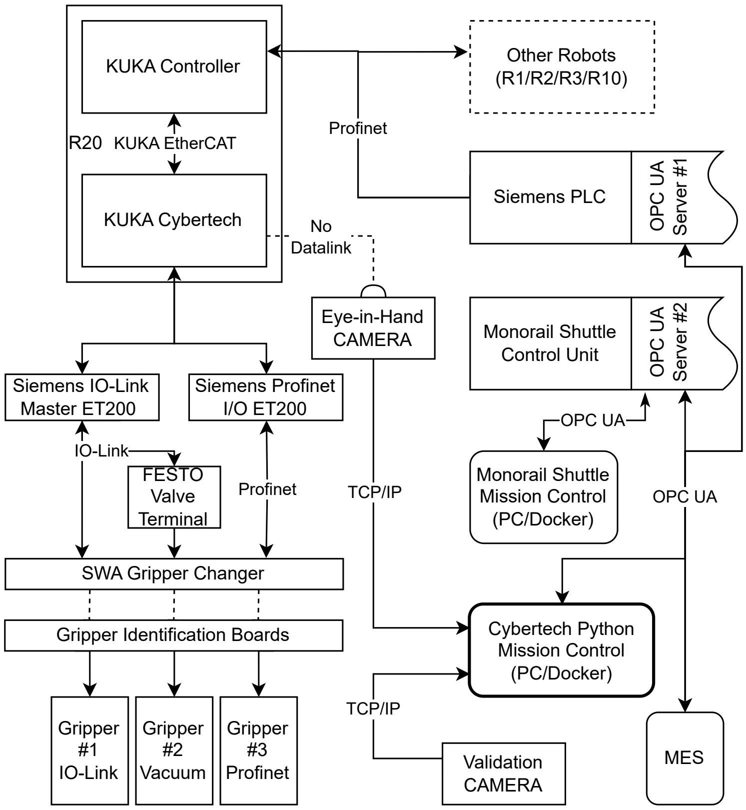


Figure 4.1: Simplified diagram of communication across systems and its hierarchy

4.1 Graphical User Interface

The development of a Graphical User Interface (GUI) (seen in Fig. 4.2) provides visual feedback during operation and ensures flexibility of the system, as it not only provides debugging information but also accommodates the possibility of the Flexible assembly line innovation processes to take place by semi-autonomous, button-driven functionality.
The GUI serves as the primary means of interaction with Cybertech, allowing operators to control its movements, select tasks, and monitor its progress. Additionally, the GUI provides a way to validate results with a secondary vision system, further described in later chapters.

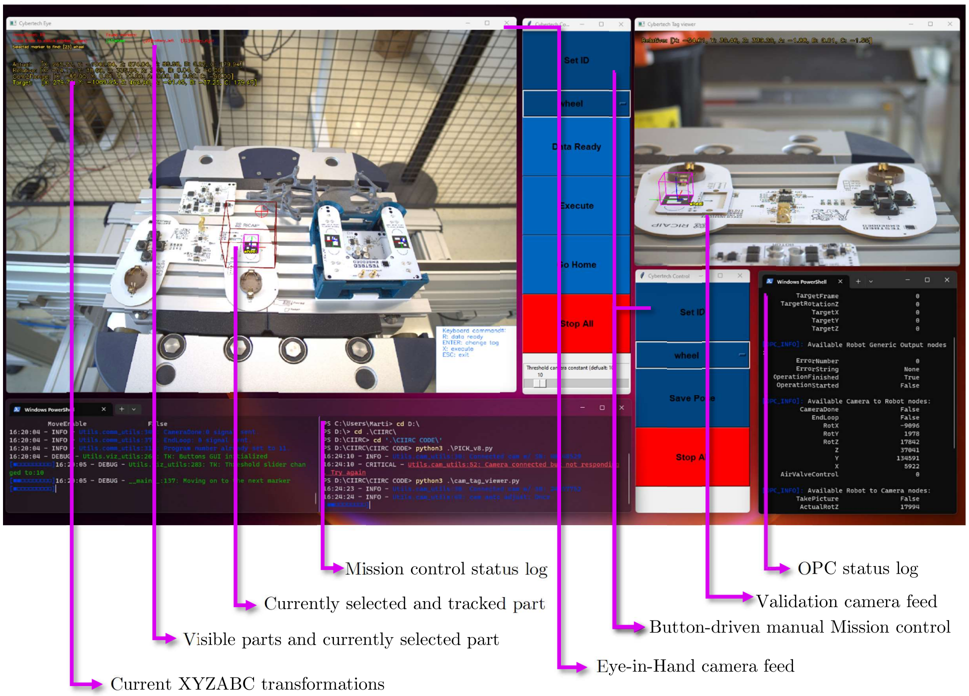


Figure 4.2: GUI depicting the Eye-in-hand view, part selection, tracking radius, visible parts, and transformations in the left window. An auxiliary vision system is visible in the top-right window, as well as the interactive button layout and status logs

4.2 Tool and Valve Control

Several types of grippers were considered for the project, including those that are electrical and based on IO-Link or PROFINET, as well as pneumatic-driven.
After careful consideration, a vacuum-pneumatic gripper was selected as the primary choice. Unlike servo-electric grippers, this pneumatic gripper offers the advantage of picking shallow objects effectively. Several versions were developed and tested for the best performance (see Fig. 4.4). The gripper is designed to accommodate the grasping of various PCBs by vacuum generators and includes three suction cups that can be controlled independently, depending on the size of the part to be picked. In order to control the independent suction cups, the controlling unit (see Fig. 4.6) will receive commands via OPC UA from Mission control. The Mission control script performs all camera-related decision-making and all necessary calculations and will be described in more detail in subsequent chapters.

As picking different parts requires the usage of different grasping techniques, the grippers need to be changed to accommodate the production of a complex

RC car assembly.

To streamline the tool-changing process from Mission control, an automatic tool changer was created (see Fig. 4.3), and an identification board was incorporated into each gripper to provide MES and Digital twin (an implementation of a virtual real-time representation of a physical Flexible assembly line that enables monitoring, analysis, and optimization of its performance) status feedback. The hardware configuration for the valve functionality was achieved using FESTO electromagnetic Valve Cards in WorkVisual and connected via IO-Link Master (reference to Fig. 4.5 and Fig. 4.6). A dedicated Valve Card is used for each suction cup and the tool changer locking and unlocking mechanism, enabling separate control of each component and picking of different parts of concave shapes. Furthermore, the tips of the suction cups on the designed gripper are spring-loaded to allow for errors in the Z-axis.

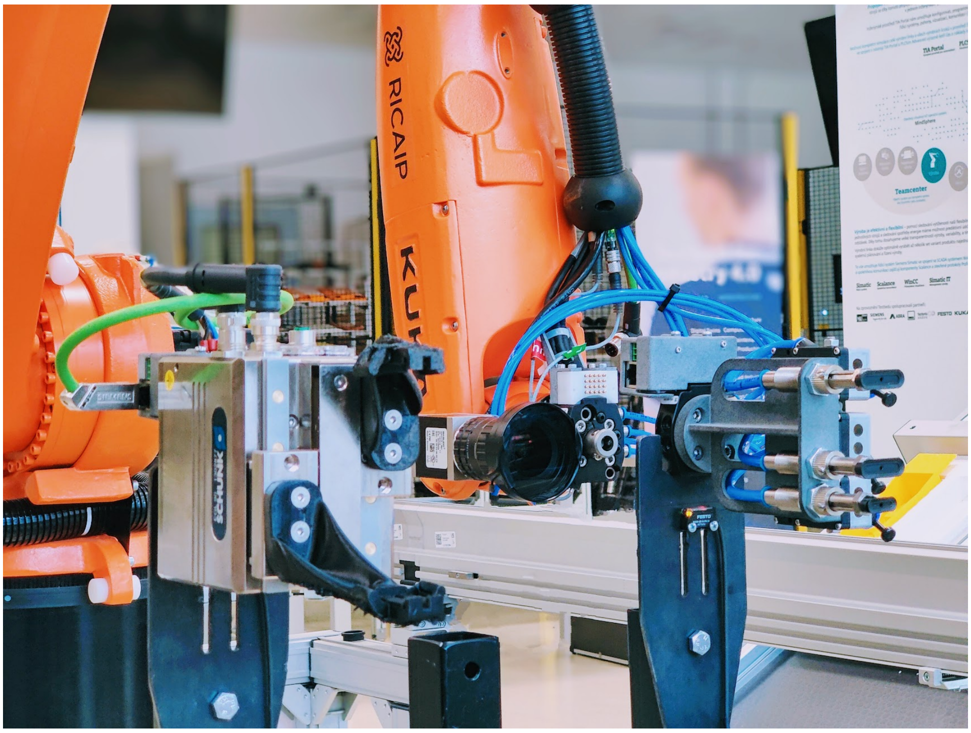
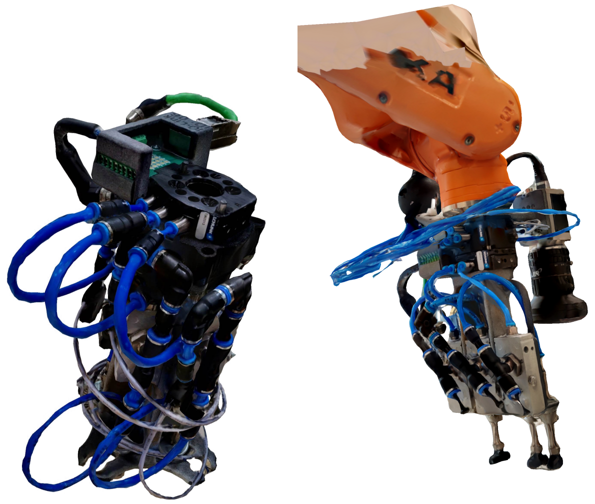


Figure 4.3: Demonstration of an automatic tool change on Cybertech



(a) (b)

Figure 4.4: Vacuum gripper versions with visible variable suction cups, pressure sensors, tool ID board, and Schunk Workpiece Adapter (SWA) tool changer

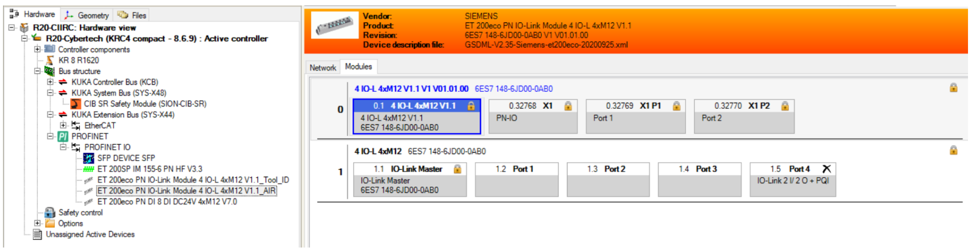


Figure 4.5: IO-Link Master mapping in Hardware Configuration

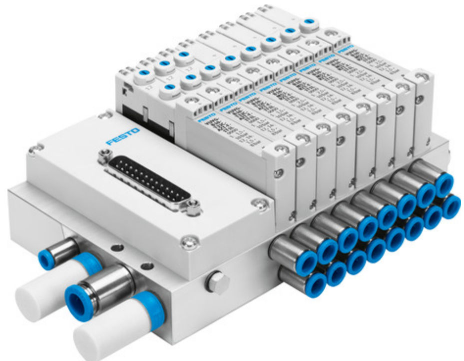


Figure 4.6: FESTO Valve terminal for Valve Control functionality

4.3 Server-based Communication and Logging

PLC Mapping

To ensure that the project is widely applicable, it was necessary to incorporate the option for remote communication. For example, this feature enables the deployment of scripts on a remote server. To achieve this, a communication system was established using OPC Unified Architecture (OPC UA) to link the PLC, Cybertech and Mission control device. As a result, robot communication is facilitated remotely and the communication system was tested successfully on both wired networks and Wi-Fi connections.

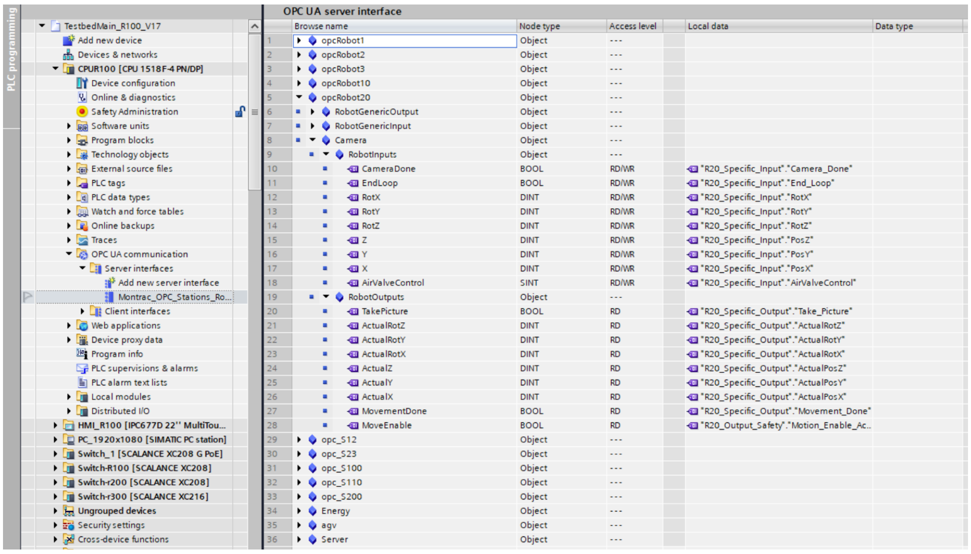


Figure 4.7: PLC mapping configuration in TIA Portal

Cybertech Mapping

In order to create a connection between the systems, an OPC UA server interface was established, along with the necessary communication values on the PLC (see Figure 4.7). A similar mapping was then performed in the Cybertech interface (see Figure 4.8). However, it should be noted that Siemens PLCs use a system of 16-bit memory words in registers, while KUKA robots use 32-bit registers. Therefore, a conversion calculator was implemented to ensure a consistent mapping between these two systems in Figure 4.12.

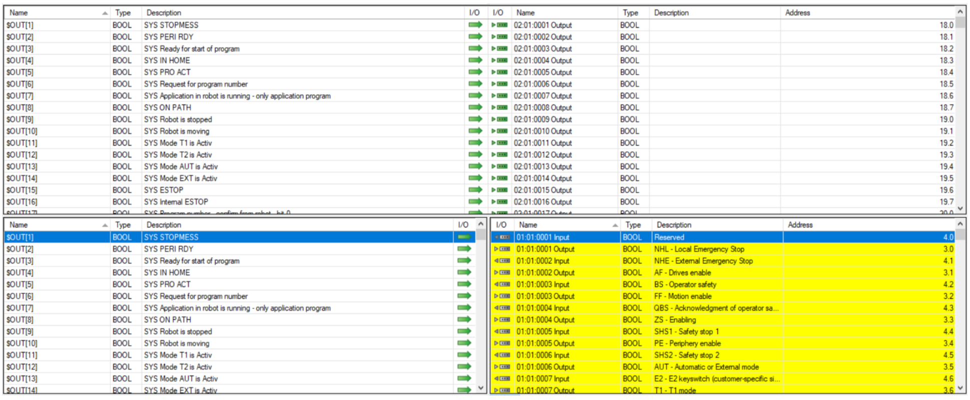


Figure 4.8: Robot mapping configuration in robot controller by Workvisual

Mission Control Mapping

On the Mission control side, i.e. remote computational device, the OPC UA client was implemented in Python, with the functionality of an automatic revision of the currently mapped values. Therefore, if a new value is mapped and added in PLC and robot, it is automatically recognized as a class variable in the Mission control script (see Fig. 4.9).

```
[OPC_INFO]: Available Robot Generic Output nodes:
         ErrorNumber                 0
         ErrorString              None
   OperationFinished              True
    OperationStarted             False

[OPC_INFO]: Available Camera to Robot nodes:
          CameraDone             False
             EndLoop             False
                RotX            -11100
                RotY              1900
                RotZ             17900
                   Z             55000
                   Y            115200
                   X              5100
     AirValveControl                 0

[OPC_INFO]: Available Robot to Camera nodes:
         TakePicture             False
          ActualRotZ             17800
          ActualRotY              1100
          ActualRotX             -7900
             ActualZ             80100
             ActualY            122200
             ActualX             18978
        MovementDone             False
          MoveEnable             False
```

Figure 4.9: Mission control OPC client with an automatic node updater

In addition to the existing features, a comprehensive Mission control logger was developed that allows for the tracking and recording of all operations and events within the system. A visual representation of the logger can be seen in Fig. 4.10, and previews a brief glance at the autonomous functionality of bin picking, researched in further chapters.

```
12:57:14 - INFO - Utils.comm_utils:349: CameraDone:0 signal sent.
12:57:14 - INFO - Utils.comm_utils:373: EndLoop: 0 signal sent.
12:57:14 - INFO - Utils.comm_utils:313: Program number already set to 11.
12:57:15 - DEBUG - Utils.viz_utils:260: TK: Buttons GUI initialized
[■□□□□□□□□□]12:57:15 - DEBUG - Utils.viz_utils:283: TK: Threshold slider changed to:10
[■■■■■■□□□□]12:57:42 - DEBUG - Utils.viz_utils:290: TK: Tag focus changed to:battery_left
[■■■■■■■□□□]12:57:42 - DEBUG - __main__:137: Moving on to the next marker
[■■■■■■■■■■]12:58:17 - DEBUG - Utils.viz_utils:179: TK button pressed: Data Ready!
12:58:17 - INFO - Utils.comm_utils:294: DataReady signal sent.
[■□□□□□□□□□]12:58:17 - INFO - __main__:232: MES allowed for approach
[■■□□□□□□□□]12:58:18 - DEBUG - __main__:335: XYZ: [58.749350281914204, 1350.810130269893, 371.73922255049075]
12:58:18 - DEBUG - __main__:336: ABC: [-91.18452473861862, 19.775271421682028, 178.83704053793124]
12:58:18 - WARNING - __main__:345: EXECUTING!!!
12:58:18 - INFO - Utils.comm_utils:252: Robot pose sent to robot
12:58:18 - INFO - Utils.comm_utils:346: CameraDone:1 signal sent.
[■■■□□□□□□□]12:58:18 - DEBUG - __main__:137: Moving on to the next marker
[■■■■□□□□□□]12:58:18 - DEBUG - __main__:137: Moving on to the next marker
[■■■■■■■□□□]12:58:20 - DEBUG - __main__:244: Waiting iteration: 1
[■■■■■■■■□□]12:58:20 - DEBUG - __main__:244: Waiting iteration: 2
[■■■■■■■■■□]12:58:20 - DEBUG - __main__:244: Waiting iteration: 3
[■■■■■■■■■■]12:58:21 - INFO - Utils.comm_utils:370: EndLoop: 1 signal sent.
12:58:21 - INFO - Utils.comm_utils:294: DataReady signal sent.
12:58:21 - INFO - Utils.comm_utils:252: Robot pose sent to robot
12:58:21 - INFO - __main__:277: PICKING
12:58:21 - DEBUG - __main__:278: Valve mode: 0
12:58:21 - DEBUG - __main__:280: Sending robot to coordinates: [  59.23 1345.91   370.42  -90.97   19.79  178.43]
12:58:21 - INFO - Utils.comm_utils:346: CameraDone:1 signal sent.
[■■□□□□□□□□]12:58:21 - DEBUG - __main__:137: Moving on to the next marker
[■■■■■■■■■□]12:58:24 - DEBUG - __main__:137: Moving on to the next marker
[■■■■■■□□□□]12:58:32 - INFO - __main__:297: ----------- PICK SUCCESS ------------
12:58:32 - INFO - Utils.comm_utils:373: EndLoop: 0 signal sent.
[■□□□□□□□□□]12:58:34 - DEBUG - Utils.viz_utils:272: TK: StopAll button pressed
12:58:34 - INFO - Utils.comm_utils:386: Disconnected from OPC UA server.
```

Figure 4.10: Mission control status log

4.3.1 Pseudo-Threaded Robot Pose Transmission

To ensure the availability of the pose of the robot at all times, Submit Interpreter (`SUB`)[1] pseudo-thread communication was established. `SUB` is run parallel and sequentially next to a running program (`Cell`), at all times, and can't be stopped depending on the `Cell`'s needs.

Axiomatically, the current pose of the robot (`$POSACT`) requires a defined reference frame and target frame (e.g. `$TOOL` and `$BAS`). The validity of `$POSACT` can change, depending on various factors, such as a tool change, robot restart, or a program (`Cell`) initialization, rendering `$POSACT` definitions invalid, as one of its components may be momentarily undefined.
Attempts were made to restrict access and broadcasting of the `$POSACT` in Fig. 4.11 when the `$POSACT` occurs to be undefined. However, the probability of the definition of `$POSACT` becoming invalid within the `IF` statement of Figure 4.11, with conditions previously met, still persists due to the specific implementation of pseudo-threaded `SUB`, and is an unresolved challenge.

[1] A program running in a KUKA robot entirely independently of the selected robot program, which can be used to handle all manner of different control tasks, such as monitoring of safety equipment or the integration of additional peripheral devices [AG23].

```
;FOLD USER PLC
;Make your modifications here
;added Martin Miksik
IF ((VARSTATE("$POS_ACT")==#INITIALIZED) AND ($ACT_BASE == 0) AND ($ACT_TOOL == 10)) THEN
   ;if pos_act is initialized, meaning base and tool is defined.
   POS_ACT_TMP = $POS_ACT
   CAMERA_X_OUT_SUB = REVERTED_BYTES_SUB(POS_ACT_TMP.X*100)
   CAMERA_Y_OUT_SUB = REVERTED_BYTES_SUB(POS_ACT_TMP.Y*100)
   CAMERA_Z_OUT_SUB = REVERTED_BYTES_SUB(POS_ACT_TMP.Z*100)
   CAMERA_ROTX_OUT_SUB = REVERTED_BYTES_SUB(POS_ACT_TMP.A*100)
   CAMERA_ROTY_OUT_SUB = REVERTED_BYTES_SUB(POS_ACT_TMP.B*100)
   CAMERA_ROTZ_OUT_SUB = REVERTED_BYTES_SUB(POS_ACT_TMP.C*100)
ENDIF
;ENDFOLD (USER PLC)
```

Figure 4.11: Submit interpreter actual pose transmission script in KRL

Furthermore, since both PLC and KUKA use different byte sequence orders, a function was created in KRL that converts a 32-bit integer value from little-endian byte order to the big endian byte order (see Fig. 4.12). The function takes an input integer value (`IN_VALUE`) and performs bitwise operations to extract and rearrange the bytes of the value to produce the output integer value (`RESULT`) in big-endian byte order. The function also handles negative input values by setting the most significant byte of the output value to 1. Additionally, as seen in Fig. 4.11, values are sent as integer values [2], later converted back to floating points in the Mission control script in Python, upon receiving values over OPC UA.

```
DEFFCT INT REVERTED_BYTES_SUB(IN_VALUE:IN)
; Function to change endian, added by Martin Miksik
INT IN_VALUE, VI_A, VI_B, VI_C, VI_D, RESULT

VI_A = IN_VALUE B_AND 'B00000000000000000000000011111111'
VI_B = IN_VALUE B_AND 'B00000000000000001111111100000000'
VI_B = VI_B / 256
VI_C = IN_VALUE B_AND 'B00000000111111110000000000000000'
VI_C = VI_C / 65536
VI_D = IN_VALUE B_AND 'B01111111000000000000000000000000'
IF IN_VALUE < 0 THEN
   VI_D = VI_D / 16777216
   VI_D = VI_D + 128
ELSE
   VI_D = VI_D / 16777216
ENDIF

RESULT = VI_D + (VI_C*256) + (VI_B*65536)
IF VI_A > 127 THEN
   VI_A=VI_A B_AND 'B01111111'
   RESULT = RESULT + VI_A*16777216
   RESULT = RESULT B_OR 'B10000000000000000000000000000000'
ELSE
   RESULT = RESULT + VI_A*16777216
ENDIF
;
;RESULT = IN_VALUE
RETURN RESULT

ENDFCT
```

Figure 4.12: Byte sequence conversion in KRL

[2] due to the mapping ability of PLC (see Section 4.3)

Chapter 5

Camera Calibration: An Overview of Intrinsic and Extrinsic Parameters

One of computer vision's fundamental problems is recovering a scene's three-dimensional structure from its images. The goal is, therefore, to determine the location of a reconstructed scene's point, in terms of millimeters, by analyzing a picture.
When interpreting a picture, only a pixel metric is at the disposal. To achieve metric world reconstruction, two key pieces of information are required. The first requirement is the knowledge of how the camera maps the perspective projection's points in the World onto its image plane. The second is the knowledge of the camera's position and orientation with respect to the World frame.
Accordingly, the theory behind camera calibration will be introduced in this chapter (with a reference to [RA04] and [Sze11]), which will allow for the camera to be treated generally and as a black-box device. Homogeneous coordinates and unknown internal and external parameters of the camera will be described and semi-automatic calibration OpenCV implementation will be developed.

5.1 Calibration Theory

Let us have a point $P \in \mathbb{R}^3$ in some[1] World coordinate system $\mathcal{W}$. A camera is represented by its coordinate frame $\mathcal{C}$, where the Z axis of the coordinate frame is aligned with the optical axis of the camera[2]. Let us also assume that the *focal length* f is the distance between the effective central projection and the camera's image plane. This can also be described as the distance at which a beam of collimated light will be focused at a single point.

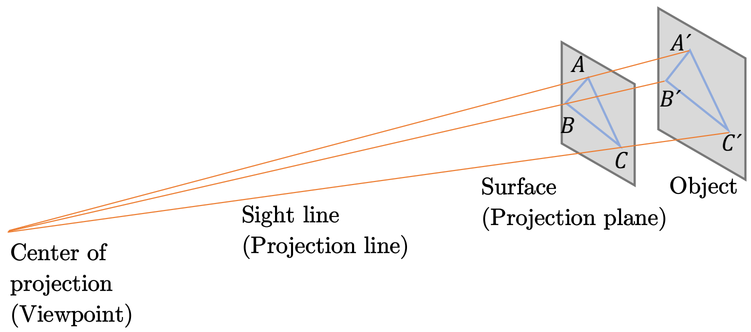


Figure 5.1: Central projection

Using a *Pinhole* [Stu14] *Forward Imaging Model* (see Fig. 5.2), also often referred to as 3D to 2D model (Eq. 5.1), the relative position and orientation of the camera coordinate frame with respect to the World coordinate frame can be found, and it is called the *Perspective projection*:

$$\boldsymbol{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \longrightarrow \boldsymbol{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{5.1}$$

From the Model, it can be seen that:

$$\frac{x_i}{f} = \frac{x_c}{z_c} \tag{5.2}$$

Therefore,

[1]For the Theory section, any World coordinate system can be defined, although it can be the `$NULLFRAME` or the Flange of the robot for our application

[2]The optical axis can be symbolized as a line passing through the center of the lenses.

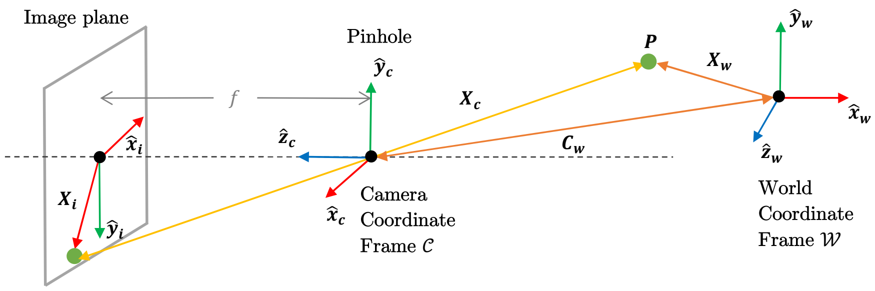


Figure 5.2: Forward Imaging Model

$$x_i = f\frac{x_c}{z_c} \qquad\qquad y_i = f\frac{y_c}{z_c} \tag{5.3}$$

Which gives us the representation of the point P in the image plane in Figure 5.2.
As the sensor operates in pixel metrics, a way to find the conversion of the point into millimeters needs to be introduced.

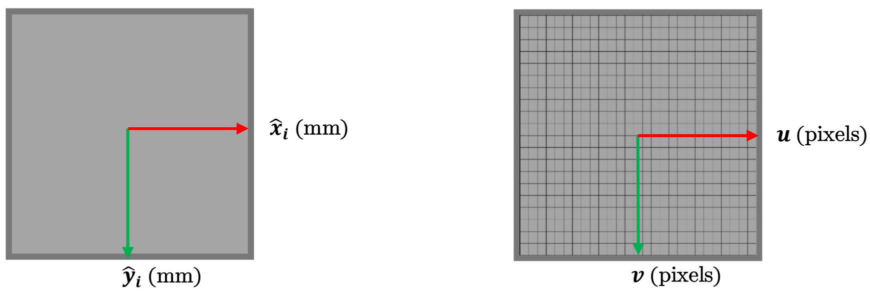


Figure 5.3: The Image Plane (left) and the Image sensor (right)

As there is no reason to believe that the dimensions of the pixels are square, *Pixel densities* m_x and m_y are introduced in the x and y directions, respectively.

$$u = m_x x_i = m_x f\frac{x_c}{z_c} \qquad\qquad v = m_y x_i = m_y f\frac{y_c}{z_c} \tag{5.4}$$

As the thesis aims for generality and assumes a black-box camera, the m_x and m_y are considered unknown.
However, it is possible to compute the pixel density by examining the manufacturer's system description and deriving the unknowns from the dimensions of the pixels and sensor. This knowledge will be used in later sections to prove the validity of the upcoming calibration process.
One more aspect needs consideration, as it has been assumed until now that the center of the image is known. The image plane's center corresponds to the image's center, where the optical axis pierces the image plane. However,

there is no reason to assume that the location of that point is known and most manufacturers do not provide this information. This point will be referred to as the *Principal point* (o_x, o_y), and will remain unknown, subject to be found in the calibration process.

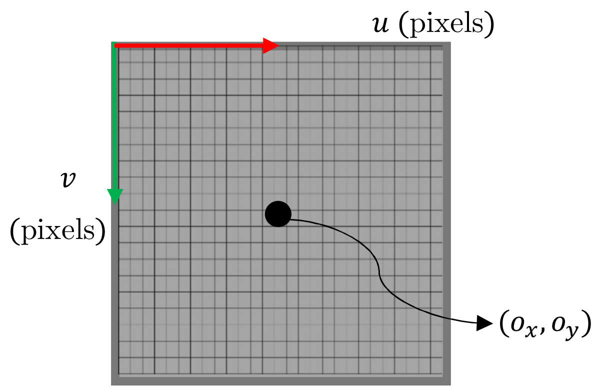


Figure 5.4: The Image sensor with shifted coordinate frame

That being said, Equation 5.4 can be modified to provide a more complete representation of the projection equations.

$$u = m_x f \frac{x_c}{z_c} + o_x \qquad v = m_y f \frac{y_c}{z_c} + o_y \tag{5.5}$$

It is important to remember that m_x, m_y, f and o_x, o_y in Equation 5.5 represent the unknown parameters.
Then, the focal length and pixel density can be combined as f_x and f_y, referred to as the *effective focal lengths*, from which the commonly known optical equations can be derived[Sze11].

$$u = f_x \frac{x_c}{z_c} + o_x \qquad v = f_y \frac{y_c}{z_c} + o_y \tag{5.6}$$

It is known that a camera generally has only one effective focal length. However, as mentioned earlier, the non-equal pixel density in the x and y directions can be handled by introducing f_x and f_y, respectively. The unknowns of equations 5.6 are commonly known as the *intrinsic parameters* and represent the internal geometry of the camera.

Note that Equation 5.6 has z_c in the denominator, making the model non-linear. To simplify projection estimation, it is preferable to express the model using linear equations.
The homogeneous representation of a 2D point $u = (u, v)$ is a 3D point

$\tilde{u} = (\tilde{u}, \tilde{v}, \tilde{w})$, where $\tilde{w} \neq 0$ is fictitious, such that

$$u = \frac{\tilde{u}}{\tilde{w}}, v = \frac{\tilde{v}}{\tilde{w}} \tag{5.7}$$

Then,

$$\mathbf{u} \equiv \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{w}u \\ \tilde{w}v \\ \tilde{w} \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \tilde{\mathbf{u}} \tag{5.8}$$

This could be graphically presented, as if every point $\tilde{u}$ on line L (except origin) represents the homogeneous coordinate $u = (u, v)$, in Figure 5.5

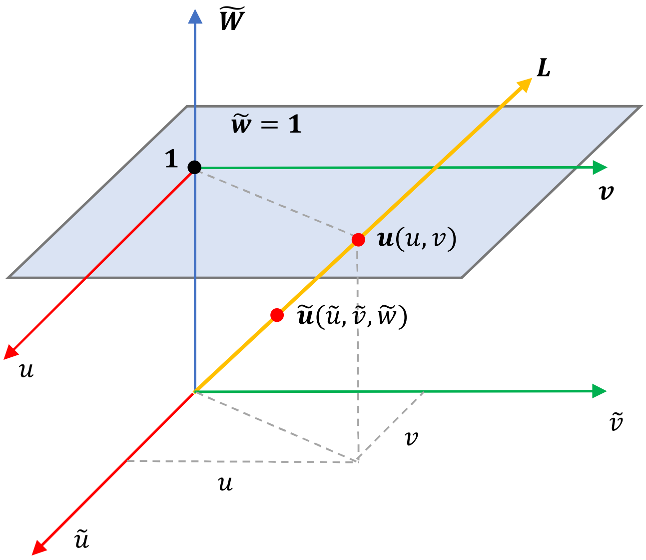


Figure 5.5: Visual representation of homogeneous coordinate transformation

The same representation can be extended to 3D coordinates, although visualization of such a representation becomes difficult or impossible, as the homogeneous representation of a 3D point $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$ is a 4D point $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{z}, \tilde{w}) \in \mathbb{R}^4$. The fourth coordinate $\tilde{w} \neq 0$ is fictitious such that:

$$x = \frac{\tilde{x}}{\tilde{w}} \quad y = \frac{\tilde{y}}{\tilde{w}} \quad z = \frac{\tilde{z}}{\tilde{w}} \tag{5.9}$$

$$\mathbf{x} \equiv \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{w}x \\ \tilde{w}y \\ \tilde{w}z \\ \tilde{w} \end{bmatrix} \equiv \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{bmatrix} = \tilde{\mathbf{x}} \tag{5.10}$$

By using the knowledge from Equations 5.7 and 5.8, the Perspective projection Equation 5.6 can be expressed in terms of homogeneous coordinates.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \widetilde{w} \end{bmatrix} \equiv \begin{bmatrix} z_c u \\ z_c v \\ z_c \end{bmatrix} = \begin{bmatrix} f_x x_c + z_c o_x \\ f_y y_c + z_c o_y \\ z_c \end{bmatrix} \tag{5.11}$$

The expression can be further decomposed as follows:

$$\tilde{u} = \begin{bmatrix} f_x x_c + z_c o_x \\ f_y y_c + z_c o_y \\ z_c \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{M_{int}} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{5.12}$$

$$\tilde{\mathbf{u}} = [K \mid 0]\tilde{\mathbf{x}}_c = M_{int}\tilde{\mathbf{x}}_c \tag{5.13}$$

Where M_{int} is the *Intrinsic camera matrix* , holding 3x3 Upper Right Triangular *Calibration matrix* K, compactly representing the camera's internal geometry.

$$K = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.14}$$

Now that the internal parameters are defined, c_w (see Fig. 5.1) can be sought. c_w is defined by translation vector t and a rotation matrix $R \in SO(3)$[3], which describes the orientation of the camera in the World coordinate frame $\mathcal{W}$. Such parameters are called the *Extrinsic parameters*.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{5.15}$$

The first, second, and third row of R in Equation 5.15 correspond to the direction of $\hat{x}_c, \hat{y}_c, \hat{z}_c$ in the World coordinate frame, respectively.

Looking at a section of Figure 5.2, represented in Figure 5.6, the camera-centric location of the point P can be expressed in the World coordinates, by

[3] 3D Rotation Group [RA04]

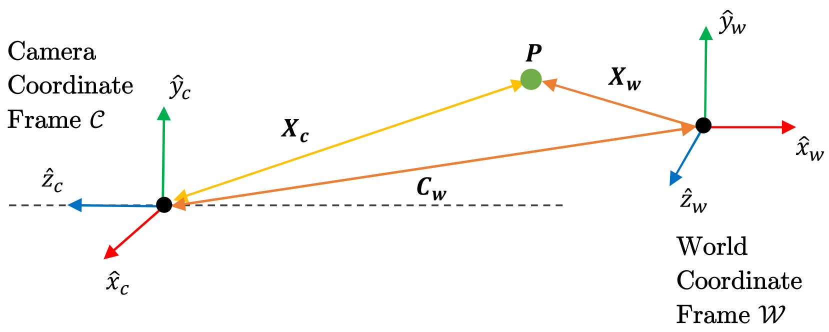


Figure 5.6: Graphical representation of pose transformation

finding x_c, represented by the yellow vector in Figure 5.6, which is just a mere subtraction of x_w and c_w, both being later multiplied by the rotation matrix. Simplified with the convention of $t = -Rc_w$, the solution can be found:

$$\mathbf{x}_c = R\left(\mathbf{x}_w - \mathbf{c}_w\right) = R\mathbf{x}_w - R\mathbf{c}_w = R\mathbf{x}_w + \mathbf{t} \tag{5.16}$$

$$\mathbf{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{5.17}$$

The Equation 5.17 represents the position and orientation of the camera in the World coordinate frame, and rewritten in homogeneous coordinates, holds the *Extrinsic matrix* M_{ext}, which defines the spatial relationship (i.e. position and orientation) of the camera (i.e. the camera's optical centre[4]) in 3D space, relative to a known reference frame (i.e. flange or tool of the robot):

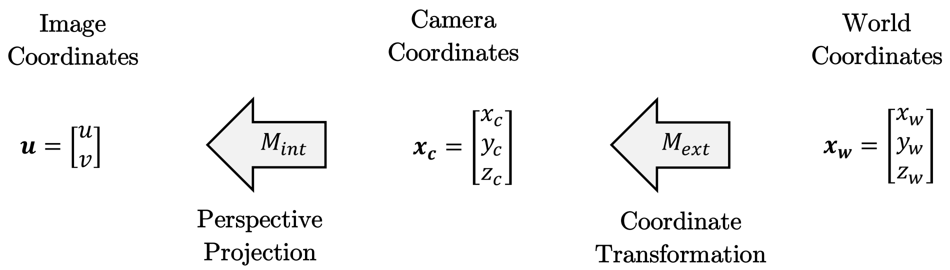
$$\tilde{\mathbf{x}}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{M_{ext}} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{5.18}$$

[4]Intersection of the optical axis and the image plane of a camera. It is the point where light rays converge after passing through the camera lens, often synonymous to the principal point, with an allowed deviation in the manufacturing process[PS06]

$$\tilde{\mathbf{x}}_c = M_{ext}\tilde{\mathbf{x}}_w \tag{5.19}$$

$$M_{\text{ext}} = \begin{bmatrix} R_{3\times3} & \mathbf{t} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.20}$$

Now, all the needed components were taken care of, the definition of a perspective projection from 3D Camera coordinates to 2D image coordinates using the Intrinsic matrix (see Eq. 5.12) was discussed, and Coordinate transformation from 3D World coordinates to 3D Camera coordinates using the Extrinsic matrix 5.18 was shown.



Finally, an expression can be made; The mapping of a point in the World coordinate frame, in pixels, of the Image coordinate frame, which is given by the product of M_{ext} and M_{int}, results in *Projection matrix* P:

$$\tilde{\mathbf{u}} = M_{\text{int}}\, M_{\text{ext}}\, \tilde{\mathbf{x}}_{\boldsymbol{w}} = P\tilde{\mathbf{x}}_w \tag{5.21}$$

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}}_{P} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{5.22}$$

Therefore, to map a point from the World to the Camera coordinate system, one needs to perform a calibration by either finding the twelve unknowns of the projection matrix or the unknowns of extrinsic and intrinsic matrices. Knowledge from this section can also be applied in the later chapters, regarding homogeneous transformations and pose estimations for Cybertech bin picking operations.
In the next section, a look at an implementation method of the computer vision approach to camera calibration will take place, as well as a discussion of possible downfalls and deviations from the theoretical procedure.

5.2 Calibration Implementation

The first step to the precise robotic arm motions controlled by vision is the need for the finest calibration possible, as the visual input will provide the primary source of the robot's surrounding perception, operation state feedback, and meeting the requirements for safety and accuracy demands.

Ideally, the camera should be calibrated only once. However, the nature of an industrial environment imposes challenges that need to be appointed for. Due to the extreme conditions the robots often operate in, the camera system may experience vibrations [BCCD20], leading the lens focus ring to shift. The thermal expansion might degrade the camera mount material, which will need to be replaced, or the vision system will need to be temporarily unmounted for robot maintenance and later be mounted back.

It might seem trivial to manually refocus the lens or mount the camera to the same position as in its previous state. However, a slight hint of a change in any of the variables, such as focal length, focus, principal point, or camera pose, perpetually deviates the actual form from the mathematical model with which the system was calibrated, rendering future operations inaccurate.

Therefore, implementation of the calibration process will be made in such a way as to be reproducible and partially automated, and knowledge gained from Section 5.1 will be used, as well as employment of widely used and open-source computer vision libraries such as OpenCV [Bra00].

5.2.1 The ChArUco Board

Any pattern of known structure and dimensions can be used for camera calibration. In computer vision, this pattern is often a black-and-white pattern, including chessboard pieces, asymmetrical or concentric circles, grids, or dots. Numerous algorithms were developed for feature matching, and calibration board recognition [QLZ10], and the chessboard is believed to be the best ease-of-use to accuracy ratio pattern. As such, a chessboard pattern will be used in this work.

As a reliable and accurate pose estimation method by ArUcO tags[Bra00] will be used during object bin picking, it would be beneficial to incorporate such tags in the calibration process too. Consequently, the ChArUco board, a combination of the chessboard and ArUco tags, was created:

"A ChArUco board is a planar board where the markers are placed inside the white squares of a chessboard. The benefits of ChArUco boards are that they provide both ArUco markers versatility and chessboard corner precision, which is important for calibration and pose estimation." stands in the documentation of OpenCV and directly complies with our requirements for robust intrinsic and extrinsic calibration.

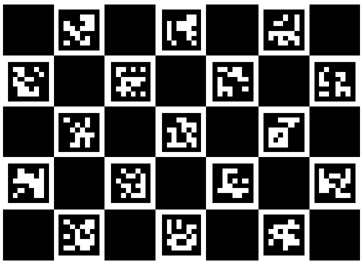


Figure 5.7: CharUco pattern generated in the desired specification for our usage

Through our research and testing of calibration patterns, it was discovered that the ChArUco pattern offers the additional benefit of being easily detectable in slightly blurred images, even when partially occluded or distorted. Moreover, it provides a higher density of calibration points than a standard chessboard pattern. This makes it advantageous for use in calibration, particularly for reliable and accurate pose estimation methods like those used in object bin picking with ArUco tags.

Figure 5.8: CharUco pattern developed on a rigid board

5.2.2 Intrinsic Calibration Implementation

As described in Section 5.1, the goal is to establish the relationship between the coordinates in the camera's image plane and the corresponding 3D World coordinates. Therefore, the parameters defining the camera's internal geometry can now be estimated for this relationship to be determined.
The first step needed is to capture several images of the calibration pattern at different poses of the robot with respect to the calibration pattern, i.e. capturing the pattern from different points of view. The pattern should occupy a significant portion of the image and the features, such as corners and ArUco markers, should be clearly visible.
In order to simplify the calibration process, a GUI was implemented, similar to one in Section 4.2, that displays relevant information for the operator and allows for easy and reproducible execution (see Fig. 5.9).

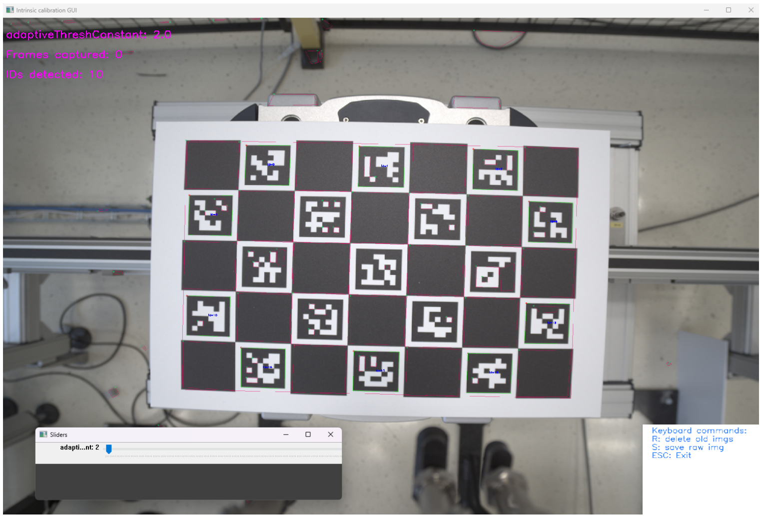


Figure 5.9: Intrinsic calibration GUI

As fine calibration is heavily determined by the number of correctly identified markers and corners that rely on correct static parameters that change depending on the lighting environment and camera focus, a joint parameter slider was created for dynamic alterations for correct identification. Although the steps described above only serve the purpose of saving sample images, the later detection is ensured to be correct, as the same feature detection parameters will be used in the upcoming calibration process.
When enough samples are generated, image features have to be detected for the image coordinates extraction, such as edges, corners, and other interest points.
The corresponding 3D coordinates of the features in the World frame must be also known, and are sought by OpenCV methods `detectMarkers()` and `interpolateCornersCharuco()`.
The final step is to compute and validate the accuracy of the estimated

parameters by projecting 3D points onto the image plane using the estimated parameters and comparing the projected points with the observed image coordinates. This process is aided by OpenCV subpixel refinement preprocessing and `calibrateCameraCharuco()` method. The pseudocode in Algorithm 1 explains the above steps.

```
Data: Require camera communication, cv2, Pattern dimensions, ArUco dictionary
Result: Camera matrix, Distortion coefficients
if camera not connected then
    Connect camera
end
if Data folder doesn't exist then
    Create folder
else
    Delete old data upon warning
end
Start camera grabbing
Open GUI
Create interactive parameter sliders
while Camera grabbing do
    Retrieve results
    convert RAW to BGR
    Update slider Thresh constant
    Detect visible ArUco markers
    Update GUI
    if 's' pressed then
        save RAW to Data folder
    end
    if 'ESC' pressed then
        Close GUI
        Disconnect camera
    end
end
Load ArUco dictionary
Load CharUco dimensions
Create CharUco
Create sub-pixel refinement method
while Frames in Data folder do
    Detect visible ArUco markers
    if Enough markers found then
        Interpolate chessboard corners
        Append detected markers
    end
end
if Enough frames proccessed then
    Calibrate camera
    Acquire Camera matrix, Distortion coefficients Save results
end
```

Algorithm 1: Intrinsic calibration outline

Intrinsic Calibration Validation

Using the datasheet of the vision system [Bas23] and calibration results, the resulting camera matrix 5.23 can be compared and validated with the corresponding values.

$$K = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}, K_{Actual} = \begin{bmatrix} 2.506 & 0 & 2.059 \\ 0 & 2.501 & 1.525 \\ 0 & 0 & 1 \end{bmatrix} \cdot 10^3 \tag{5.23}$$

Based on the information from the manufacturer and the knowledge gained in Chapter 3, it is known that the pixels are square. This is the first deviation from the theory, as the *effective focal lengths* f_x and f_y differ in values.
It was stated in Section 5.1 that in case of equal pixel x and y dimensions, the f_x and f_y should be equal. However, the lens itself may introduce distortion, which can cause the focal length to vary in the x and y dimensions. For example, if the lens has barrel distortion, the image may appear wider in the x-dimension than in the y-dimension, resulting in different focal lengths. Therefore, results can be rendered valid.

Next, the *aperture* value can be computed, which is a measure of the width of the lens opening, and it is usually expressed in terms of the f-number, which is the ratio of the focal length to the aperture. The larger the f-number, the smaller the aperture, and vice versa. To compute the aperture itself, the value of f_x from the calibration matrix is used and multiplied by the width of the pixels in the image sensor, which was obtained from the sensor specifications. As the system has a variable aperture, the computed value of 8.6 mm directly corresponds to a valid value.

Next, the *Skew factor* can be evaluated, which is represented by the second value of the first row of the camera matrix. The skew factor is a measure of the degree to which the image plane is not aligned with the pixel grid. The skew factor is usually small and can be ignored in many cases. However, it can be important in some applications, such as when precise measurements of angles or distances are required in the image. The resulting value 0, therefore, satisfies our requirements.

Finally, the validation of the *principal point coordinates* o_x and o_y can be performed. Despite the absence of this information in the manufacturer's documentation, it can be estimated that the principal point is positioned at the image's center, which would lead to our results being very similar to the optimal values. Further evaluation can be performed by reversed dimension comparison to known object dimensions, such as ArUco tags. Then, the camera matrix and a measurement of an object of known dimensions in the scene can be compared. ArUco tags can be directly used for such measurements, as they are detectable by our method and a GUI for 3D reconstruction and measurements has already been created in Section 4.1. If the principal point coordinates are correct, then the dimensions in the image should be consistent with the expected values.

5.2.3 Extrinsic Calibration Implementation

With regards to Section 5.1 and Section 3.1.1, an extrinsic Eye-in-Hand calibration refers to the process of determining the spatial relationship between the coordinate frame of a vision system and the coordinate frame of a manipulator. Put simply, the goal is to describe the position and orientation of the camera (the "eye") with respect to the robot's end-effector or flange (the "hand"), where the necessary frames are represented in Fig. 5.10.

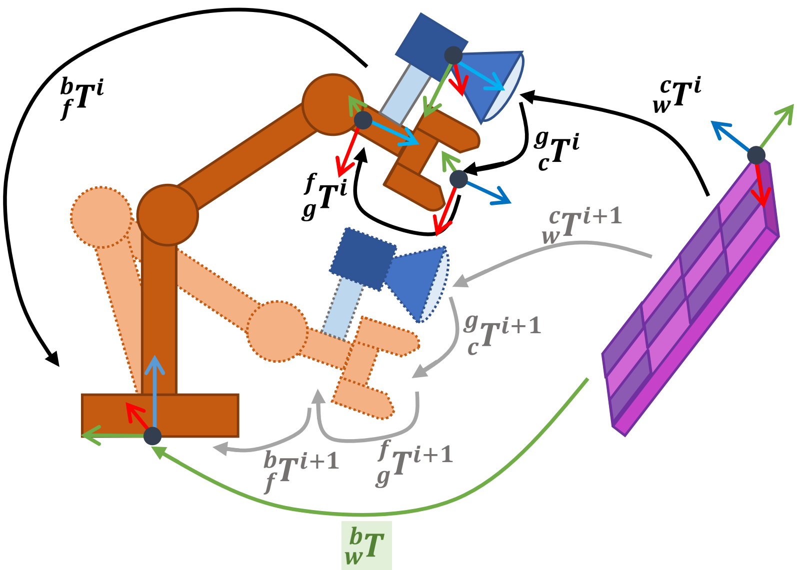


Figure 5.10: Exstrinsic calibration process

Where both in Fig. 5.10 and later in Equation 5.25, $T \in SE(3)^5$ represent the homogeneous transformation matrices [Cra89], as seen in Equation 5.24. The frame of reference is indicated by the left subscript of T, while the target

[5]Special Euclidean group[Sze11]

frame by the left superscript, then, the right superscript is used to indicate the example number, b is the robot base[6], f is the flange, g is the gripper (i.e. tool), c is the camera and w is the World (e.g. calibration pattern or parts). E.g. ${}^{y}_{x}T^{(i)}$ indicates the usage of i-th transformation example from the reference frame x to the target frame y.

$$T = \begin{bmatrix} R_{3\times3} & t \\ 0_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.24}$$

In order to perform the calibration, it is necessary to provide Intrinsic calibration data (see Equation 5.23) and collect a set of correspondences between points in the camera's field of view and the corresponding points in the manipulator's coordinate frame.
For this step, a secondary script utilizing the capture methods from Algorithm 1 was created with ChArUco feature matching and pose estimation extraction, as well as GUI similar to Fig. 5.9.

In addition to the previous Intrinsic calibration steps (see Section 5.2.2), the manipulator's end-effector poses must be extracted in order to calculate the relationship between the camera and end-effector. This was done by establishing a continuous data transfer of the robot's actual pose in the Submit Interpreter `SUB`, as described in Chapter 4.

In theory, the sought relationship can be found by solving the $AX = XB$[7] equation, with reference to Fig. 5.10, library[Bra00], Equation 5.25, and paper [MB23].

$$\begin{gathered} {}^{b}_{g}T^{(i+1)} \cdot {}^{g}_{c}T \cdot {}^{c}_{w}T^{(i)} = {}^{b}_{g}T^{(i+1)} \cdot {}^{g}_{c}T \cdot {}^{c}_{w}T^{(i+1)} \\ \left({}^{b}_{g}T^{(i+1)}\right)^{-1} \cdot {}^{b}_{g}T^{(i)} \cdot {}^{g}_{c}T = {}^{g}_{c}T \cdot {}^{c}_{w}T^{(i+1)} \cdot \left({}^{c}_{w}T^{(i)}\right)^{-1} \\ A_i X = X B_i \end{gathered} \tag{5.25}$$

Using the GUI, camera feed, and robot control implementation discussed in Chapter 4, a semi-automatic scanning operation for the calibration pattern was developed to ensure reproducible results, as discussed in Section 5.2 (see Algorithm 2 and Algorithm 3). Further analysis of convention unification

[6] Sometimes referred to as $NULLFRAME in Cybertech

[7] A, X, and B are homogeneous transformations. A represents the transformation from the World frame to the camera frame, B represents the transformation from the robot base frame to the end effector frame, and X represents the sought transformation from the camera frame to the end effector frame

used by OpenCV and KUKA will be presented in Section 6.1.
The resulting homogeneous transformation matrix corresponds to ${}^{f}_{c}T$ and can be further used within controlled-based transformations to retrieve ${}^{g}_{c}T$ in Fig. 5.10.
The results can be partially validated by comparing the translation component of calibration results and the 3D layout of known spatial relationships of the camera, camera holder, and end-effector (see Fig. 5.11) with a deviation in the unknown location of estimated optical center in terms of millimeters.

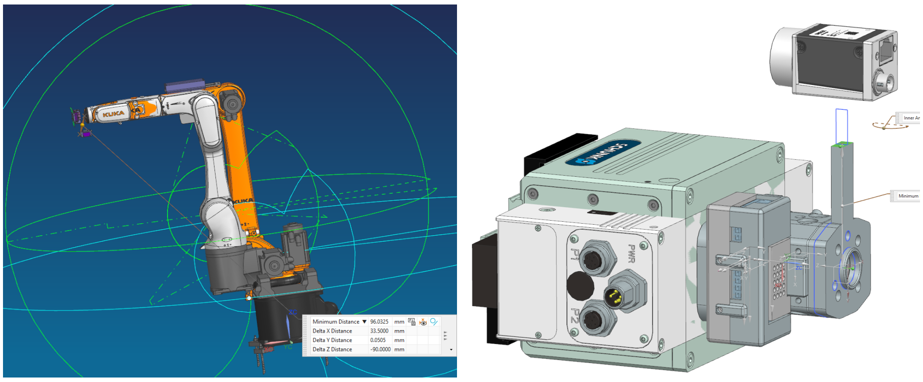


Figure 5.11: Extrinsic calibration results validation in CAD layout

```
Require : camera class, cv2, pattern dimensions, intrinsic data
Result  : spatial relationship camera to flange
if camera not connected then
|   connect camera;
end
StartCameraFeed();
OpenGUI();
RequestChangePGNO();
while CameraGrabbing and RobotNotDone do
|   ReadRobotPose();
|   if TakePicture then
|   |   DetectCalibrationPattern();
|   |   EstimatePose();
|   |   SendCameraDone();
|   end
end
CalibrateEyeInHand();
SaveResults();
SendCalibrationDone();
```

Algorithm 2: Extrinsic calibration process Python side

```
Require : calibration PGNO, Safety ACKN
SendRobotPose() <--Thread-- SUBMIT INTERPRETER;
while not CalibrationDone do
|   movePTPcalibration();
|   SendTakePicture();
|   WaitForCameraDone();
end
```

Algorithm 3: Extrinsic calibration process KRL side

Chapter 6

Final Deployment and Validation

The previous chapters have covered the development and implementation of various components of the Flexible assembly line's Cybertech warehouse robot, including vacuum gripper, GUI, and camera calibration.
With these pieces in place, the final step is to implement a Mission control for bin picking operations on new parts, deploy the robot on the assembly line and validate its performance.

This chapter will therefore cover the final Mission control implementation, deployment, and testing of the robot, including any challenges encountered during the process and how they were addressed. The results of the validation process will be presented, along with any recommendations for further improvements or future work.

6.1 Convention unification

As hinted in Section 5.2.3, a challenge was encountered during the implementation of the Extrinsic calibration that could persist in bin picking operations. The issue was due to inconsistencies in the conventions used by the methods for pose estimation in [Bra00] and those utilized by Cybertech for the robot's actual pose registration. To address this, it was necessary to unify the conventions employed by both systems.
As the said pose estimation function is returning the poses of the calibration pattern (and to-be-picked parts) in fixed-axis rotation vectors `rvec`, the con-

version to a rotation matrix, and vise versa, is achieved by the Rodrigues transform and its inverse:

$$\begin{aligned} &\theta \leftarrow \text{norm}(rvec) \\ &r \leftarrow rvec/\theta \\ &R = \cos(\theta)I + (1-\cos\theta)rr^T + \sin(\theta)\begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \end{aligned} \tag{6.1}$$

In contrast, the poses ${}^{b}_{f}T^{(i)}$ yielded by Cybertech (described in Section 4.3.1 and Fig. 5.10) are presented in the format:

$$XYZABC \tag{6.2}$$

where XYZ represents the translation vector t and ABC denotes the Euler notation in a sequential combination of $R_z - R_y - R_x$[1] rotations about axis z, y, and x. For the conversion of the rotation part of the Cybertech pose to be unified with the notation of OpenCV, rotation matrices must be composed in the correct sequence, using the right-hand rule [FLS10]:

$$\begin{aligned} &\theta \leftarrow \text{radians}(A/B/C) \\ &R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \\ &R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \\ &R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &R = R_z R_y R_x \end{aligned} \tag{6.3}$$

Upon composition of the transformation matrices T, computation of Equation (5.25) can be performed by OpenCV's `CalibrateHandEye()` methods by a fully autonomous and efficient calibration technique [TL89] and the knowledge gained can be further used in the bin picking operations.

[1]Technically, this is a Tait-Bryant notation, rather than a strict Euler notation

6.2 The Mission Control

Control Options

The method proposed in this work provides two control possibilities. The first option is ideal for assembly line's innovation processes, debugging and includes a camera feed, tracking visualizations, status log, and intermediate 3D transformations (developed with [Hun07]), along with a semi-autonomous button-driven mode or the ability to add a completely new part within seconds. This facilitates the testing of the method's functionality, whether it is for new product development or a new robot deployment within the Flexible assembly line.
As Cybertech is expected to perform autonomous bin picking dependent on MES requests, a MES simulation ("mocking") script was created in such a way, that the GUI buttons (described in Fig. 4.2) do not interfere with the Mission control script directly, but only change appropriate OPC UA interface values, as MES would in real operations. Therefore, the robot is allowed the movements and computer vision-based operations under operator request, while the continuity and integration of MES is still maintained.

Figure 6.1: Visualization of adding a never-seen-before part

Then, a new part can be easily added to the Mission Control script by part specification i.e. name, Tag ID, and dimensions:

```
    get_tag_name =  24: "controller_left",
                    25: "controller_right"
    def get_tag_dimensions(tag_id):
        controller_left = 16.5 #[mm]
        controller_right = 16  #[mm]
        if tag_id == get_tag_id["controller_left"]:
            return controller_left
        if tag_id == get_tag_id["controller_right"]:
            return controller_right
```

The second control option is fully autonomous and integrated with MES methodologies, such as requests for object bin picking from a warehouse AGV to an assembly line's distribution shuttle and vice versa. The only pre-requisite for full autonomy is the requirement of an array of all parts, an initial pose, from which the scanning operation starts, and pre-defined place coordinates.
Then, any Cybertech assembly line operation is performed as follows, chronologically:

- Movement in the vicinity of inbound parts
- Scanning operation for parts
- Part selection and tracking
- Align movement
- Pick movement
- Place movement
- Home movement

Target Transformation

The following pseudocode outlines the transformation calculations utilized within the method for achieving a generalized target pose[2] ${}^{b}_{w}T$, applicable to a range of robotic systems, including but not limited to KUKA robots.

$$
\begin{aligned}
{}^{b}_{f}R, t_f &\leftarrow \text{GetRobotPoses}() && (6.4)\\
{}^{b}_{f}T &\leftarrow \text{ComposeT}({}^{b}_{f}R, t_f) && (6.5)\\
{}^{f}_{c}R_{orig}, t_c &\leftarrow \text{GetExtrinsicCalib}() && (6.6)\\
{}^{f}_{c}R &\leftarrow \text{Rodrigues}({}^{f}_{c}R_{orig}) && (6.7)\\
{}^{f}_{c}T &\leftarrow \text{ComposeT}({}^{f}_{c}R, t_c) && (6.8)\\
K &\leftarrow \text{GetIntrinsicCalib}() && (6.9)\\
{}^{c}_{w}R, t_w &\leftarrow \text{GetPartPose}(K, \texttt{distCoeffs}) && (6.10)\\
{}^{c}_{w}T &\leftarrow \text{ComposeCybertechT}({}^{c}_{w}R, t_w) && (6.11)
\end{aligned}
$$

[2]Where ${}^{b}_{w}T$ corresponds with Fig. 5.10, but now w represents the to-be-picked part.

Where:

- (6.4) is ensured by the robot, through the use of a `SUB`[3] and OPC UA.
- (6.5) composition uses the knowledge from Section 5.1.
- (6.6) function returns raw extrinsic calibration data from Section 5.2.3.
- (6.7) prepares the R by Rodrigues transform from Equation 6.1.
- (6.8) composition uses the knowledge from Section 5.1.
- (6.9) function retrieves intrinsic calibration data from Section 5.2.2.
- (6.10) uses pose estimation technique `estimatePoseSingleMarkers()` [Bra00].
- (6.11) composes target transformation for Cybertech by Convention 6.2.

Finally, target pose estimation in the robot's coordinate Base frame is evaluated and the robot is ready for Pick operation, or Approach operation if deemed necessary:

$$ {}^{b}_{w}T = {}^{b}_{f}T \cdot {}^{f}_{c}T \cdot {}^{c}_{w}T \tag{6.12} $$

The overall Mission control outline, including the involvement of MES requests and replies, such as tool checking, GUI initialization, or safety requirement acknowledges, is shown in Fig. 6.2. The bin picking operations follow a 'First look, then move' strategy, where the Approach movement is repeated before the Pick operation in case the, later described, tracker detects any deviation in a statically aligned part during robot movements (i.e. pose estimation error).

Part Tracker

Working with ArUco tags can be challenging when dealing with multiple parts of the same type and tag IDs, as the open-source library [Bra00] was not designed to handle such scenarios.
Furthermore, if the visibility of a part is momentarily lost due to robot movements (i.e. camera movements and its subsequent image blur), another

[3] Submitter interpretter (see Chapter 4)

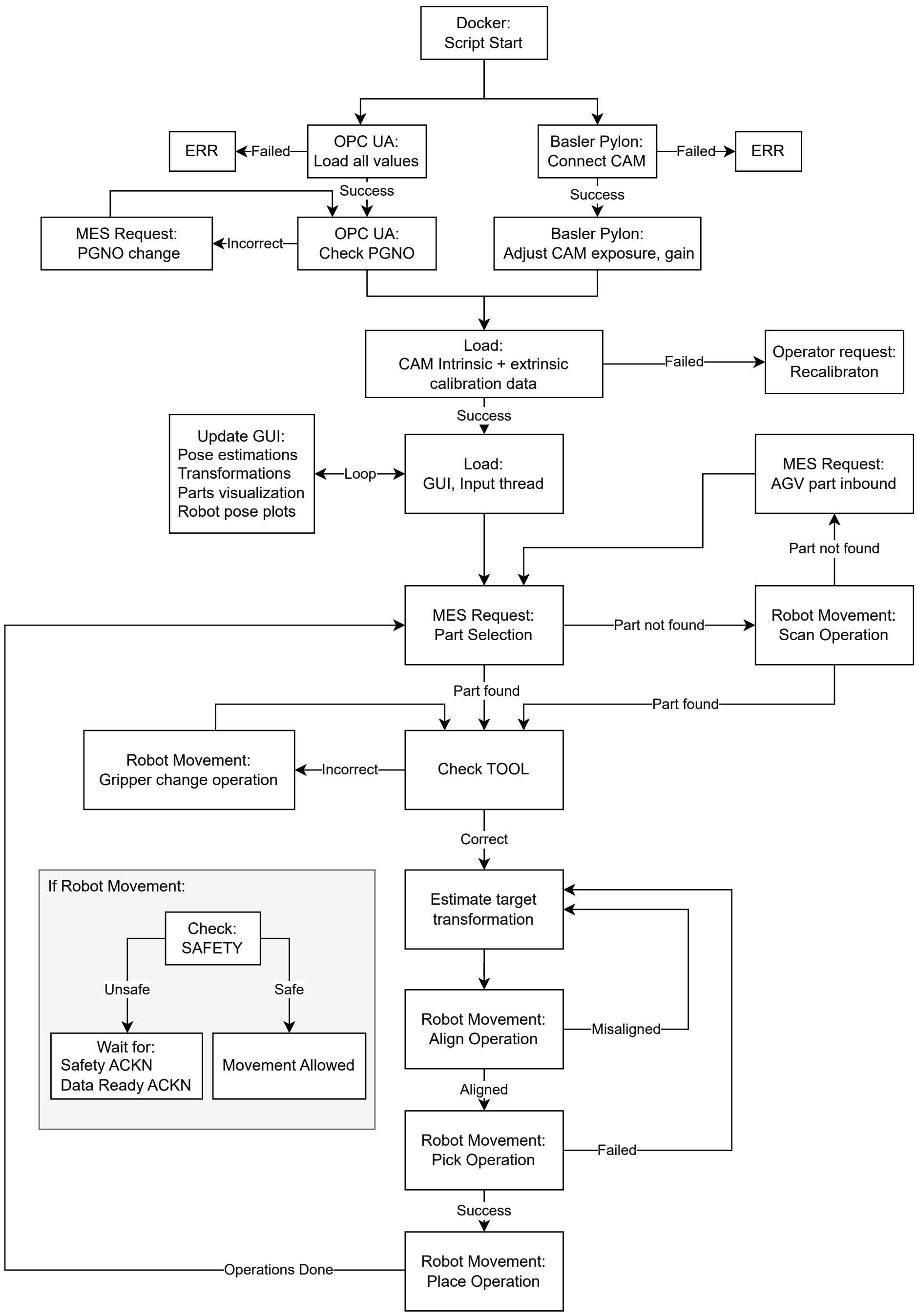


Figure 6.2: Python Mission Control script

part of the same type may be selected for an upcoming approach iteration (see Fig. 6.2). To tackle this issue, a part tracker script was developed. The script uses a 3D re-projection of a previous pose estimation and repeatedly attempts to re-localize the part within a predefined vicinity radius.
This ensures that the most appropriate part is selected and tracked, even if its visibility is lost during a robot's approach operation. By employing this

script, the system can ensure accurate and efficient picking of the desired part, even in complex scenarios where multiple parts of the same type are present.

6.3 Script Structure

As previously contextualized in earlier chapters and partially depicted in Figure 6.2, Equations 6.8 to 6.11 and Algorithm 1, and Algorithm 2, the Mission control, GUI, and MES simulation and calibrations for both semi-autonomous and fully autonomous operations are implemented using the Python programming environment on a remote computing device (i.e. not robot-based) and it's structure is visualized in Figure 6.3.

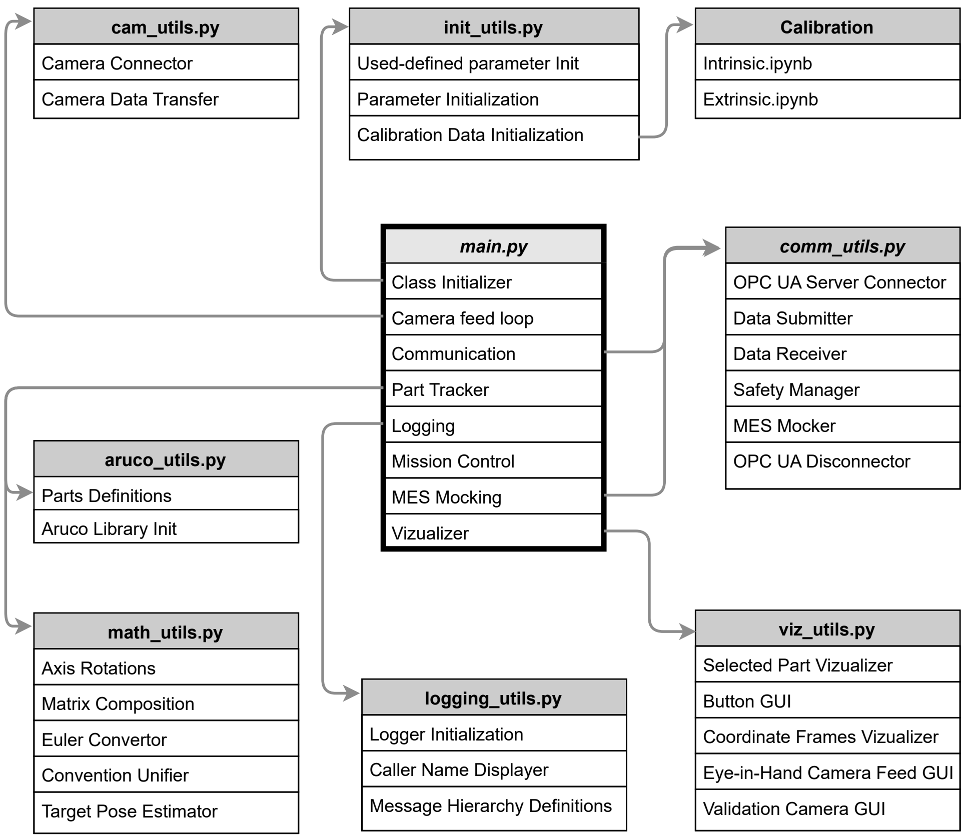


Figure 6.3: Python Control Script Hierarchy

Similarly, the robot's part script was partially depicted in SUB data transmission in Figure 4.11, and its use in Algorithm 3 and Figure 6.2. Then,

Figure 6.4 gives a coherent representation of initialization and sequential operations used, in hierarchical order.

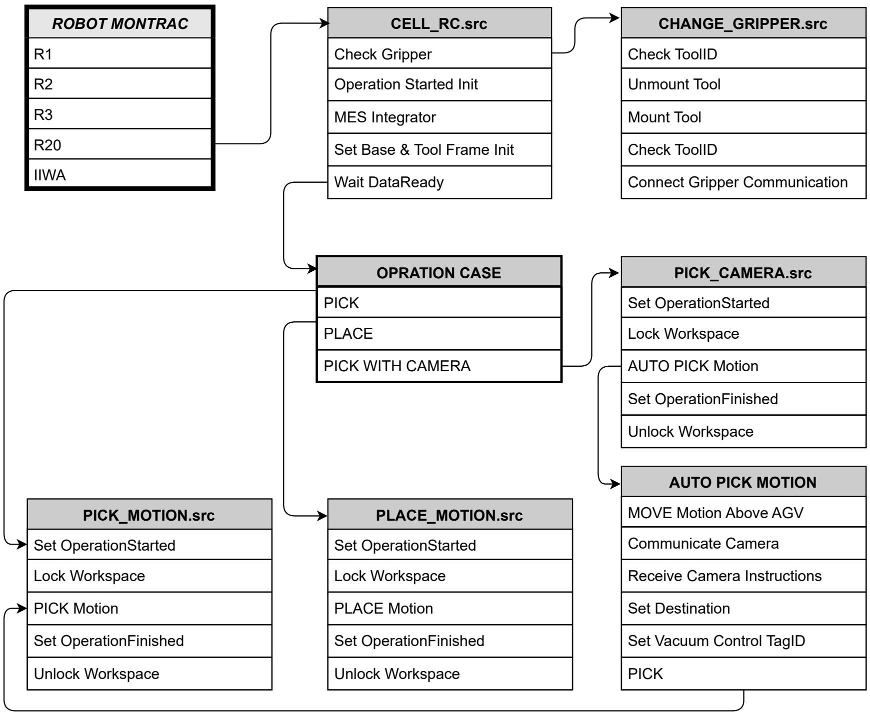


Figure 6.4: Robot hierarchical Mission control functionality

6.4 AGV Frame Calibration

The autonomous Pick & Place system, which is extensively described in the thesis, utilizes a methodology of calibrating each part's coordinate frame for accurate autonomous picking. This approach has proven to be highly effective and can be extended to other scenarios as well.

In particular, let's consider the inbound AGV shown in Figure 6.5 that transports parts from the warehouse. The AGV itself is subject to an error during the delivery and placement of the warehouse shelf, estimated to be within a range of ± 3 mm. The Eye-in-Hand approach employed by Cybertech for autonomously picking up parts compensates for this error.

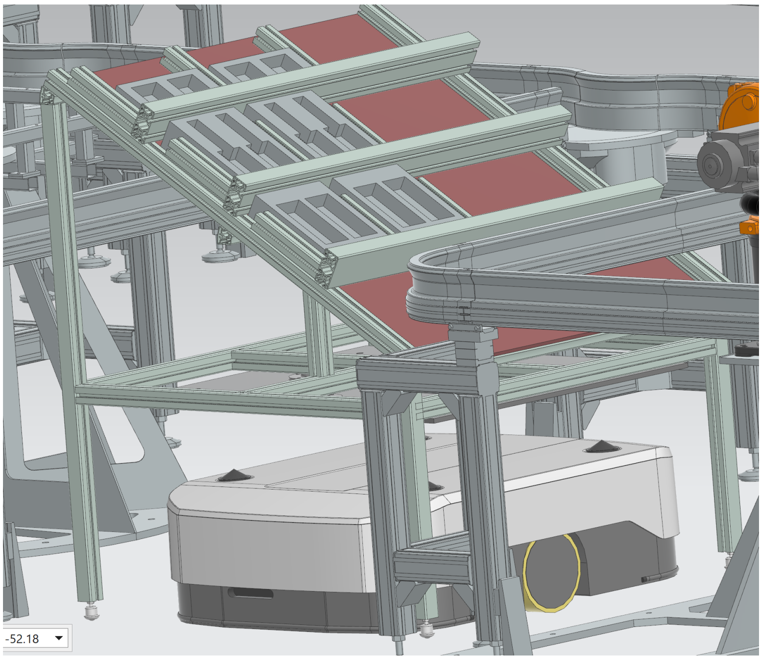


Figure 6.5: Inbound warehouse AGV showcasing the potential error in the placement of the warehouse shelf within the XY plane.

However, there are certain parts, such as legacy products from previous projects, that cannot be autonomously picked and rely on strict Tool-Base frame definitions. In the event that the warehouse shelf experiences movement due to the aforementioned error, the picking operations for these parts would be affected.

To address this challenge, the methodology described in the thesis can be extended to calibrate the shelf itself. By placing ArUco tags at strategic locations and employing the approach outlined in Section 6.2, the shelf can be calibrated each time it is moved. Furthermore, this calibration process can be automated so that when an AGV arrives, Cybertech's system automatically calibrates the rig's position, ensuring the accuracy and reliability of the picking operations for those parts that are not picked based on computer vision.

6.5 System Validation

The system is now fully set up and ready for validation.
To recapitulate, the robot's task is to autonomously pick up components from a warehouse shelf using the methodology described in Mission control, as well as the transformations utilized in extrinsic calibration in coherence with the Manufacturing execution system needs. The robot will collect the required components for assembling a complex RC car, as depicted in Figure 6.6, and broadcast its mission status for further assembly processing.

During the validation process, the system will be thoroughly tested to ensure its reliability and effectiveness. Any issues that arise during this phase will be documented and addressed to guarantee that the system functions optimally. For instance, the system was enhanced to include the capability of recovery after any safety requirements disturbance, such as the entry of a person into the robot's workspace. Additionally, the system was redesigned to allow for the addition or movement of parts during an already-running operation.

6.5.1 Proof-of-Concept Validation

In order to validate the functionality and proof-of-concept presented in the preceding sections, an initial test of the proposed approach was performed. This involved repeatedly picking a single part from an AGV station in various positions and orientations.
To achieve this, the end-effector was manually moved to match the pose of the part for each run iteration (i.e. set the ground truth) and subsequently autonomously picked the part from a randomized initial pose of the robot. The ground truth target frames are depicted in Figure 6.7 and Table 6.1, along with their estimated values using the approach presented in this work.

The proof-of-concept testing has demonstrated that the proposed method-

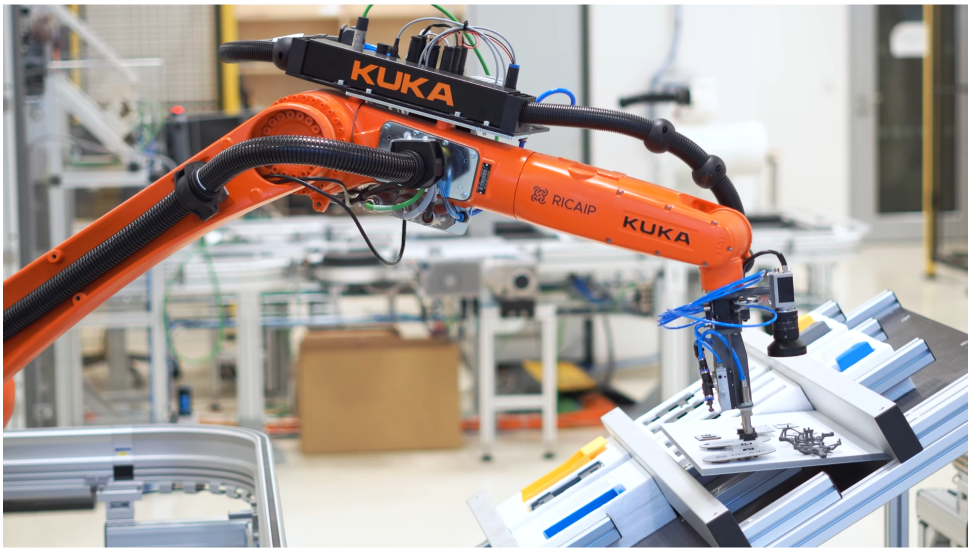


Figure 6.6: Cybertech performing autonomous picking operation from inbound warehouse AGV

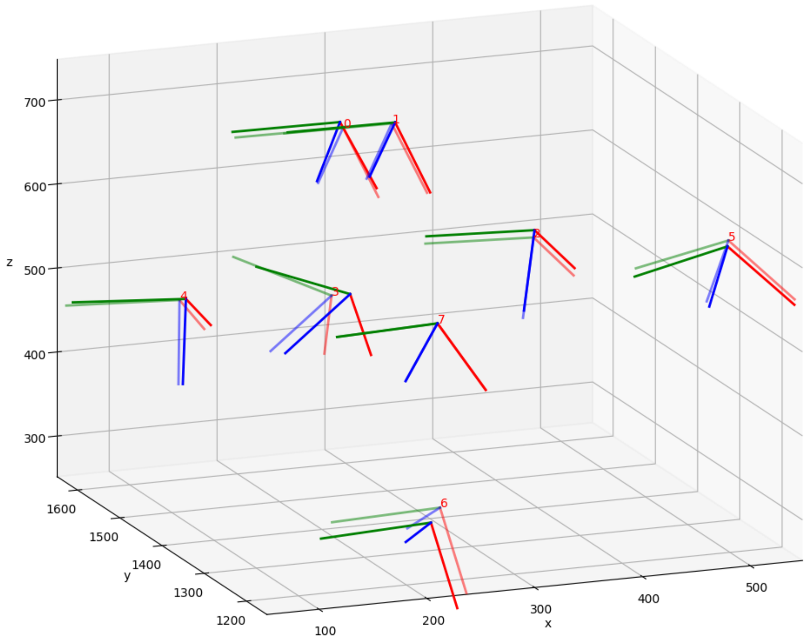


Figure 6.7: Ground truths (solid) and estimation visualizations (semi-transparent) of an end-effector target frames (i.e. the position of parts)

ology has the potential to be applicable to the task at hand, although Table 6.1 and Fig. 6.7 show that the errors obtained are not suitable for accurate bin picking operations.
As a result, various modifications were implemented.

Run	ΔX (mm)	ΔY (mm)	ΔZ (mm)	ΔA (deg)	ΔB (deg)	ΔC (deg)
0	0.47	8.61	3.78	0.07	-3.23	0.04
1	2.21	-2.48	1.80	0.02	0.72	-0.01
2	0.45	-1.11	8.97	0.00	0.00	0.00
3	19.47	5.67	-2.60	15.18	2.87	3.20
4	7.41	3.69	0.90	0.07	2.11	1.47
5	-4.62	-9.65	-3.24	-0.34	-0.35	-2.08
6	-14.66	-16.97	-10.66	0.05	0.16	-3.99
7	0.03	0.10	0.04	0.01	0.00	0.00

Table 6.1: Errors in X, Y, Z, A, B, C for each run.

For instance, a rigid board with a printed calibration pattern was created, and intrinsic & extrinsic calibrations were performed again. Moreover, Basler sharpness indicator [Bas23] was utilized to estimate the optimal approach distance, and a polarizing filter was attached to the lens to eliminate undesired reflections. Additionally, sub-pixel detection for pose estimation was introduced to enhance the precision of the results, with a goal of sub-millimeter accuracy. Upon these modifications, the step of final validation is ready to be performed.

6.5.2 Final Validation

As stated in [MB23], to verify the proposed strategy, the robotic arm is first moved to a position where a shelf, table, or a similar structure containing parts is expected to be positioned before every Pick & Place operation.
The parts are distributed in a range of 1 to 1.5 meters from the ground and at the range, roughly equivalent to 30° of the possible manipulator's base rotation, and 160 mm to 800 mm from the robot (i.e. the distance along the Z-axis of the camera coordinate frame).
The parts are not contained in any form of holder and are allowed movement. Parts are rotated in yaw, pitch, and roll in the range of $\pm 45°$. The parts are then picked and placed to a predefined coordinate on a planar assembly line shuttle, as seen in Fig. 6.8, where a second camera is positioned and measures the deviations in X, Y, Z, and A (Yaw) axis from the ground truth placement, defined by manual Pick & Place operation from, and to, strictly defined coordinates, estimated by a calibrated static pose estimation method [Bra00]. An unlimited quantity of parts of several types is allowed to be visible. Regarding the limitations of the method, stacking the parts of the same type on top of each other is not feasible, as the underlying object could be mistakenly tracked. Then, due to the specific implementation of the part tracker (described in Section 6.2), it is necessary to keep the parts separated from each other by at least 2.5 cm.
In the processing pipeline, several errors can be introduced, such as in:

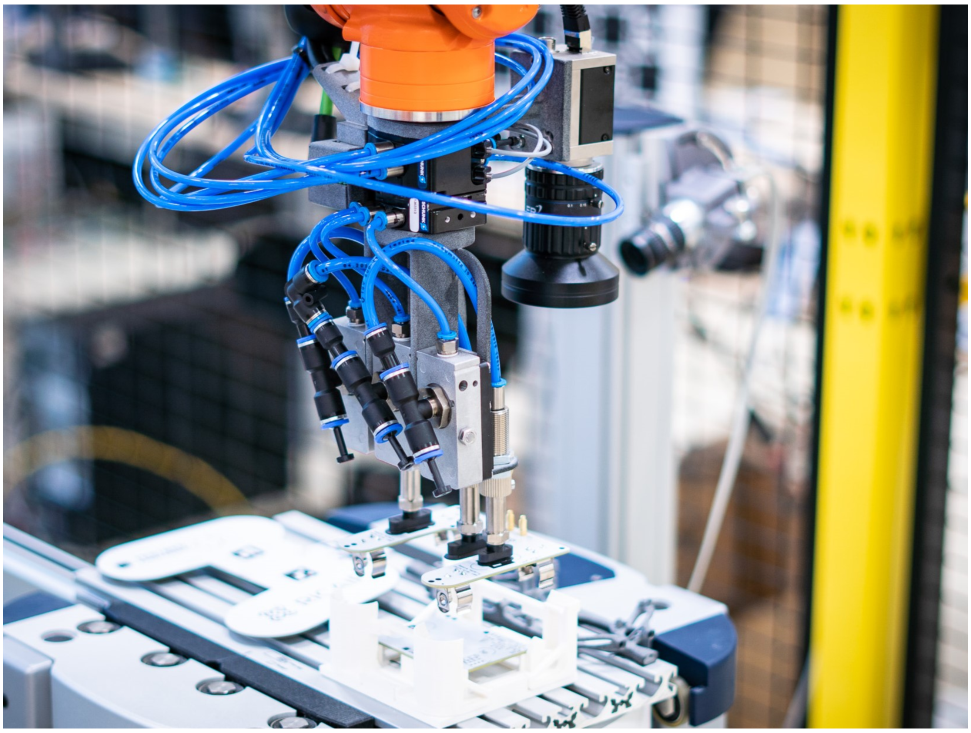


Figure 6.8: Place operation on one of the autonomously picked parts, utilizing an Eye-in-Hand system (camera on the robotic arm) and a visual validation system (camera in the background)

- Intrinsic calibration
- Extrinsic calibration
- TCP calibration
- Pick pose estimations
- Robot accuracy

A set of 50 experiments was conducted with various parts and are visualized in Fig. 6.9. No outlier was detected or removed from the observations. The occurrence of the robot failing in fulfilling the Pick operation was observed once due to the reflection of a part from all scanning angles. The error of the pose estimating validation system was estimated to be below the robot's immeasurable motor encoder capabilities and thus not covered in the results. Upon converting the measurements of the relative spatial relationship of the validation system and the placed part from the ground truth, it can be concluded that the Euclidean distance deviation is ±0.7888 mm and virtually no error in the measured yaw rotation. The measurements and errors can be visualized with a reference to a 1 euro cent in Fig. 6.10.

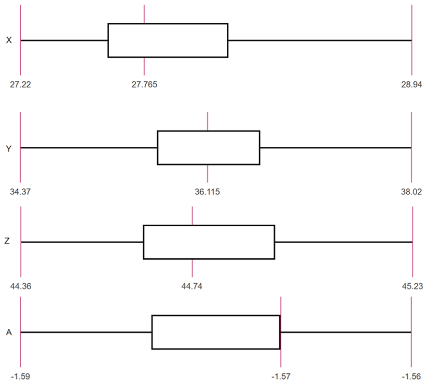


Figure 6.9: Box plots of relative validation errors in X, Y, Z [mm] and A (yaw) [deg], the whisker symbolizes the ground truth

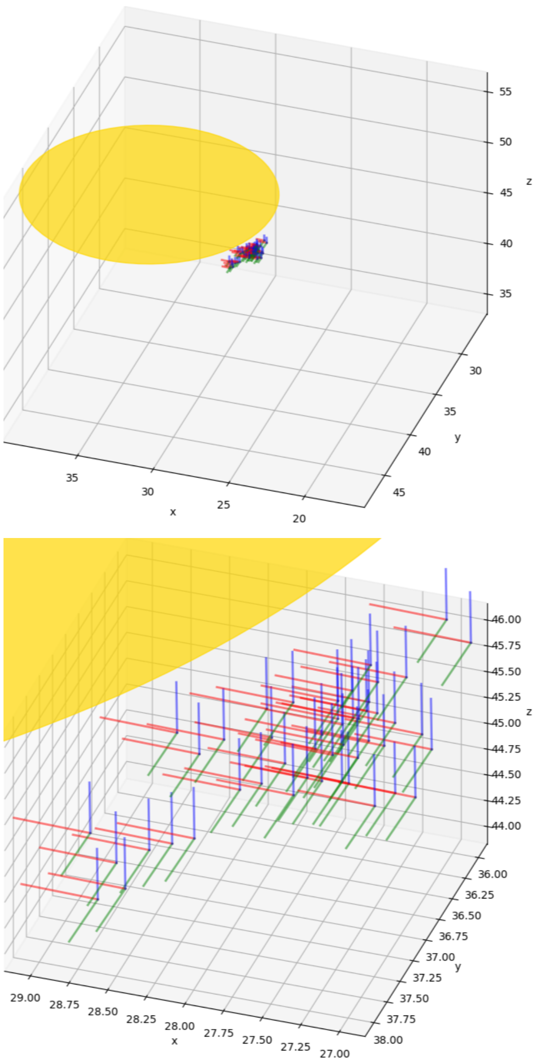


Figure 6.10: 3D plot of pose estimation validations with a yellow circle as a reference to a 1 euro cent.

Chapter 7

A Summary of Findings and Recommendations for Further Research

As presented in [MB23], addressing the challenge of industrial and robust bin picking can be approached through various methods. Bin picking systems that use Eye-to-Hand systems (see Fig. 3.1) either by generating a 3D point cloud of parts to be picked [PKSC17] or Neural Network-based servoing [TAK21] can be utilized, although they may not be ideal for densely populated assembly lines, such as in the case with the Flexible assembly line described in Chapter 2. The Eye-to-Hand vision system can be hindered by the robotic arm, and vice versa, thus creating obstruction issues. Furthermore, the robot can only use vision-based operations in the region of the camera field of view, further reducing adaptability and applicability.
Therefore, the Eye-in-Hand methodology was selected for this work in Chapter 3, as well as the selection of the vision system, and emphasis was put on the significance of selecting the appropriate system in the early stages of the project.

A fully integrated method was developed, drawing upon the knowledge gained from Chapter 4 and Chapter 5. This method demonstrated the obsolescence of pre-defined coordinate part placements by utilizing autonomous warehouse vehicles to deliver new parts to the robot in randomized orders, placements, and quantities. The autonomous picking of these parts was achieved successfully. The implementation of this scenario was presented in Chapter 6, with the highlight of the potential for calibration of the entire warehouse shelf to meet the requirements of other projects that cannot use computer vision-based servoing. The experiments revealed high accuracy in autonomous scanning and picking operations, even when dealing with parts of various shapes and sizes. Additionally, the method utilized controller-based Flange-Base transformations, therefore allowing the method to be applied to a wide variety of robots that are capable of utilizing computer vision throughout their entire workspace. Moreover, the camera mounted on the robot's arm provided constant visual feedback to the operator, as well as operation status feedback. Similarly as in [TH18], the experiments demonstrated precision within a millimeter range, but this work managed to retain the Flange-Base transformations controller-based, thus allowing for the implementation of the introduced method in a broader spectrum of systems.

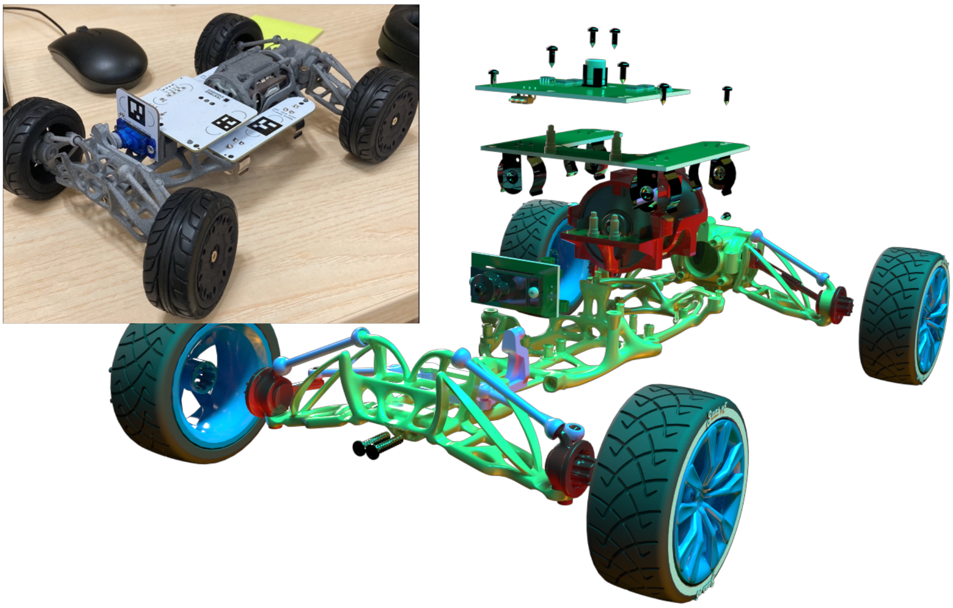


Figure 7.1: Remotely-Controlled Car Assembly

7.1 Future Considerations

Neural Net Pose Estimation

A graphical illustration of a scenario in which the feature matching algorithm results in severely erroneous detection is depicted in Figure 7.2. The incorrect initial detection leads to inappropriate transformations and target destinations for the robot. While fulfilling other prerequisites for the approach, such as the presence of a visible tag, stable pose estimation, and accurate transformations, the robot is ready for a pick operation and may collide in the approach movement due to this erroneous occurrence. Although this is extremely rare, the issue has not been addressed, as it is represented by the feature-matching algorithm performed in the OpenCV pipeline. Therefore, consideration of pose estimation subsidized by a Neural Network shall be made.

Figure 7.2: Invalid Part Pose Estimation

Gripper Feedback

As explained in Chapter 4, the developed gripper is fitted with pressure sensors. Utilizing them, such as in the Pick operations by direct servoing in the Z-axis linear movement, would be beneficial to ensure precise picking and avoid overshooting in the Z-direction.

Campus 5G Camera Data Transfer

Thanks to Testbed for Industry 4.0's collaboration with leading partners in the field of telecommunication, the facility is equipped with a state-of-the-art

Private 5G Campus Network Core. The application of this technology in camera data transfer could prove advantageous, as it has the potential to increase FPS[1] and enable remote and wireless visual feedback in the GUI.

7.2 Broader Implications

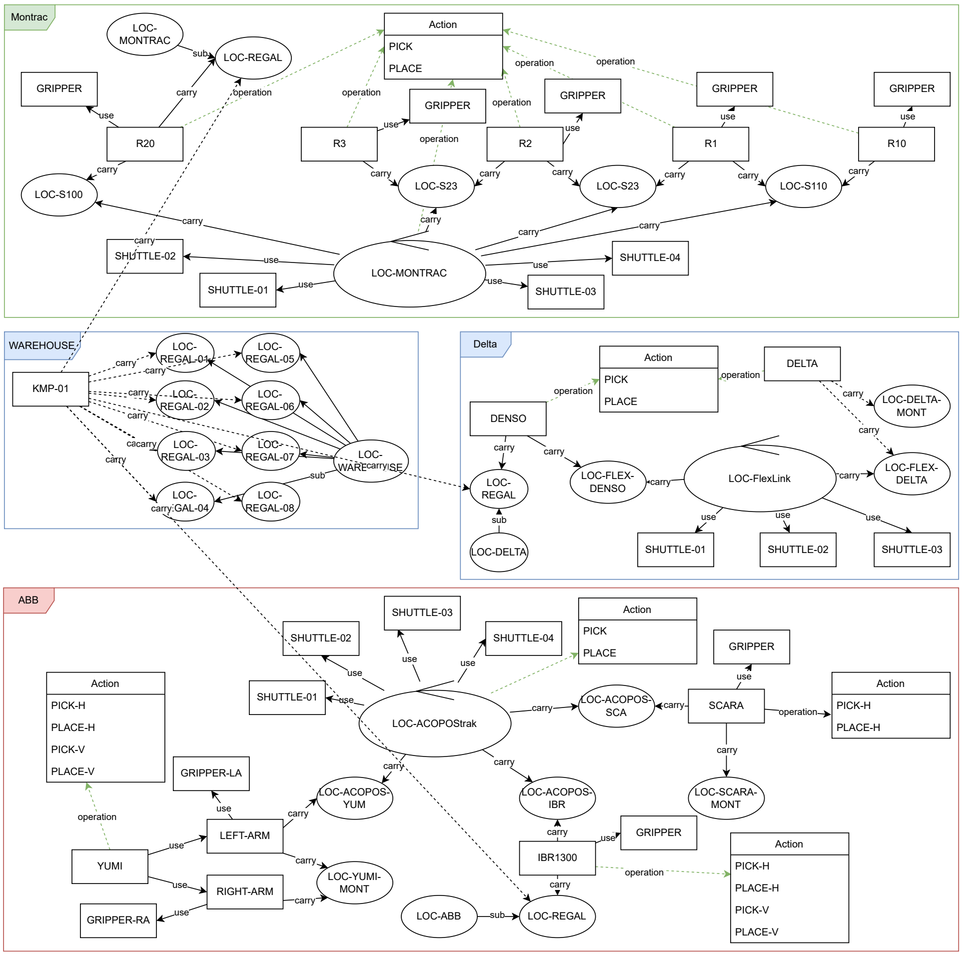


Figure 7.3: Car Assembly Distributed production

As mentioned throughout the thesis, the Cybertech project has a direct impact on a larger-scale milestone.
Although the broader implications of this project are outside the scope of this thesis, it is worth noting that Figure 7.1 provides a reference to the assembly of the final product of the new product and helps to contextualize the mission for Cybertech on the Flexible assembly line.

[1] Frames per second

The successful implementation of the proposed method warrants the consideration of the final stages of the project, where Fig. 7.3 provides a glimpse of the deployment of the assembly in a distributed manner, with visible computer vision-enabled Cybertech robot on the Flexible assembly line ("Montrac") coordinated with the Delta station and ABB assembly line, as well as AGV preprocessed part distribution. This potentially paves the way for the method's deployment in RICAIP partnering facilities, such as in the Central European Institute of Technology (CEITEC), the German Research Center for Artificial Intelligence (DFKI), or the Center for Mechatronics and Automation Technology (ZeMA).

Figure 7.4: The author, KUKA Cybertech & the Flexible assembly line

Appendix A

Funding Acknowledgements

This work was supported by Cluster 4.0, Methodology of System Integration, project number EF16_026/0008432, funded by the Ministry of Education, Youth and Sports of the Czech Republic, Operational Program Research, Development, and Education, and is part of the Research and Innovation Centre on Advanced Industrial Production (RICAIP) project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 857306.

Appendix B

Bibliography

[ABV20] Udayanto Dwi Atmojo, Jan Olaf Blech, and Valeriy Vyatkin, *A plug and produce-inspired approach in distributed control architecture: A flexible assembly line and product centric control example*, 2020 IEEE International Conference on Industrial Technology (ICIT), 2020, pp. 271–277.

[AG23] KUKA AG, *Kuka.workvisual software*, kuka.com (Accessed online, 2023).

[Bas23] Basler, *Basler product documentation*, Accessed online, 2023.

[BCCD20] Matteo Bottin, Silvio Cocuzza, Nicola Comand, and Alberto Doria, *Modeling and identification of an industrial robot with a selective modal approach*, Applied Sciences **10** (2020), no. 13, 4619.

[Bra00] G. Bradski, *The OpenCV Library*, Dr. Dobb's Journal of Software Tools (2000).

[CFH92] P Chutima, IS Fan, and SJ Harrington, *Flexible robotic assembly control systems*, 1992 Third International Conference on Factory 2000,'Competitive Performance Through Advanced Technology', IET, 1992, pp. 71–76.

[Cor23] Unified Architecture Core, *Ua opc 10000-1 ua part 1: Overview and concepts, ver. v 1.05.02*, opcfoundation.org (Accessed online 2.2023).

[Cra89] J. Craig, *Introduction to robotics: Mechanics and control. 2nd ed.*, Addison-Wesley, 1989.

[Dou22] P. Douda, *Implementation of a distributed mes in testbed for industry 4.0*, Dspace CTU (2022).

[FLS10] Richard Phillips Feynman, Robert Benjamin Leighton, and Matthew Sands, *The Feynman lectures on physics; New millennium ed.*, Basic Books, New York, NY, 2010, Originally published 1963-1965.

[HGJD08] Ciaran Hughes, Martin Glavin, Edward Jones, and Patrick Denny, *Review of geometric distortion compensation in fish-eye cameras*, IET Irish Signals and Systems Conference (ISSC 2008), 2008, pp. 162–167.

[Hun07] J. D. Hunter, *Matplotlib: A graphics environment.*, 2007.

[IL23] IO-LINK, *Io-link system description*, io-link.com/ (Accessed online 2.2023).

[JKO18] Václav Jirkovský, Petr Kadera, and Marek Obitko, *Opc ua realization of cloud cyber-physical system*, 2018 IEEE 16th International Conference on Industrial Informatics (INDIN), 2018, pp. 115–120.

[Jí22] L. Jílek, *Production implementation on a robotic assembly cell*, Dspace CTU (2022).

[KRL23] KUKA KRL, *Kuka robot language kr*, https://www.kuka.com/ (Accessed online 2.2023).

[KSKP21] Viktor Kozák, Roman Sushkov, Miroslav Kulich, and Libor Přeučil, *Data-driven object pose estimation in a practical bin-picking application*, Sensors **21** (2021), no. 18, 6093.

[MB23] M. Mikšík and P. Burget, *Adaptive industrial robotics: Eye-in-hand approach for precise pick and place operations*, IEEE-IROS, Detroit, US, In Review, 2023.

[MFR+22] M. Mikšík, H. Fjellheim, R. G Revilla, L. Malek, and H. I. Abdalla, *Real-time stereo object tracking and classification project report*, Danmarks Tekniske Universitet, Perception for Autonomous systems, 5 2022.

[NDKV22] Petr Novák, Petr Douda, Petr Kadera, and Jiří Vyskočil, *Pymes: Distributed manufacturing execution system for flexible industry 4.0 cyber-physical production systems*, 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2022, pp. 235–241.

[NVW20] Petr Novák, Jiří Vyskočil, and Bernhard Wally, *The digital twin as a core component for industry 4.0 smart production planning*, IFAC-PapersOnLine **53** (2020), no. 2, 10803–10809, 21st IFAC World Congress.

[PKSC17] Ales Pochyly, Tomas Kubela, Vladislav Singule, and Petr Cihak, *Robotic bin-picking system based on a revolving vision system*, 2017 19th International Conference on Electrical Drives and Power Electronics (EDPE), 2017, pp. 347–352.

[PRO23] PROFINET, *Process field network*, siemens.com (Accessed online 2.2023).

[PS06] Peter Peer and Franc Solina, *Where physically is the optical center?*, Pattern Recognition Letters **27** (2006), no. 10, 1117–1121.

[QLZ10] Wang Qi, Fu Li, and Liu Zhenzhong, *Review on camera calibration*, 06 2010, pp. 3354 – 3358.

[RA04] Hartley Richard and Zisserman Andrew, *Two-view geometry*, Multiple View Geometry in Computer Vision, Cambridge University Press, March 2004, pp. 237–238.

[SAHM22] Ahmad Sajjad, Wasim Ahmad, Salman Hussain, and Raja Majid Mehmood, *Development of innovative operational flexibility measurement model for smart systems in industry 4.0 paradigm*, IEEE Access **10** (2022), 6760–6774.

[Sas23] M. Saska, *A quickstart guide to camera and lens selection for computer vision*, Multi-robot Systems Group, CTU (Accessed online, 2023).

[Sie23] Siemens, *Totally integrated automation portal*, siemens.com (Accessed online, 2023).

[Sta23] T. Staruch, *Multi-agent systems for production planning*, Dspace CTU (2023).

[Stu14] Peter Sturm, *Pinhole camera model*, pp. 610–613, Springer US, Boston, MA, 2014.

[Sze11] Richard Szeliski, *Computer vision*, Springer London, 2011.

[TAK21] Fuyuki Tokuda, Shogo Arai, and Kazuhiro Kosuge, *Convolutional neural network-based visual servoing for eye-to-hand manipulator*, IEEE Access **9** (2021), 91820–91835.

[TH18] Tsing Iuan James Tsay and Pei Jiun Hung, *Behavioristic image-based pose control of mobile manipulators using an uncalibrated eye-in-hand vision system*, Artificial Life and Robotics **23** (2018), no. 1, 94–102 (English).

[TL89] R.Y. Tsai and R.K. Lenz, *A new technique for fully autonomous and efficient 3d robotics hand/eye calibration*, IEEE Transactions on Robotics and Automation **5** (1989), no. 3, 345–358.

[VIC22] CTU VIC, *Mediatéka, Čvut v praze, výpočetní a informační centrum*, 9 2022.

[Vor22] S. Voronov, *Device for hand guiding of an industrial robot*, Dspace CTU (2022).

[VRDJ95] Guido Van Rossum and Fred L Drake Jr, *Python reference manual*, Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[WHS23] Xiao Wang, Jiasheng Huang, and Hanwen Song, *Robot-world and hand–eye calibration based on quaternion: A new method and an extension of classic methods, with their comparisons*, Mechanism and Machine Theory **179** (2023), 105127.

[WVN+19a] Bernhard Wally, Jirí Vyskočil, Petr Novák, Christian Huemer, Radek Šindelar, Petr Kadera, Alexandra Mazak, and Manuel Wimmer, *Flexible production systems: Automated generation of operations plans based on isa-95 and pddl*, IEEE Robotics and Automation Letters **4** (2019), no. 4, 4062–4069.

[WVN+19b] Bernhard Wally, Jiří Vyskočil, Petr Novák, Christian Huemer, Radek Šindelář, Petr Kadera, Alexandra Mazak, and Manuel Wimmer, *Production planning with iec 62264 and pddl*, 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), vol. 1, 2019, pp. 492–499.

[WVN+21] Bernhard Wally, Jiři Vyskočil, Petr Novák, Christian Huemer, Radek Šindelář, Petr Kadera, Alexandra Mazak-Huemer, and Manuel Wimmer, *Leveraging iterative plan refinement for reactive smart manufacturing systems*, IEEE Transactions on Automation Science and Engineering **18** (2021), no. 1, 230–243.