BACHELOR DEGREE PROJECT

UNIVERSITY
OF SKÖVDE

VISION-DRIVEN ASSEMBLY ROBOT

Bachelor Degree Project in Production/Mechanical Engineering G2E, 30 credits
Spring term 2022

Antonio Pozo León
Alexander Von Knoop Ruíz

Supervisor/Handledare: Bernard Schmidt
Examiner/Examinator: Magnus holm

Abstract

This project focuses on the integration of Computer Vision (CV) and robotics to automate object assembly tasks using a collaborative robot. By combining both disciplines, the project aims to improve productivity and safety in multiple environments by enabling safe and efficient assembly operations. To this end, an RGBD camera will be used as a vision system to capture both color and depth data from the environment. A further aspect to highlight is the importance of education and training for operators to enhance accessibility and familiarity with manipulator robots beyond the robotics industry and for the creation of adapted paths without knowledge of conventional robot path programming. The objectives cover conducting research, developing simulation environments, implementing algorithms, and their application in both real and virtual robots. This project is intended to enhance automation and efficiency while promoting the advantages of computer vision and robotics in industrial applications.

Keywords: Computer Vision; Collaborative robot; RGBD Camera; Assembly Process

Certification

This thesis has been submitted by Antonio Pozo León and Alexander Von Knoop Ruiz to the University of Skövde as a requirement for the degree of Bachelor of Science in Production/Mechanical Engineering.
The undersigned certifies that all the material in this thesis that is not my own has been properly acknowledged using accepted referencing practices and, further, that the thesis includes no material for which I have previously received academic credit.

Antonio Pozo León *Alexander Von Knoop Ruiz*

Skövde 2023-06-02
School of Engineering Science /Institutionen för Ingenjörsvetenskap

Acknowledgments

We would like to extend our heartfelt gratitude to the University of Skövde for providing us with the opportunity to undertake this project and for imparting valuable knowledge throughout our journey. We are also deeply appreciative of our examiner and supervisor, whose guidance and support were instrumental in shaping the direction and success of our work. Their insights and feedback enriched our understanding and propelled us forward.

We extend our heartfelt gratitude to our families for enabling us to pursue our studies abroad, particularly in the incredible experience that is Erasmus. Their support and belief in our aspirations have been pivotal in making this journey possible. Without their encouragement and understanding, navigating the challenges and embracing the opportunities that come with studying in a new environment would not have been as achievable. We truly value the role our families have played in shaping this enriching chapter of our academic and personal growth.

Last but not least, we want to extend our gratitude to all the friends we've connected with during this experience. Your companionship, shared moments, and encouragement have truly enriched our journey. From exploring new places together to navigating the challenges of studying abroad, your presence has made a significant impact. These friendships have become an integral and cherished part of our Erasmus adventure, reminding us that the connections we've formed are one of the most valuable takeaways from this remarkable experience.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

3D	Three Dimensions	Space with width, height, and depth
ABB	Swedish-Swiss multinational corporation ABB	International Engineering company leader in technology, electrification and for social and industrial purposes
APIs	Application Programming Interface	Application Programming Interface
AR	Augmented Reality	Overlays digital information onto the real world, enhancing perception and interaction
ASSAR	Industrial Innovation Arena	A meeting place where education, innovation, and research bring new solutions for the industry
C#	C Sharp	Object-oriented programming language
Cobots	Collaborative Robots	Robots designed to work safely alongside humans, equipped with features that allow them to detect when a human is nearby and adjust their behavior accordingly
CV	Computer Vision	Field of artificial intelligence focused on enabling machines to interpret and understand visual information
DLT	Direct linear Transform	Algorithm for determining the pinhole camera parameters by extracting correspondences between 2D image points and 3D world points
DSRM	Design Science Research Methodology	Problem-solving paradigm used as a guideline for designing an innovative artifact

GoFa	GoFa CRB 15000 Robot	ABB robot model
IDE	Integrated Development Environment	software application that helps programmers develop software code efficiently
IR	Infra-Red	Electromagnetic spectrum with wavelengths longer than visible light
IRC5	IRC5 robot controller	Multi-robot controller from ABB enabled for the control of the end-effector, the additional axes, and peripheral equipment with PC tool support that optimizes the robot performance for short cycle times and precise movements
PCD	Pointcloud	Data file that collects 3D information of a certain object or environment
PCL	Point Cloud Library	open-source library project for 2D/3D image and point cloud processing
RGB	Red, Green, and Blue	Color model in which the red, green, and blue primary colors of light are added together in various ways to reproduce a broad array of tones
RGBD	RGB & Depth Camera	Camera that provides both depth (D) and color (RGB) data as the output in real time.
ROS	Robot Operating System	Set of software libraries and tools that help you build robot applications.
SDK	Software Development Kit	A set of tools that provides a developer with the ability to build a custom app which can be added on, or connected to another program

TCP	Tool Center Point	Concept used in robotics that defines the position of a robot's end-effector in relation to its tool
TCP/IP	Transmission Control Protocol/Internet Protocol	Set of communication protocols used for connecting devices and networks to each other over the Internet
ToF	Time-of-flight	Technology used in various applications to measure the distance between an object and a sensor

1. Introduction

In this chapter, the project presents various industry-related disciplines and the challenges they are intended to address. Once the problem has been clarified, the aim and specific objectives to be achieved are set out, while certain limitations are established and a vision of the sustainability perspective of the project is shown. Finally, an overview that clarifies to the reader which chapters will be found next and their content is included.

1.1. Background

Nowadays, robots can be defined as programmable machines capable of performing automatically complex tasks typically conducted by humans (Ben-Ari et al., 2018). The concept of a robot dates to ancient times, with myths and legends featuring mechanical beings with human-like qualities (Asada & Slotine, 1991). However, the first modern robot was built in 1954 by George Devol, and it was designed for industrial use (Malone, 2011). Since then, robots have become increasingly sophisticated and are used in a wide range of industries, from manufacturing and logistics to healthcare and entertainment.

Computer Vision (CV), on the other hand, is a multidisciplinary field that combines information extraction techniques, pattern recognition, and increasing use of artificial intelligence to understand information from visual data such as digital images. The goal of computer vision is to enable a computer through algorithms to interpret and perceive visual information in the way a human would (Wiley & Lucas, 2018). Computer vision applications cover a wide range of possibilities, from automating self-driving cars and assembly lines to medical imaging and Augmented Reality (AR).

Regarding the field of robotics, just as communication between a robot and external sensors equipment allows for greater automation and flexibility in its operations, implementing a computer vision system offers the robot a more advanced understanding of its environment. Concerning industrial environments, robots are often used to perform repetitive tasks that require precision and speed. However, if the working conditions may vary, robots need to be able to perceive and interpret their environment. Therefore, by incorporating computer vision technologies into robots, they can perform tasks with more adaptability thanks to a better understanding of their environment. Some applications resulting from this approach could be found in manufacturing plants, where robots can use computer vision to identify and sort products on a production line, or to inspect products for defects. Other scenario could be at warehouses, where its combination can provide robots visual information about its location and the objects around it to navigate, avoid obstacles, or to locate and retrieve items from storage shelves. In addition, robots can use computer vision in healthcare settings to assist with surgeries or to help patients with mobility issues (Shahria et al., 2022).

Real-world examples of this intersection between robots and computer vision in industrial environments can be seen in companies like Boston Dynamics, which has developed robots that use computer vision to navigate complex environments and perform tasks like opening doors and climbing stairs (Guizzo, 2019). From the European Union, projects such as SYMBIO-TIC have successfully integrated robots with machine vision by focusing on symbiotic human-robot collaboration in manufacturing processes (Pérez et al., 2016). After considering this information, the integration of computer vision and robotics has opened new possibilities for automation, safety, and efficiency in industrial environments. As technology continues to evolve, one can expect to see even more advanced robots that can perform increasingly complex tasks with greater precision and autonomy.

1.2. Problem Statement

The increasing need for robots in specialized assembly stations requires a larger number of stations with a consequent preparation of the station and programming of the pick and place routes to be performed during assembly. This type of preparation means that, in the event of a change in the task, a large part of the program has to be updated. It should be noted that those workers in charge of this updating task require previous training for it. This lack of flexibility in industrial environments demonstrates its need for prior specialization and training to ensure productivity and worker safety.

It also highlights the need to educate and train operators and increase outreach efforts to make robot manipulators more accessible and familiar to people outside the robotics sector, as it requires a high level of technical expertise to program the creation and updating of routes. This could involve developing more user-friendly interfaces, providing training and educational resources to help operators learn how to use these machines effectively, as well as promoting the benefits of robot manipulators to potential customers and stakeholders. In turn, the work that such stations are often used for may be in environments that are hazardous to humans, so accessing the station to update its movements may be a complex task or may require stopping the production line.

To contribute to the development of adaptive assembly stations, the integration of vision systems in robotic stations is shown to be a powerful discipline through which to give the robot controller the ability to automatically generate code for the robot to perform a task based on information captured from the environment, thus extending the sensorisation of the manipulator robot.

1.3. Aim and Objectives

The aim of the project is to develop a system that uses computer vision to enable a robot to interact with a human in both real and virtual environments for assembly processes. These could be achieved by the following objectives divided in two groups: research and implementation of the project.

In the research part could be found:

- Conduct a literature review and research current developments in the field to gather relevant information about software and related work.
- Acquire competence in various programming languages and environments for the effective development of the required software for a successful application.

Then, in the project implementation is required:

- Create a simulation environment using virtual reality technology to represent the workspace and objects.
- Develop a control system that allows the robot to accurately respond to the movements of the objects with computer vision.
- Develop an algorithm for recognizing and identifying selected objects in the workspace.
- Implement the developed algorithms and control systems on a real robot, if resources permit, and test them in the physical workspace.

1.4. Delimitations

One of the challenges faced in this project is the difficulty in accurately recognizing objects in the workspace. This can lead to errors in the robot's movements. To mitigate this challenge, a solution has been selected: the use of a predefined library of pieces. By utilizing a library, time is saved, and the risk of failure is minimized. Additionally, colored pieces are employed for easier segmentation, aiding in object recognition.

Determining the optimal camera configuration is crucial for effective integration of the vision system. The project considered two options: using two or more cameras or employing a single fixed or movable camera. The use of two cameras would be a more suitable implementation of the remote teleoperation application when the station is in a dangerous space for the operator. However, the use of multiple cameras was rejected due to the considerable data processing requirements. After careful consideration, it was decided to attach the camera to the robot. This configuration facilitates algorithms for object recognition and allows for a similar analysis from both the operator's and robot's perspectives, enhancing the system's overall efficiency. In relation to camera configuration, working with an adaptive implementation by analysing both workspaces is better than using predefined part positions (this situation would not require a camera at the robot station). Starting from random part positions allows for better adaptability to cope with possible placement errors in the predefined position or during the assembly process.

The choice of equipment and setup is vital for the successful implementation of the project. The resources provided by the university include an Azure Kinect DK, ABB GoFa CRB 15000 robot, Schunk Co-act EGP-C 40 gripper, and the facilities from Industrial Innovation Arena (ASSAR). Although an alternative robot option was available, it was not chosen due to its size

and other factors such as safety concerns. Consequently, the ABB GoFa CRB 15000 robot was selected for its flexibility, safety features, and ease of programming. Due to Schunk Co-act EGP-C 40 gripper the size of pieces used must be adapted to its range (12-24 mm). Furthermore, the available space for the setup limited the robot's movements, ensuring a correct movement of the robot during the pick and place. The parts to be handled by both the robot and the operator will consist of small prismatic 3D-printed pieces. Each piece will have a distinctive color and different geometries to show the adaptability of the project when processing and handling different parts at the same time.

In the project, certain programs used for robot programming, such as RobotStudio for ABB, require licenses that are not freely available. However, the project team has secured student licenses, which grant access to the necessary software for the successful implementation of the project. These licenses ensure that the team can utilize RobotStudio effectively, allowing for robot programming and simulation. In addition to RobotStudio, other programs such as Visual Studio are employed in the project. Visual Studio is a license-free development environment that provides a range of tools and features for software development and application creation.

1.5. Sustainability

Over the years, the concept of sustainability has continued to evolve and gain importance in various fields, including business, economics, politics, and social sciences. Today, sustainability is considered a critical element in achieving long-term success and prosperity for individuals, communities, and the planet as a whole.

In the early days, sustainability was mainly focused on environmental concerns, such as protecting natural resources and ecosystems from degradation and pollution (Agency, 2019). However, as awareness grew, the concept expanded to include social and economic aspects, such as reducing poverty and inequality, promoting human rights, and creating jobs and economic growth. Nowadays, sustainability is a multifaceted and interdisciplinary concept that encompasses many dimensions, including environmental sustainability, economic sustainability, and social sustainability (see Figure 1.1). It aims to balance the needs of the present generation with the needs of future generations, recognizing that resources are limited and must be used in a responsible and efficient manner (UNDP, 2015).

Figure 1.1: Sustainability Venn diagram adapted from Maricopa (2021)

Moreover, sustainability is not only a matter of ethics or moral responsibility but also a practical necessity; businesses, governments, and individuals must adapt to a changing world and find ways to live and operate sustainably to ensure a secure future for themselves and future generations. Thus, introducing this perspective within industry itself represents a holistic and integrated approach to development that seeks to balance economic growth, social progress and environmental protection from the manufacturing of goods and the design of production lines. It has become an increasingly important consideration in various fields, including robotics. Among the various fields where their consideration has increased considerably more are robotics and process automation. Some aspects related to this project in terms of sustainability can be considered.

In a social aspect, a sustainability view can improve the safety and well-being of workers by taking over tedious, repetitive, or dangerous tasks, such as assembling objects in hazardous environments. This can lead to improved job satisfaction, reduced stress, and injuries, and increased free time for other activities. Additionally, robotics technology can contribute to the development of new skills and job opportunities in related fields, promoting social mobility and diversity. In connection with the environment, the proposed assembly station can help to reduce waste, emissions, and resource consumption by optimizing manufacturing processes and reducing errors and defects. For example, by automating assembly tasks, robots can reduce the amount of scrap materials generated and minimize the energy required for production.

Additionally, the proposal can enhance economic sustainability by increasing productivity, reducing costs, and enabling new business models adapted to flexible stations for adaptative production lines. By automating repetitive or low-skill tasks, robots can free up human labour for more valuable activities, such as research, design, or customer service. Additionally, robotics technology can enable new forms of production, such as customized or on-demand manufacturing, which can increase customer satisfaction and revenue. While the proposed

project offers several benefits, there are also potential drawbacks and challenges to consider. The upfront costs of implementation may strain organizations with limited resources, potentially affecting their ability to invest in the required technology. Furthermore, the technical complexities involved in developing accurate computer vision systems and integrating them into the assembly process can add to the overall cost and implementation challenges. Adapting to varied assembly environments poses additional complexities, while the maintenance requirements and potential job displacement impact.

In summary, the integration of a sustainability perspective into the development of a robot guided with computer vision, can lead to a more efficient, safe, and responsible approach to technological innovation. Otherwise, it's essential to evaluate and address drawbacks to ensure a successful implementation of the project minimizing any potential negative impacts. All these concepts are going to be extended to the project in Chapter 7.3: Sustainable Development.

1.6. Overview

The project has opened with an **Introduction** throughout Chapter 1 that sets the stage for the research and emphasizes the significance of sustainability. Subsequently, the **Theoretical Framework** (Chapter 2) explores various aspects such as computer vision, depth cameras, software development tools, RobotStudio, and industrial robots. The upcoming **Literature Review** (Chapter 3) offers a comprehensive analysis of vision-assisted assembly, human-robot interaction, camera calibration, and object recognition techniques. Thereafter, the **Methodology** followed (Chapter 4) consists of a design science research approach to provide guidance to bring robustness to the resulting project. All of these foundational chapters lay the groundwork for developing a feasible **Implementation** detailed in Chapter 5. The subsequent **Analysis and Results** (Chapter 6) shed light on camera calibration, TCP and work object definition, and execution examination. The project's significance in the content and development is further debated in **Discussions** (Chapter 7), emphasizing the integration of sustainable practices and their impact. Finally, the **Conclusions** (Chapter 8) encapsulates the key findings, while the **Future Work** (Chapter 9) provides avenues for future improvement and exploration. In the Appendix A can also be found a Time Plan carried out for this thesis.

2. Theoretical Frame of Reference

This chapter will present, in the first instance, a project design overview so the reader may begin with an initial idea of the project and know the components and methods necessary for its implementation. Subsequently, the theoretical concepts necessary to show the appropriate tools for the purpose of the thesis will be presented.

2.1. Design Overview

Figure 2.1 presented offers a comprehensive overview of the key topics that will be addressed throughout the project. It outlines the integration of computer vision techniques, specifically the utilization of an RGB & Depth Camera (RGBD) camera, to capture and process images for object modelling and manipulation from both user and robot. The industrial robot, in this case, the GoFa CRB 15000 Robot (GoFa) robot produced by Swedish-Swiss multinational corporation ABB (ABB), will play a crucial role in carrying out the assembly tasks based on the information provided by the computer vision system.

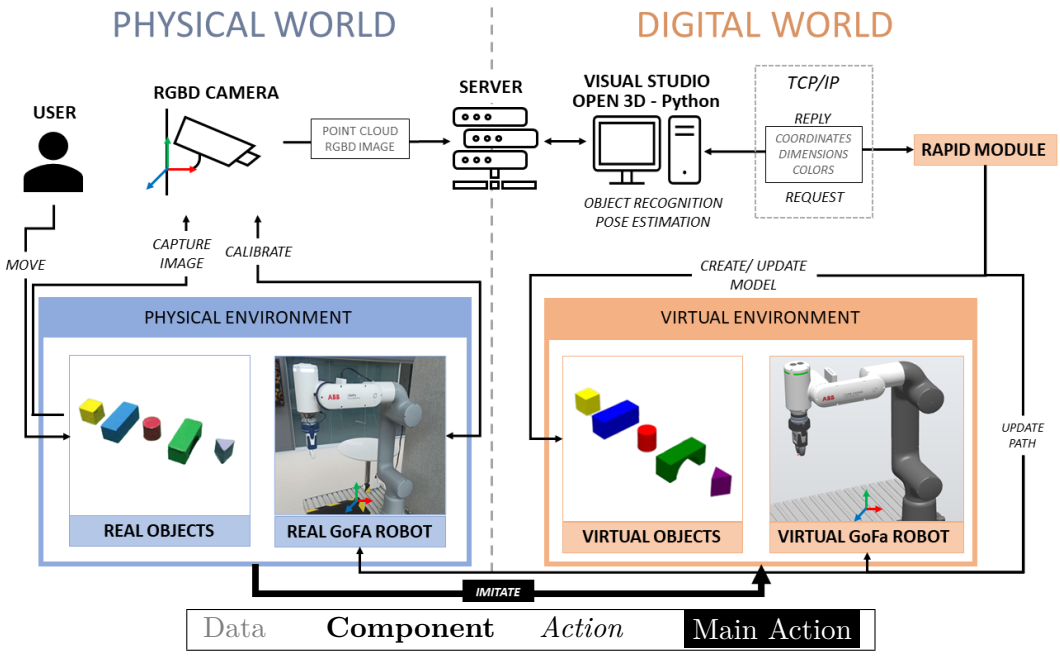


Figure 2.1: Project Design Overview

The project also highlights the importance of development software, including Visual Studio environment, for programming and software development in Python and C Sharp (C#). Additionally, the use of OpenCV and Open3D libraries for image processing, along with software for camera connection and communication, will contribute to the system's overall functionality. The creation of virtual environments, digital models, and paths, as well as socket communication, further demonstrate the comprehensive nature of the project's scope. By considering these key aspects, the project is intended to leverage computer vision, development software, and industrial robots to achieve efficient and secure automation of assembly tasks.

2.2. Computer Vision

Computer vision, as defined in Background, is the field of study that develops the ability of computers to simulate through algorithms how a human can interpret and extract valuable information from images from visual sensors. It draws on a range of disciplines, including computer science, engineering, mathematics, and psychology, to develop algorithms and technologies that can analyze and make sense of images and video (Wiley & Lucas, 2018).

2.2.1. Industrial depth cameras

Depth cameras are a type of camera that captures not only color information but also the distance from the camera. This allows for the creation of Three Dimensions (3D) point clouds, which can be used for a variety of applications in computer vision, such as object detection, tracking, and recognition. Depth cameras work by emitting light, usually infrared, and then measuring the time it takes for the light to bounce back to the camera after hitting an object. By using this time-of-flight measurement, the camera can calculate the distance of each pixel from the camera (Klette et al., 1998). There are several types of depth cameras available on the market today, each with its own strengths and weaknesses. Some of the most common types include:

- Time-of-flight (ToF) cameras: They emit a pulse of light and then measure the time it takes for the light to return, which is used to calculate the distance to the object. These cameras are typically used for short-range applications, such as facial recognition or gesture recognition (Iddan & Yahav, 2001).

- Structured-light cameras: These cameras project a pattern of light onto the scene and then use stereo cameras to capture the distorted pattern. By analysing the distortion of the pattern, the camera can calculate the depth of each pixel. Structured-light cameras are known for their high resolution and accuracy. These cameras are often used in industrial applications for quality control or inspection.

- Stereo cameras: These cameras use two or more cameras to capture images from slightly different angles, and then use triangulation to calculate the depth of each pixel. Stereo cameras are known for their accuracy and simplicity but can be limited by the baseline distance between the cameras. These cameras are often used in robotics and autonomous vehicles for navigation and obstacle avoidance.

Some key difference between commercial and industrial depth cameras is their durability and reliability due to they are often built to withstand harsh environments. Industrial depth cameras are also typically designed to provide highly accurate depth data with low noise levels, which is important for applications such as 3D scanning and quality control. Commercial depth cameras, on the other hand, may sacrifice some accuracy for the sake of affordability and accessibility. Another difference between commercial and industrial sensors is their level of customization and configurability. Industrial ones are often designed to be configurable and customizable to meet the specific needs of different industrial applications. This includes features such as adjustable exposure times, gain, and image processing algorithms.

In manufacturing, depth cameras can be used for robotic pick-and-place operations, where they help robots identify and locate objects in a cluttered environment. This is made possible by the depth data captured by the camera, which allows the robot to differentiate between objects of different heights and shapes. The information provided can be used to create accurate digital models for design, inspection, and quality control purposes by detecting anomalies in the geometry. In terms of robotic applications, human-robot interaction with Collaborative Robots (Cobots), can also be improved by implementing depth cameras so it enables safe and efficient collaboration by providing real-time information about the position and movement of humans in the robot's environment. This allows the robot to adjust its behavior accordingly to avoid collisions and ensure safe interaction with humans.

2.2.2. RGBD Camera

In terms of the types of depth cameras discussed earlier, RGBD cameras can be considered a subset of structured-light cameras and time-of-flight cameras. Structured-light RGBD cameras project a pattern of light onto the scene and then use color cameras to capture the distorted pattern, while time-of-flight RGBD cameras emit a pulsed light signal and measure the time it takes for the light to bounce back to the camera. Both types of cameras capture both color and depth information in a single device, making them a popular choice for a wide range of applications in computer vision and robotics. By capturing both color and depth information in a single device, RGBD cameras eliminate the need for separate color and depth sensors and the associated calibration issues. This makes them more convenient to use and reduces the overall complexity of the system. In terms of the increasing use of machine learning algorithms for computer vision applications, the RGBD can be used to create powerful implementations by interpreting visual data in real time. This involves training models to recognize specific objects, faces, or patterns in visual data, and then using these models to identify similar objects or patterns in new data.

Overall, for these reasons, the use of an RGBD camera can be a valuable tool for researchers and developers working in the field of computer vision, providing advanced depth sensing and Red, Green, and Blue (RGB) camera capabilities that can be used to develop new and innovative applications. The provided camera, the Azure Kinect DK camera (see Figure 2.2), has this series of characteristics of the RGBD cameras plus other aspects that will allow correct adaptability to the implementation.

Figure 2.2: Kinect Azure Camera DK (Brad Stephenson, 2019)

Here are some of the key characteristics of the Azure Kinect DK that are relevant to our project:

- Depth sensing: The Azure Kinect DK camera can capture high-resolution depth information with a range of up to 10 meters, using a time-of-flight sensor that emits and measures infrared light.
- RGB camera: In addition to depth sensing, the camera also includes a 12-megapixel RGB camera that captures color images at 30 frames per second.
- SDK and APIs: The Azure Kinect DK comes with a Software Development Kit (SDK) and a set of Application Programming Interface (APIs) that allow developers to create custom applications and integrate the camera with other software platforms.
- Open source: Microsoft has released many of the software components of the Azure Kinect DK as open source, allowing developers to modify and adapt them for their own use.
- Compact and portable: The Azure Kinect DK is a small, portable device that can be easily mounted on a tripod or attached to a computer for use in a variety of settings.

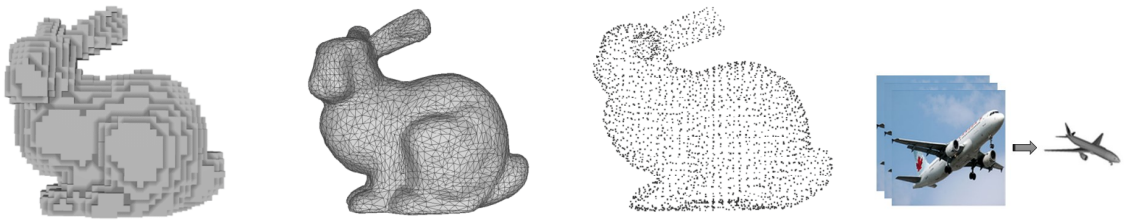
In general, its technical aspects, together with its portability to adapt its installation on the workstation and its SDK to facilitate the development of the application, make the Azure Kinect DK camera a versatile and powerful tool for capturing and processing visual and spatial information (see more technical specifications in C.2). Its capabilities demonstrate the potential to be applicable in fields such as computer vision, robotics, and augmented reality, among others (Microsoft, 2020).

2.2.3. Objects Modeling

One of the important tasks in computer vision is object modeling, which involves representing objects in a digital form that can be processed and analysed by a computer.

There are several methods used in computer vision to model objects, including:

- Geometric/Volumetric modeling: (Lowe, 2004) proposed geometric modeling using geometric shapes such as polygons, curves, and surfaces to represent objects in digital form. Geometric models can be used to represent objects in 2D or 3D space.
- Feature-based modeling: This involves using specific features of an object, such as edges or corners, to create a model of the object (Lowe, 2004). Feature-based models can be used to identify objects even when their orientation or position changes.
- Point cloud modeling: Rusu & Cousins (2011) proposed point cloud modeling, which involves representing objects as a collection of 3D points in space. Point cloud models are often used in applications such as 3D scanning and mapping.
- Deep learning-based modeling: This involves training deep neural networks to recognize and classify objects in digital images and videos (LeCun et al., 2015). Deep learning models can be trained to recognize objects in a wide variety of contexts and can be highly accurate.



(a) Geometric **(b)** Feature-based **(c)** Point cloud **(d)** Deep learning-based ad. from Fu et al. (2021)

Figure 2.3: Objects Modeling types (Gao et al., 2022; Fu et al., 2021)[1]

Overall, computer vision offers a wide range of tools and techniques for modeling objects, and the choice of method will depend on the specific application and the characteristics of the objects being modeled. In this project, point cloud modeling may be a suitable approach to model objects for several reasons:

- High accuracy: Point cloud models can be highly accurate (Nguyen & Le, 2013), especially when many points are used to represent the object. This makes it a suitable approach for applications where accuracy is critical, such as 3D scanning or mapping.
- Flexibility: Point cloud models can be generated from a variety of data sources, including lidar, photogrammetry, and structured light scanning. This flexibility makes it a suitable approach for projects where multiple data sources are used.

- Point cloud processing tools: There are several open-source and commercial point cloud processing tools available, such as the Point Cloud Library (PCL) and CloudCompare (Rusu & Cousins, 2011). These tools provide a wide range of functions for processing, filtering, and analysing point cloud data.

- Object recognition: Nguyen & Le (2013) proposed that point cloud models can be used to recognize and classify objects, even when their orientation or position changes. This makes it a suitable approach for applications such as robotic vision, where objects may be encountered in a wide range of contexts.

The selection of an object modeling approach relies on the project's particular requirements and limitations. Consequently, when aiming to create flexible models suitable for both the virtual environment and manipulation by the user and robot, a straightforward modeling technique is preferred. On the other hand, point cloud modeling proves to be suitable for tasks such as object recognition and direct data communication from the camera.

2.3. Development Software

2.3.1. Visual Studio

Software development has required tools for writing and compiling code in different languages as well as execution and debugging for evaluation of results, which is why integrated development environments or IDEs are more fundamental tools for software designers as they increase the productivity of developers by combining all these capabilities in a single environment designed for ease of use, among the different IDEs available for different languages, Visual Studio is a powerful and versatile option.

Visual Studio is a feature-rich Integrated Development Environment (IDE) used for building software applications. It supports multiple programming languages and frameworks such as C++, C#, .NET, and Python. The IDE includes productivity tools such as IntelliSense, debugging tools, and testing tools that help developers write high-quality, reliable, and optimized code. Visual Studio also provides a range of collaboration tools and supports cloud development through Azure. Overall, Visual Studio is a versatile and powerful IDE that supports the entire software development lifecycle and is widely used by developers for building complex software applications (Code, V. S., 2019).

2.3.2. Open3D

The open-source library OpenCV aims to provide an easy-to-use computer vision infrastructure that offers many implementable functions in different domains such as object inspection in the industry, user interfaces, camera configuration, detection in medical images, and automation and sensitization of robots. However, the processing of three-dimensional data captured in a physical environment by sensors such as LiDAR or cameras that allow 3D reconstruction and modeling, are represented in the form of point clouds, meshes, and other representations. The software required for basic processing differs significantly from the above-mentioned 2D image processing, although some functions also work for pre-processing

3D images. It is also considered a rather laborious task compared to other types of data. Open3D, created to address this need, is an open-source library that enables the rapid development of software that works with 3D data. These are the main functionalities offered by the Open 3D library (Zhou et al., 2018):

- Geometry: Data structures and basic processing algorithms
- Camera: Camera model and camera trajectory
- Odometry: Tracking and alignment of RGBD images
- Registration: Global and local registration
- Integration: Volumetric integration I/O Reading and writing 3D data files
- Visualization: A customizable GUI for rendering 3D data with OpenGL
- Utility: Helper functions such as console output, file system, and Eigen wrappers
- Python: Open3D Python binding and tutorials

As the backend is implemented in C++11, it has facilitated its compatibility with other programming languages such as Python or C#. In addition, being open source, it allows not only greater flexibility and configurability to the user but also the contributions to the library by the community are very present (Zhou et al., 2018).

2.3.3. Camera Connection

As mentioned above, the Azure Kinect DK comes with a SDK with which to develop applications that integrate the camera with other software platforms. For this, Microsoft provides a series of steps to achieve a simple connexion, with which to activate, transmit and capture images in real-time from the Kinect through Visual Studio code. The process, roughly speaking, is as follows (Cedmonds, 2022):

1. Install the corresponding NuGet package (for C#) or library (for Python) in Visual Studio, so you can access the functions related to them by using the corresponding header.
2. Find the devices that are connected to your computer and get the name of the desired camera from the list (you are also able to connect with several devices).
3. Open the device and read its data. This will give you access to all the information about the state of the camera, so as the data from each of its sensors. Therefore, now it´s possible to show its content as an output or save it by making a capture and placing it in a selected folder.

2.4. Digital Models

The Digital Model is described as the digital construction linked to a physical system (see Figure 2.4), in this case, a robot cell, on which its design has been based, which has collected all the necessary information, to mirror as closely as possible what can be observed in the real world in terms of dimensions, positions, forces, and kinematics. This model lacks automatic communication or synchronisation with the real environment, so a change in the state of each environment will not be reflected in the other one unless these modifications are incorporated manually. Due to this situation, its use is mainly focused on the simulation of processes that do not require a simultaneous process with the real robot, either because it is not available or because tasks are being carried out that may compromise its physical integrity (Kritzinger et al., 2018).

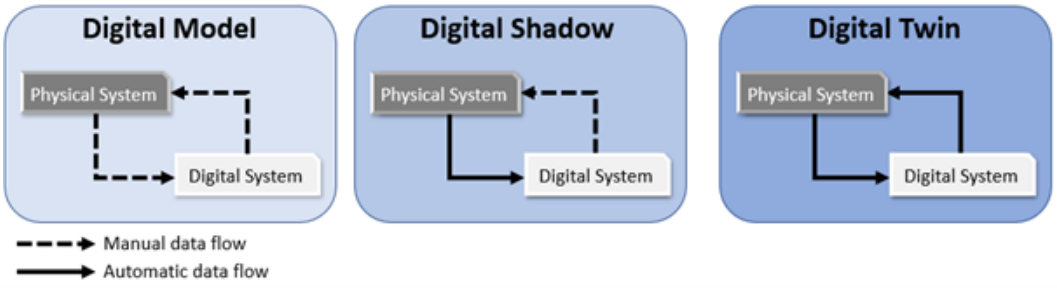


Figure 2.4: Data flow in Digital Model, Digital Shadow, and Digital Twin, adapted from Nikula et al. (2020) [2]

By incorporating a more complex level of integration to this model, an automated one-way dataflow can be implemented whereby any change in the real environment will cause the same behavior in the simulated model, known as Digital Shadow (Kritzinger et al., 2018). Finally, the complete integration between both systems, known as a Digital Twin, is obtained through the implementation of a two-way automated dataflow where the behavior of one of the two environments is reflected in the other, thus achieving a perfect synchronisation (Kritzinger et al., 2018).

2.5. RobotStudio

RobotStudio is a program designed by ABB for robot programming, modeling, and simulation of robot cells, allowing the user to program the real robot executable code without the need to have the robot or set of robots and their respective workspaces physically present (Robotics, 2007b). Among the different functionalities of RobotStudio, those applicable to the design of the implementation will be described, such as the creation of routes, the modeling of parts, communication with external programs, the use of the controller in the virtual environment and its own programming language

2.5.1. Virtual Environment

The programming of robots through the RobotStudio environment, known as offline programming, allows tests to be carried out in a controlled manner in a virtual environment before transferring it to the real robot for it to execute. For this purpose, a virtual controller simulating the IRC5 robot controller (IRC5) (Robotics, 2007b). RobotStudio can also be used with real IRC5 controllers for the execution of the program on the real robot, known as online programming, which is usually done through the Flex pendant of the controller itself (Räsänen, 2020). In this way, it can be executed both on the real robot and on the simulator either Physical-to-cyber synchronization (Digital Shadow) or Cyber-to-physical synchronization (Digital Twin).

This environment (see Figure 2.5) now has the possibility to select from a wide variety of ABB robots, controllers, and add-ins to provide additional functionality such as the creation, import or export of CAD models, simulation of real equipment such as sensors and it can feature a virtual reality mode.

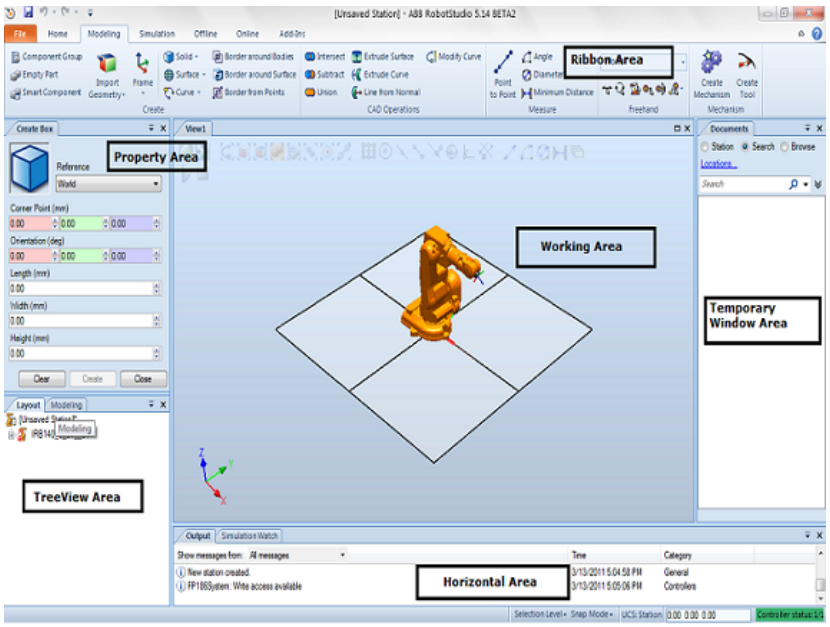


Figure 2.5: RobotStudio User Interface ABB Developer Center (n.d.)

2.5.2. RobotWare and RAPID

RobotWare is ABB's controller software for industrial robots. It brings flexibility and reliability combined with built-in process control functionality and broad communication capability (ABB, 2004) as well as accurate performance. In terms of speed, path tracking, obstacle avoidance, and synchronization with external devices such as conveyors and cameras, it makes this software a powerful tool for I/O management and thus HMI (human-machine interaction) (ABB, 2014). This effective and configurable software is due to ABB's robot language: RAPID.

2.5.3. Path Creation

When dealing with an offline programming task, RobotStudio has a lot of functionalities such as creating robot targets conveniently through its interface, configuring and adjusting routes, positioning objects, and setting targets on them, among others (Holubek et al., 2014). Its purpose is to facilitate the programmer in the task of creating routes, especially when there is the added complexity of working or interacting with workpieces.

When positioning the object in the virtual environment as it would be in the real workstation, the commands for "set position" are available. It also has several functionalities that help to adapt to the shapes of the surface of the workpiece, so when creating an individual target, the path could be generated following a "linear" but also "curve" movement (Holubek et al., 2014). A very important issue is to consider what position of the tool comes to the workpiece, a fundamental aspect when determining each target. This is why there are functions to perform offsets, with which to include an additional value to an existing target in position and/or orientation of the end effector to increase the precision in this target without modifying its values during the offline programming stage, to include a series of positions dependent on this target or even the possibility of modifying the position dynamically.

Additionally, once the route has been configured, to verify that the expected motion is going to be carried out, there are other functionalities besides executing the code, which visually shows the movement that would be carried out in one of the sections of the route, as well as its feasibility or reachability. Finally, for program execution, the IRC5 controller offers Advanced Robot Motion Functionality for reducing path deviation and ensuring the robot's motion control on one or several robots, either following a coordinated motion or independently of each other (ABB, 2014).

2.5.4. Socket Communication

Transmission Control Protocol/Internet Protocol (TCP/IP) (see Figure 2.6) is a set of communication protocols used for connecting devices and networks to each other over the Internet. It consists mainly of the division of data into packets via TCP, the sending and routing to the local or global network via IP and finally, TCP takes over again in the reassembly of data at the destination device (Blank, 2006). Due to its high scalability and efficiency, it is the protocol on which communications over the Internet are based, which has led it to be compatible with all computers and a large majority of devices (Blank, 2006).

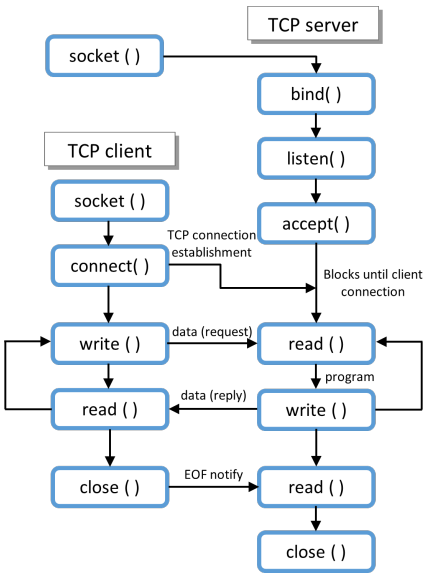


Figure 2.6: Socket programming workflow, adapted from Bohuslava et al. (2017)

To establish this communication, RobotStudio has a set of functions that support socket programming. In turn, the support of threads in C# will allow the existence of parallel loops (Zahavi et al., 2018) with which to analyse the information from the Camera and the corresponding processing to obtain the necessary parameters for robot Studio, such as the distances or coordinates of the object concerning the Camera or its dimensions, while the other loop continues listening to the possible Report Studio requests to reply as quickly as these parameters have been calculated.

With this programming method, a client-server communication between two programs that are running simultaneously takes place, where the Server creates a TCP/IP socket (node) to which it indicates the IP address and the port number through which the communications will be carried out while waiting for the client's connection. Once the connection is established, both programs will be able to request and exchange data in two-way communication (Telang et al., 2018). The following figure shows the scheme the program will follow to establish communications for data exchange.

2.6. Industrial Robots

Industrial robots are programmable machines designed to perform tasks automatically, with high accuracy and repeatability, within a manufacturing environment. These robots can be used to perform a wide range of tasks, such as welding, painting, assembly, packaging, and material handling. The use of industrial robots has become increasingly common in manufacturing and industrial settings, where they offer numerous benefits over traditional manufacturing processes.

The need of more complex equipment for manufacturing in hazardous environments for humans and its development is not new. As early as 1949, the first manually remote-operated manipulator was designed by means of master-slave interaction, where the tongs on the slave arm copy the exact movement corresponding to a handle from the master arm grasped by an operator. Despite its simple mechanism, this device can replicate the 3 linear motion and 3 rotational motions, allowing the operator to manipulate dangerous objects behind safety glass (Goertz, 1949). The industrial robot's history dates back to the 1950s when the first industrial robot, the Unimate, was developed by George Devol and Joseph Engelberger. The Unimate was used to automate the handling of hot metal parts in die-casting factories. Since then, industrial robots have undergone significant developments and improvements, both in terms of their hardware and software capabilities (O'Regan & O'Regan, 2013).

Driven by the need for increased efficiency, productivity, and quality in manufacturing environments, they have been offering a range of benefits over traditional manufacturing processes, including improved accuracy, repeatability, and speed. They can also operate continuously, reducing downtime and improving overall productivity. Furthermore, robots can perform dangerous or unpleasant tasks, such as handling hazardous materials, thereby improving worker safety and health (Spong & Mohammadi, 2022; International Federation of Robotics, 2021). Today, industrial robots are used in a wide range of industries, including automotive, electronics, pharmaceuticals, and many others, offering numerous benefits over traditional manufacturing processes. They are also increasingly being used in new industries, such as logistics and e-commerce, where they can be used for tasks such as order picking and material handling. The use of industrial robots is likely to continue to grow in the coming years, as manufacturers seek to gain a competitive edge in an increasingly challenging business environment.

2.6.1. Collaborative Robots

Cobots are a type of robot designed to work alongside humans in a shared workspace. Unlike traditional industrial robots, which are typically fenced off for safety reasons, collaborative robots are equipped with sensors and other safety features that allow them to operate safely near humans.

Cobots are designed to be easily programmed and flexible, so they can be quickly reconfigured for different tasks or moved to different locations on the factory floor. They are also typically smaller and more lightweight than traditional industrial robots, which makes them easier to move and integrate into existing manufacturing processes. Its characteristics are making them increasingly popular in manufacturing and other industries, as they offer a way to increase productivity while also ensuring worker safety. (Lefranc et al., 2022).

There are two main types of collaborative robots:

- Power-and-force-limited: Designed to limit their power and force output to a safe level, so that if they encounter a human, they will not cause injury.
- Safety-rated monitored stop: Designed to monitor their surroundings and stop if they detect a human in their workspace.

The provided Cobots, the GoFa CRB 15000 Robot (GoFa) (see Figure 2.7), is a powerful collaborative robot designed for industrial use that can safely work alongside human workers, making it ideal for manufacturing and other industrial applications. The CRB 15000 has a payload capacity of 5 kg and can reach a TCP speed of 2.2 m/s (ABB, 2021)). The robot is equipped with safety features such as safety-rated monitored stop, speed and separation monitoring, hand guiding, and power and force limiting (see more technical specifications in Appendix C.1). This ensures that the robot can operate safely around human workers without causing any harm. The robot can be programmed using ABB's RobotStudio software, which is easy to use and allows for quick programming and implementation of tasks.

Figure 2.7: GoFa CRB 15000 Robotics (2007a)

The GoFa CRB 15000 has a compact design, which makes it easy to integrate into existing production lines. It is also highly flexible and can be easily reprogrammed to perform different tasks. The robot can handle a wide range of applications such as assembly, material handling, machine tending, and pick and place operations. Overall, the ABB GoFa CRB 15000 is a versatile, safe, and powerful industrial robot designed to work collaboratively with human workers. For these reasons is the manipulator robot chosen in the project.

2.6.2. Robotic gripper

Articulated robots themselves are not designed to interact directly with The External World. This task is relegated to an additional device to the robot located at the end of its last joint, known as the end effector (Dai et al., 2022). Depending on their design, these tools can cover a wide range of specific functions such as polishing, welding, soldering, gripping and precise object placement, tasks which are coordinated with the robot by means of a direct hardware connection and sending signals to trigger the relevant actions. In this way, the articulated robot's movement and path tracing combined with the actions of this tool allow the robot to extend its capabilities to interact with the environment in a precise and adapted way to the task in hand.

One of the most common tools consists of a gripper end effector, which consists of two opposing fingers with a mechanism that moves these parts closer or further apart so that the gripper has two states: open or closed. This design allows the robot to pick up and drop

parts that are in the robot's environment as long as they meet the conditions of size, weight and distance from the robot. The gripper in question is the Co-act EGP-C 40 (see Figure 2.8), designed for small components and with compatibility with the ABB GoFa robots and communication with the controller via I/O signals. Among its specifications it is also worth mentioning that it can support a recommended load of less than 0.7Kg and its opening and closing range is between 12-24 mm, so the parts to be handled must meet these conditions of weight and width (SCHUNK, 2018). More information on the specifications of this gripper can be found in Appendix C.3.

Figure 2.8: CAD Model of Schunk Co-act EGP-C 40

3. Literature Review

In this chapter, a literature review will be carried out with the purpose of going into the existing research related to the project and analysing its relationship and possible application. Among the different existing studies, those dealing with human-robot interaction, 3D object recognition or the implementation and calibration of cameras in industrial environments will be the most relevant to be considered.

3.1. Vision-assisted assembly

Rokhim et al. (2021) presents (see Figure 3.1) an intelligent system that uses computer vision and Open-CV computer program to detect objects and transform picture data into position and orientation data for robot control. The system begins with a camera placed at the end of the robot arm to capture the pick and place work area. Then the images captured by the camera will be analyzed and processed by a PC, using a digital image processing method. The process that takes place on this PC uses Open-CV. Then the coordinate data are sent to the controller of the UR5e robotic arm. PC communicating via Robot Operating System (ROS) using Universal Robots ROS driver. MoveIt package is also used in this project. Polyscope, a Universal Robots software, serves as the interface of the controller that can control and monitor the robotic arm directly.

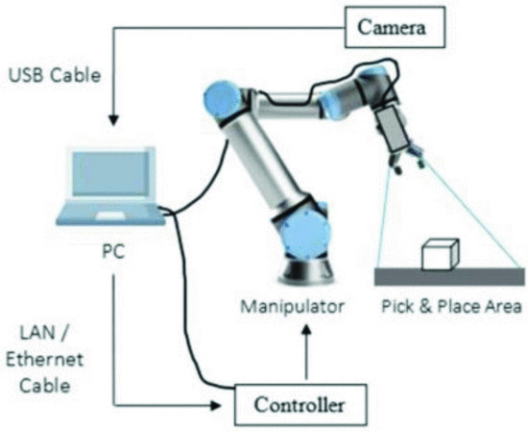


Figure 3.1: Hardware integration design (Rokhim et al., 2021)

Rokhim et al. (2021) also discusses the geometrical transformations and statistical approaches used in this system, including image segmentation, edge detection, and contour detection. The system is designed to detect objects with varying positions and orientations, and the authors provide a detailed explanation of the algorithms used to achieve this. In addition, the motivation of the study discuss the advantages and limitations of existing systems and highlights the need for more intelligent and adaptable systems that can handle varying objects and workpieces.

Moreover, Wang & Mohammed (2012) presents a system for remote assembly using vision assistance and 3D models. The purpose of the study is to demonstrate a system that allows remote users to perform assembly tasks using vision assistance and 3D models. The system aims to provide a more efficient and cost-effective solution to assembly tasks that would otherwise require the presence of skilled labour on-site. By enabling a remote user to assemble objects using a virtual representation of the workspace and objects, the assembly process is assisted by visual feedback from a camera mounted on the robot. This system uses a depth-sensing camera to capture the workspace and objects, which is then used to generate a 3D model of the objects and workspace. The 3D model is used to assist the remote user in selecting the objects to be assembled and generating assembly plans. The system also provides real-time visual feedback to the remote user to assist in the assembly process.

Furthermore, El Makrini et al. (2017) proposes an architecture for collaborative assembly tasks between a human and a robot. The goal of the architecture is to enable safe and efficient collaboration between a human and a robot in assembly tasks. The proposed architecture consists of three main components: a perception system, a task planner, and a controller. The perception system uses a combination of RGB-D cameras and force sensors to detect and track the human and the robot and their respective tasks. The task planner generates plans based on the detected tasks and the capabilities of the human and the robot. The controller executes the plans and ensures safe collaboration by monitoring the forces exerted by the human and the robot.

The proposed architecture is evaluated in a human-robot assembly task scenario, where the robot assists the human in assembling a toy car. The results show that this architecture enables safe and efficient collaboration between the human and the robot and it can adapt to different scenarios and task requirements. In relation with the thesis, the proposed architecture for collaborative assembly tasks, could fit with the goal of demonstrating the use of computer vision in robotics to automate the assembly of objects using a manipulator robot. This architecture can help ensure safe collaboration between the robot and the user, which is important for increasing the accessibility and familiarity of manipulator robots to people outside the robotics industry.

3.2. RGBD Cameras Calibration

Darwish et al. (2019) propose a method for calibrating consumer-grade RGBD sensors for accurate indoor scene reconstruction. The goal of the method is to improve the accuracy of indoor scene reconstruction using RGBD sensors, which are commonly used in robotics and

computer vision applications. The proposed method (see Figure 3.2) involves the application of Zhang's method to obtain the intrinsic parameters of the Camera, i.e. the coefficients that define how the Camera and its lens capture and represent the World and the extrinsic parameters, which define the position and orientation of the Camera with respect to the 3D world.

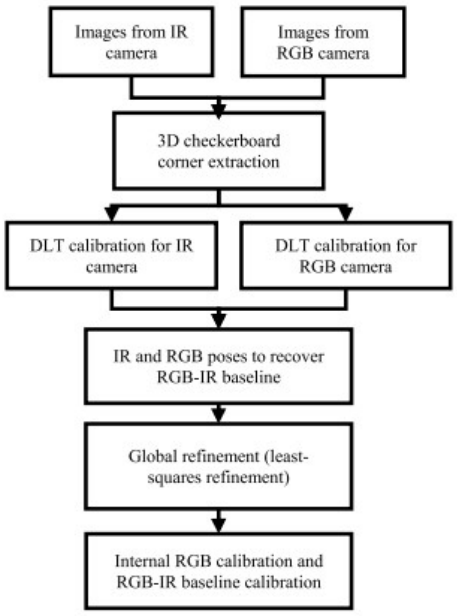


Figure 3.2: Calibration workflow for the RGBD (Darwish et al., 2019)

Zhang's method (Wu et al., 2019) is based on corner detection, for which the camera is used to capture a calibration pattern with known and easily identifiable corners and distances and then evaluate how the parallel and perpendicular lines of the plane in a 3D environment have been distorted when captured in the 2D plane of the photo, i.e. find the Direct linear Transform (DLT) from a 2D picture. The corner coordinates detected in the image and their correspondence with the world coordinates, formulate a set of equations for a least squares regression that relate the camera's intrinsic parameters to the image coordinates. For this process to be effective, different captures of the chessboard will be taken varying its distance from the camera, its position and its orientation. Finally, an iterative refinement is carried out to minimise errors in the parameters obtained by evaluating whether these parameters would give rise to the projections of the original captures.

The improvement of this proposed method lies in the use of the values measured by the Infra-Red (IR) camera, of which its position and orientation with respect to the camera is known, to perform the global refinement since, although the known dimensions of the board squares have allowed estimating their distance, the results show that this new parameter estimation is more accurate and robust, making the method particularly efficient when dealing with challenging lighting conditions and sensor noise.

Figure 3.3: Hand-Eye Calibration with static chessboard (Nari Shin, 2020)

These methods are commonly used in industrial applications that require visual information of the environment or a multi-camera system that needs to know the position of the other cameras with respect to itself. However, when it comes to an application which incorporates an articulated robot, this can provide the process with additional information acquisition as well as a greater degree of automation. Based on the need to determine not only the intrinsic parameters of the Camera but also its location with respect to the surrounding world and the objects that the Camera can observe, the hand-eye calibration (Nari Shin, 2020) allows the acquisition of these values in such a way that they are directly used by the robot itself.

The implementation starts by installing the Camera close to the last joint of the robot, whose coordinates are known at all times as the robot can move to a specific location. Since the coordinates of the last joint with respect to the robot's base are known at all times as the robot can be commanded to move to a specific location, the hand-eye calibration aims to find the relationship between this reference system and that of the Camera. The difference with respect to the previous method is that the chessboard is in a static position, such as the table of the robot's workstation. Meanwhile, the Camera is the one that undergoes the variation of position and orientation with respect to the world since, for each capture, the end effector of the robot is placed in a different pose so that the Camera perceives the chessboard from a different perspective (see Figure 3.3). As the position and orientation of the end effector is undoubtedly known for each capture, the relationship between the coordinates of the corners detected and the position of the corners in the 3D world can be related with greater precision. Consequently, at each next capture, the position of each object in the 2D image plane is able to provide, by direct transformation, the coordinates of the objects with respect to the robot base reference system, very useful information for determining the desired pose of the end effector.

Both proposals will be relevant to the thesis as they propose a viable method to calibrate the RGBD sensors, which may be needed for the vision system of the project. The first proposed method can help improve the accuracy of object detection and recognition, which is crucial for generating trajectories for the robot to perform assembly tasks. Accurate calibration can also help generate a virtual representation of the workspace and objects, which can enhance the interactive building experience for ASSAR visitors. As for the second approach, while it offers the additional advantage of establishing a direct relationship between the posi-

tions of the Camera and the robot, its application may require more advanced programming and communication. However, the installation of the Camera next to the end effector may bring other benefits in capturing the working environment and in the use of an alternative calibration system.

3.3. RGBD data processing for scene recognition

He et al. (2017) have developed a pipeline capable of identifying in an RGBD image of a scene the multiple objects in the scene and their orientation. The motivation lies in the fact that local image descriptors such as the widely used SIFT, which are found in traditional recognition pipelines, have a low success rate when working with texture-less objects such as polished metal pieces or glass, which is a big gap in the applications of computer vision in manufacturing and assembly processes in industry. Therefore, it is proposed to improve the pipeline to have independence from object texture and invariance to illumination. Figure 3.4 is an outline of the procedure of the algorithm used for the detection of the different positions of the objects in the scene, where several processes are carried out to increase its accuracy and reduce false identifications, to subsequently obtain the orientation of the object, with its corresponding refinement, from the reference orientation of the model.

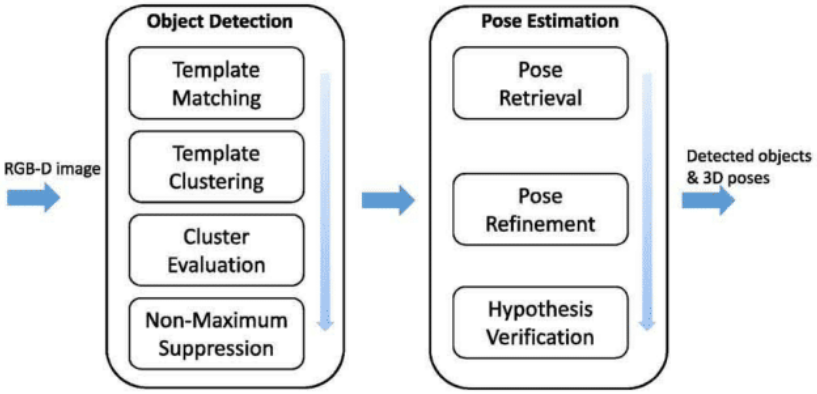


Figure 3.4: Pipeline proposal for 3D object recognition and pose estimation (He et al., 2017)

Starting from the RGBD image of the workspace and a previously elaborated 3D model in the same way of the object to identify, the method starts with the creation of bounding boxes that have a matching value exceeding a given threshold to subsequently eliminate the cases that differ from the trend by clustering these boxes. Finally, the use of a bounding box will produce from each cluster a single frame, thus obtaining the location of the sets of points that most closely resemble the model.

Once its location in the scene is located, an initial estimate of the position in space and its orientation is obtained, using the observable rotation values between different models of a cluster for the orientation and calculating the necessary translation from the difference between the coordinates. Finally, an ICP refinement is applied in which a voxelization is required to have the same density of points and employs a dynamic distance threshold proportional to the largest distance of each iteration.

(a) Obtained bounding boxes **(b)** Remaining bounding boxes after clustering **(c)** Detected objects in position and orientation

Figure 3.5: Visualization of real objects recognition and pose estimation (He et al., 2017)

The proposed pipeline is a highly efficient recognition method when working with a known object and does not require prior training as is required in applications involving the use of Deep Learning. That is why a predefined model of each object to be identified will be used to show more clearly the effectiveness of the pose estimation without the need for identification and clustering by bounding boxes.

The implementation of the above proposal proves to be effective in cases where the object to be identified is known today. However, when the correct use of a data set and the ability to identify that is obtained thanks to the training of a certain algorithm is enhanced, the result can be applied in environments where not all the information may be available and still operate correctly. Han et al. (2022) proposes a method for 3D scene reconstruction using functional objects, which are objects with specific functions or affordances, that can be used as landmarks to reconstruct the scene. The goal of the method is to provide a more practical and efficient way for robots to understand and navigate indoor environments. The method (see Figure 3.6) starts by detecting functional objects in the scene using a pre-trained object detector and extracting feature descriptors from the objects. The feature descriptors are used to generate a 3D point cloud of the scene, which can be used to create a mesh representation of the environment. The mesh representation can then be used by robots to plan paths and perform tasks in the scene. The proposed method is evaluated on two datasets and compared with other state-of-the-art methods for scene reconstruction. The results show that the proposed method is more accurate and efficient, especially when dealing with cluttered and complex environments.

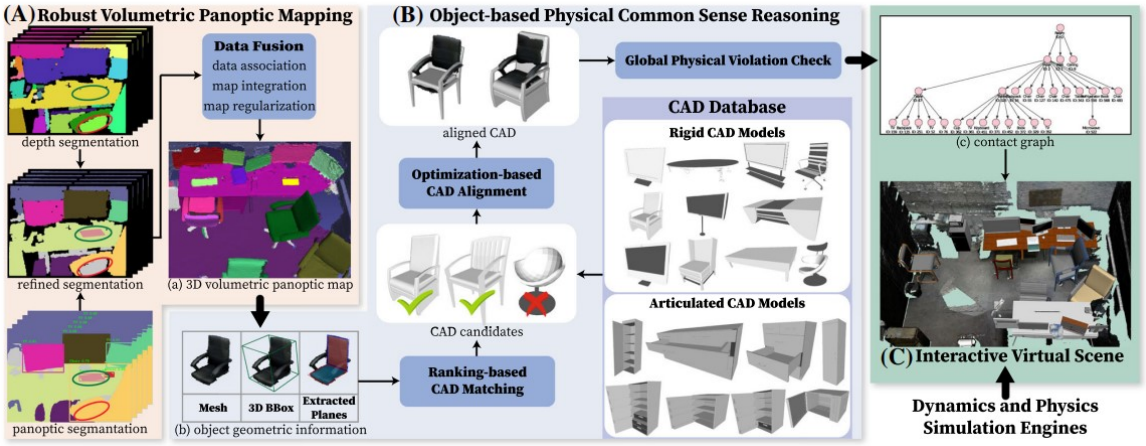


Figure 3.6: System architecture for scene reconstruction (Han et al., 2022)

It will be relevant research as it proposes a method for 3D scene reconstruction using functional objects, which can be useful for the thesis in terms of detecting and recognizing objects in the workspace. The proposed method can provide a practical and efficient way for the robot to understand the environment and plan paths for assembly tasks.

4. Methodology

This chapter will explain the importance of using a well-defined research methodology and outline how the selected methodology is adapted to this project, since it will serve as a framework for the project structure.

4.1. Research Methodology

The goal of scientific research in any field is to gain new knowledge through an empirical system. These results should also be able to be reproduced and cross-checked by other researchers and serve as evidence for future research (Schaller, 2016). Likewise, the Research Process for developing a problem-solving solution, as in this case, requires a systematic approach by which to collect and evaluate data rigorously to achieve a feasible solution.

This is why a methodology is required that suits the research so that a model, method, or artifact is created to address the previously defined problem. Therefore, this solution will result from a rigorous analysis in which the design is developed by assessing existing knowledge observed during this development phase. It should also include communication of the limitations of the study and the potential for further research (Peffers et al., 2007). Thus, the author provides a guided lecture to the reader by which to understand the steps being taken and the motivation for their actions. This enhances the author's credibility, rigor, and objectivity.

In design-oriented projects, several established methodologies are currently being used, so we will proceed to an overview of the steps that make up the selected methodology, highlighting how it can shape the thesis, thus presenting itself as a suitable framework.

4.2. Design Science Research Methodology

DSRM is a problem-solving paradigm that provides a set of structured guidelines for rigorous production, presentation, and evaluation in the study and design of an innovative artifact (Geerts, 2011). Its objectives are :

- Provide a nominal process for design science research.
- Set the basis of the study in the previous literature of the reference disciplines.
- Conduct the study guided by a structured mental model.

According to Peffers et al. (2007), DSRM consists in seven steps which can be described and set as the framework for the project in the following list:

1. **Identification of the problem and motivation:** The correct definition of the problem at a conceptual level is fundamental to determining the aim of the research and developing an effective solution. Since this identification arises from the study of the current situation through the State of the Art and by explicitly stating the limitations that exist, the author is showing his own perspective by determining an existing problem that he proposes to address, thus motivating both the author and the reader to the need to search for a solution.

   The identification carried out in *Section 1.4: Problem Statement* shows the limitation that exists in the industrial sector due to the lack of flexibility in robotic assembly systems, therefore, the proposal of implementing a vision-driven system seems to offer a solution to this lack. This identification is the result of the observation made in the previous section *Section 1.1: Background*. In turn, *Chapter 3: Theoretical Frame of Reference* and *Chapter 4: Literature Review* will provide more detailed information relevant to the development of the solution, although these will be carried out once the problem has been determined.

   Therefore, the thesis presents a problem-centered entry point, as the development of the rigorous solution starts from the discovery of a problem after the initial investigation. As shown in Figure 4.1, the starting point is not always the problem statement, but the one that triggers the need to apply the DSRM. This means that one could have started from a set of objectives, from a device to be incorporated into the solution, or from a solution reached without rigour.

2. **Define objectives:** Starting from the problem identified in the previous stage, the solution can be approached based on many proposals so that after a thorough study of the theoretical concepts involved in the current situation and, the different technologies that may be interesting for its resolution, different objectives are established on different aspects of the problem that can be feasibly solved. In addition, some identified problems have been considered but not set as objectives due to their nature, complexity, or effectiveness. Therefore, a framework is proposed to limit the conditions for the study of the proposal, i.e., selected feasible objects will not be the subject of study as the author has considered what could be addressed in a more advanced model of the proposal.

   The objectives proposed in *Section 1.3 Aim and Objectives*, illustrate to the reader the aspects of the problem that are proposed to be solved, i.e., the development of a system guided by an RGBD Camera and its implementation in the virtual environment of the studied robot. In turn, the study conditions set out in *Section 1.4: Delimitations* that establish the model of the workstation, the robot, and the items that it will be able to manipulate, among others, are shown.

3. **Design and development:** Now that the objectives have been clarified, the creation of an artifact to the problem posed and capable of providing a solution can take place. Its design will be composed of different desired functionalities, which may depend on each other and therefore require a structured design according to their needs. Since each functionality is intended to satisfy certain objectives, the result of this development will therefore be the solution to the problem posed.

   As mentioned in *Stage 1*, *Chapter 3* and *Chapter 4* are meant to be the main source of knowledge whereby each of the functionalities can be designed based on theoretical concepts and real implementations. Therefore, the whole design development will be detailed hereafter in *Chapter 5: Implementation*.

4. **Demonstration:** Illustrate the behavior of the device for the effective solution of the problem, making it easier for the reader to understand its execution. Case studies can be included in which different parameters will be analysed to evaluate aspects such as time and accuracy (Venable et al., 2017). From the different experiments that will be carried out in this *Chapter 6: Analysis and Results*, the effectiveness of the vision system that has been proposed is highlighted since it will clarify the most suitable conditions and parameters for this implementation.

5. **Evaluation:** After having carried out the experiments in the previous stage to determine the methods or parameters that produce greater effectiveness, it is time to compare the real results in the demonstration of the final artifact with the initial objectives. To do this, given the nature of the project, it can be evaluated whether the artifact is fully operational to solve the cases and objectives for which it has been designed. All results obtained during the evaluation stage will also be included in *Chapter 6: Analysis and Results*, as this evaluation is considered as another analysis, even though it has a more important nature.

6. **Communication:** Finally, after having obtained a final device capable of providing a solution to the problem posed, the problem that it has been decided to tackle and the importance of solving it in the sector in question, i.e., in automated assembly stations in industrial environments, is once again emphasized. As for the device itself, it is worth highlighting the novelty it represents in the sector and justifying the design finally achieved, focusing on the rigorousness with which this result has been achieved. The purpose of this thesis is certainly to transmit all this information. However, this stage is synthesized in *Chapter 8: conclusions*.

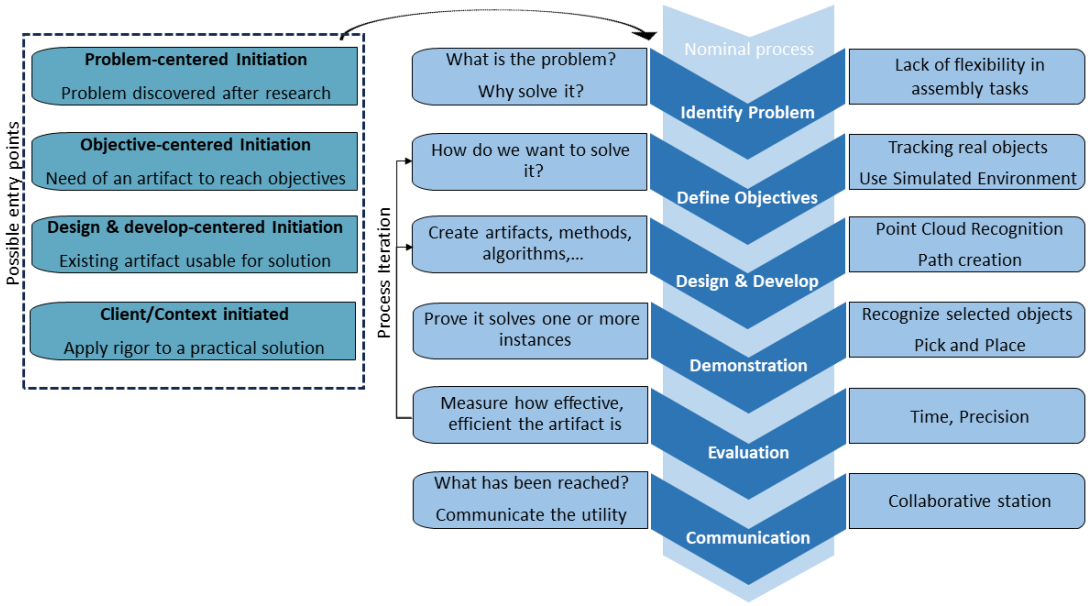


Figure 4.1: DSRM Process Model adapted from Peffers et al. (2007)

5. Implementation

After researching the existing theoretical and project processes relevant to this thesis, as well as the methodology to be carried out, it is possible to start the implementation. In this chapter, the parts that make up the implementation of the thesis will be written down. This will be mainly divided into the image processing of the real environment by computer vision and the creation of the virtual environment and trajectories in Robot Studio.

5.1. Wortkstation setup

The working area where all the equipment is located is ASSAR, a place where the University, in collaboration with companies from the technology sector, offers itself as a place where it can contribute to the research and development of innovative solutions for the industry. The most important elements that make up the list of necessary equipment are the Azure Kinect DK camera from Microsoft, the GoFa CRB 15000 Robot from ABB and the Co-act EGP-C 40 gripper from Schunk. The setup is completed with a series of items designed or adapted to the desired conditions of the real environment for the project in question (see Figure 5.8).

Figure 5.1: Robot and Workstation Setup

5.1.1. Gripper

Due to compatibility with the robot, the provided gripper, the Schunk Co-act EGP-C 40, has been fitted to the robot end-effector. Its connection enables a signal to be created to trigger automatic opening or closing from the controller. Two pads have been added to the ends of the gripper to offer more margin for error when picking the block (see Figure 5.2). They have a flexible but compact foam core covered with insulating tape to grip the gripper. The trapezoidal shape will prevent possible collisions when approaching the pieces.

Figure 5.2: Adapted Co-act EGP-C 40 gripper

5.1.2. 3D Blocks

The objects that have been selected to be analysed by means of the camera, modelled within the virtual environment, and finally manipulated by the gripper consist of two identical sets of blocks with different sizes, shapes and colours made by a 3D printer (see figure 5.3). The need to duplicate the set of blocks is due to the fact that one set will be handled by the operator while the other will be on the robot workspace.

Among its characteristics we can highlight that all the pieces have a width of 2 cm, as this is the length that the gripper will leave when it is in the ”closed” state and therefore they can be gripped. On the other hand, to facilitate the modelling of the pieces by means of the program found in the robot studio, the pieces have been designed with basic geometric shapes such as prisms or cylinders. Finally, each block has been painted with a distinctive colour to facilitate its segmentation in the point cloud of the rest of the elements.

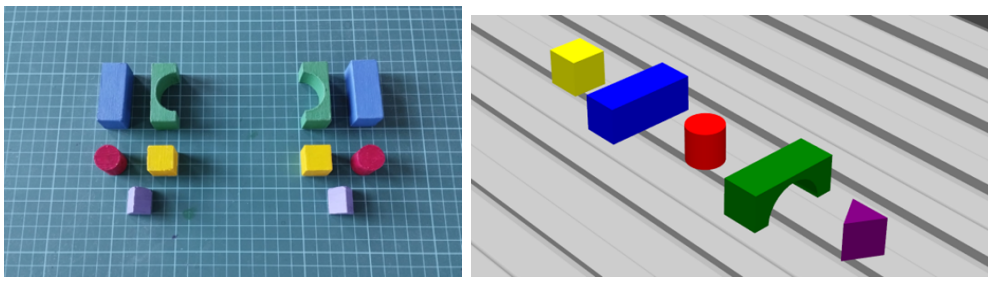


Figure 5.3: 3D printed blocks (left) and 3D model blocks (right)

5.1.3. Boards

The GoFa is already installed on a folded metal worktable. Both for its reflectance and its unevenness make it necessary to use a different surface that suits the working conditions of the camera. For this purpose, a board covered with white sheets of cardboard has been used (see Figure 5.4). Besides reducing the working space so that the resulting trajectories do not require excessively large turns, the white colour will allow the depth sensor to obtain measurements without providing noise, as well as offering a background that facilitates the segmentation by colours of the pieces that are on it. This 37x49 cm board is not in a fixed position with respect to the base of the robot, so it has 4 black dots located at 1 cm on each side that will serve to locate the board and establish a common reference system for the image recognition environment and for the virtual environment.

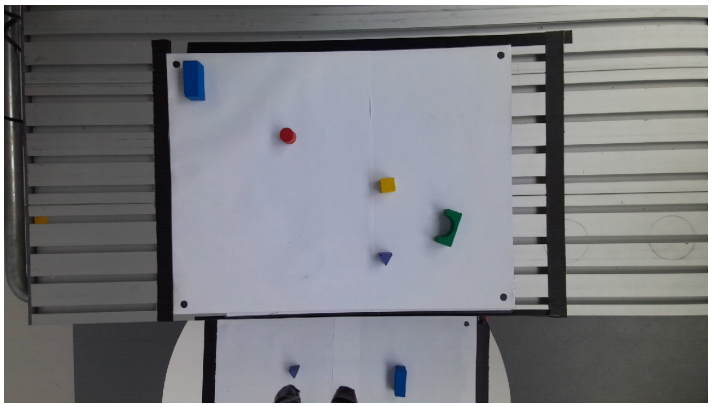


Figure 5.4: Board over robot workstation

5.1.4. 3D Blocks

Similarly, the user workstation, which consists of a non-fixed table, has a work surface with the 4 points located at the same distance between them, so that both reference systems will be established so the object recognition can work indistinctly. With respect to the height at which the camera will be placed for the capture, the robot will be at the same height at both workstations, although the recognition system will re-scale each capture so that the points that are at a known distance reflect that same distance during image processing.

5.1.5. Camera clamp

The camera must be located at a point where it can observe the entire surface and the parts on it are clearly identifiable, which is why an overhead camera view has been chosen. Regarding its set up, the use of an additional structure to support the camera from a fixed position allows a predefined calibration between the reference system of the camera and the work surface. However, this can obstruct the robot's working space as well as hinder the image capture by being between the camera and the table.

For this reason, an additional part has been designed and manufactured by means of a 3d printer with which to fix the camera to one of the furthest links of the robot, enabling an eye-in-hand camera (see Figure 5.5). This system also allows both workstations to be captured with the same camera. To this end, a robot position has been defined for each workspace. Therefore, the implementation carried out forces the user workstation to be in front of the robot within its workspace. A series of clamps are also included along the robot to allow flexibility of movement of the power and communication cables without interfering with the paths taken.

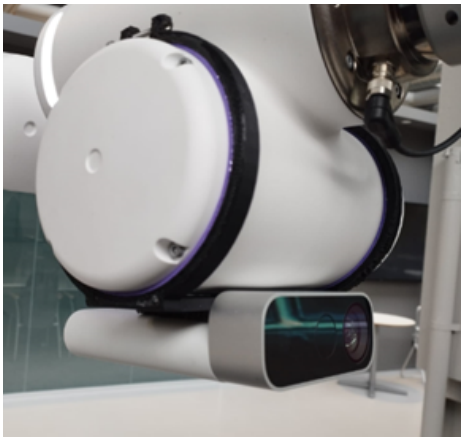


Figure 5.5: Camera clamp

5.2. Communication

As described in Figure 5.6, three interconnected environments within a private network named "ASSAR 2030" utilise TCP/IP sockets:

- Camera Data Analyzer: This environment functions as the client and establishes connections with the other two environments.
- Virtual Robot Controller: Located at IP address 192.168.0.229, this environment receives information from the camera data analyzer through port 8000. Its primary role is to control the visual simulation of the station autonomously, without the need for a physical robot. Workers interact with this controller with the help of a RobotStudio plugin to manipulate the robot's actions and monitor the overall process.

- Physical Robot Controller: Positioned at IP address 192.168.0.210, this environment also receives information from the camera data analyser via port 8000. Additionally, it receives signals that control the performance of both the virtual and physical robots from the Add-In (an external software module) in the RobotStudio environment via Ethernet.

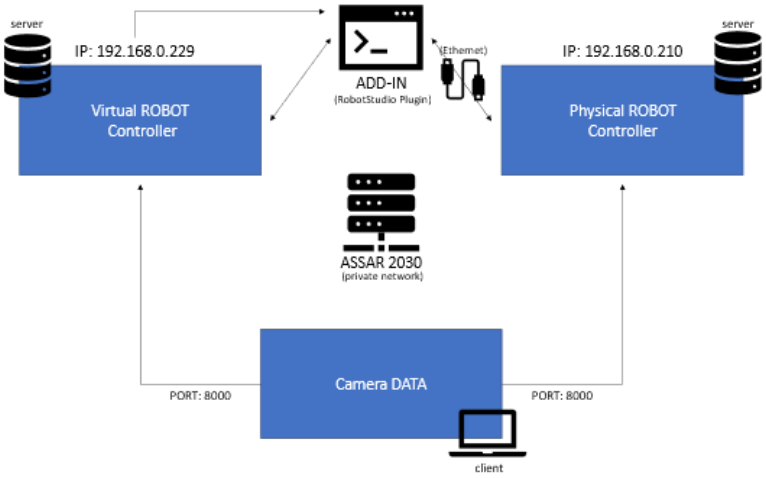


Figure 5.6: Communication chart

5.3. Visual data processing

5.3.1. Camera application

The integrated development environment used is Visual Studio, so initially, the Kinect Azure SDK installation for Python is required. In this way, we can use the libraries that allow us to develop an application with which to manage and access the data and measurements from the camera. This application establishes the connection with the device, configures the way in which it will capture the images and extracts the captures made along with other relevant values such as the dimensions of the image and the intrinsic parameters of the camera. From this moment on, the camera remains deactivated until a message received from RobotStudio indicates that another capture should be made. Finally, the camera is deactivated at the end of the complete execution and not before, since, for each capture, a reactivation and configuration of the device would be required, which makes it an inefficient manoeuvre.

During normal execution, it will initially be used to capture the initial state of the parts on the robot workstation to identify the number of parts on it and their position and orientation with respect to the work object table so that they can be replicated within the virtual environment. Subsequently, the operator workstation will also be captured to determine the new position of the parts to be replicated by the robot. In the same way, each time the operator manipulates a part, the following capture will be made without the need to visualise the robot station again since the virtual environment already knows the last position.

5.3.2. Image processing

From each capture, the 2D projection of the scene in colour, the measurement of the distance of the objects with respect to the device and all the attributes inherent to each capture are extracted from the camera. The intrinsic parameters of both sensors and the relationship between the reference system of both sensors are also taken, although only in the first capture since they are invariant. These two images will then be saved in the appropriate format to store for each pixel 1 channel for the distance in millimetres of the Depth image or 3 channels for the RGB values of the colour image.

Note that due to factors such as the resolution of the sensors, the field of view or the position in which they are integrated in the camera, the two images do not show completely the same points of the scene, i.e., there are points which are visible at one end of one image but not at the other end. This is why a transformation of one image to the viewpoint of the other sensor from the extracted ratio is required to align the two images, making the transformed image the same dimensions as the other image. For this thesis, the acquisition of a higher resolution depth measurement has led to the decision that the alignment will be done by transforming the RGB image to Depth image due to the camera resolution (1920x1080) will allow providing more points for the point cloud than the depth sensor can provide with the Narrow Field-Of-View mode (640x576).

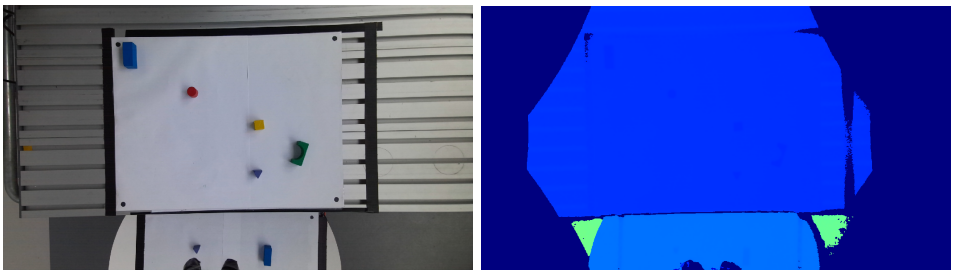


Figure 5.7: Color (left) and Depth (right) captures from camera

Once aligned, these can be combined to give rise to an RGB-D image of the scene and finally, considering the intrinsic parameters of the selected sensor to undo any distortion made by the lens, you have the desired point cloud and can visualise a 3D representation of the scene.

5.3.3. Reference System

The process to determine the reference system (see Figure 5.8) is as follows:

1. Start from the previously captured colour image and transform it into grayscale.

2. Use a threshold on the previous image to obtain a segmentation by identifying the darkest points. The use of an adaptive threshold will help to solve problems related to illumination and noise, which can consider the black point and the blue block as a single segment due to the shadow cast on the point.

3. Apply a blur to eliminate isolated points caused by thresholding and to help to smooth out sharp edges or fine details in an image. This will allow to obtain strong and well-defined circles shapes.

4. Apply the Hough circle transform to identify circular-shaped segments with given dimensions and characteristics.

5. Assign each of the four circles its corresponding point based on its distance from the upper left corner, i.e. the origin of the image (width = 0, height = 0). The 4 points will then delimit the space in which the pieces are located, there are points with specific functions.

6. According to step 5, the point P0(x0, y0) will be the centre of the reference system or work object, the line joining the points P0 and P1 establishes the orientation of the X axis and the line joining the points P0 and P1 establishes the orientation of the Y axis, thus extracting the 2D reference system from the colour capture.

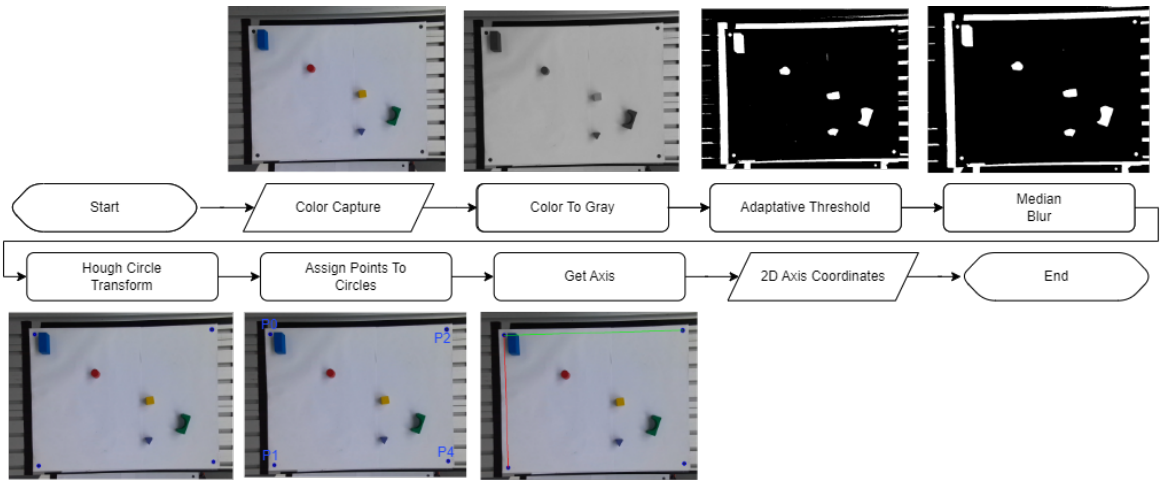


Figure 5.8: Reference system processing workflow

5.3.4. Pointcloud creation

Once you have both captures with the same resolution and the work object, the creation and processing of the 3D point cloud can proceed (see Figure 5.9). This process will also include some visualisation of the different stages at Figure 5.10.

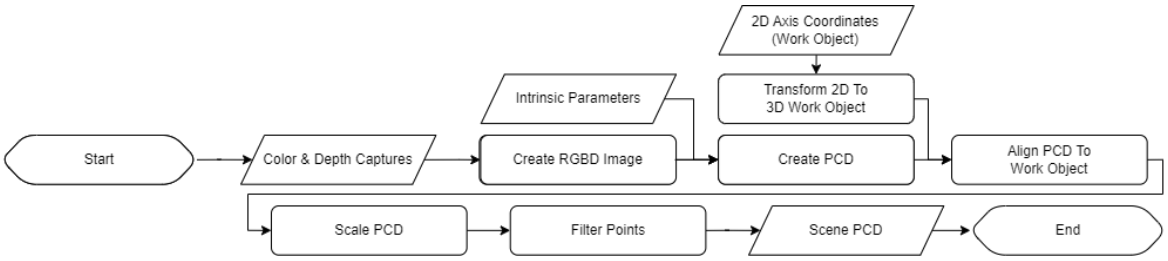


Figure 5.9: Pointcloud creation workflow

1. Create an RGBD image from the two captures, thus combining the colour and depth data in the same file.

2. Transform the RGBD image into a 3D point cloud. Despite having the metric depth distance for each point, the positioning of the points into a 3D world coordinates, and the intrinsic parameters of the Camera are necessary as it defines the relationship between the object's depth and its apparent size in the image. The resulting Pointcloud (PCD) can be seen in Figure 5.10a

3. Align the PCD according to the 2D work object. Once we have the coordinates of the centre of the 4 points, we take the value of the depth of the Depth image that is in those coordinates and we calculate with the intrinsic parameters of the Camera its position in space measured in metres. The point cloud has as a reference system the lens of the color camera (B), so that, by knowing both centres of the different reference systems, the PCD can then be translated in such a way that $P_0(x_B, y_B, z_B)$ of our reference system (A) becomes the origin of coordinates, i.e. $P_0(0, 0, 0)$.

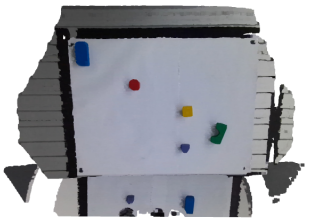
$$^{B}Transform_{A} = Translation \begin{pmatrix} x_B \\ y_B \\ z_B \end{pmatrix} \cdot Rotation(X, \alpha) \cdot Rotation(Y, \beta) \cdot Rotation(Z, \gamma)$$

   To orient the reference system, the vectors X and Y are created with the help of the points P1 and P2, respectively, while the vector Z is the result of the vectorial product of the two previous normalised vectors. From these vectors, it will be possible to determine the Euler angles for each axis (α, β, γ) which will indicate the rotations necessary to align the PCD to this reference system (see Figure 5.10b).

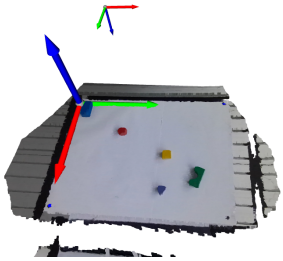
4. Scale the PCD from the known distances between the 4 points in case there are differences between the real measurement and the one obtained in the previous step. As

shown in Figure 5.10c, among the possible states that we could create the PCD, represented by each set of the four coordinates located at different heights, although all sets of points keep the ratios of the image, the one that will be used has to verify and ensure that the width of the table in the PCD is equal to the real measurement, doing the scaling in relation to both values.

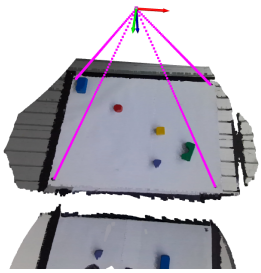
5. Filter the point cloud by distance. Starting from the origin of the reference system, the point cloud can be filtered by selecting only the points between the origin and a given distance for each axis. On the X-axis it would be the known distance between points P0 and P1, on the Y-axis the distance between points P0 and P2 and on the Z-axis, although a copy of the PCD today showing the table will be kept, we will work with a point cloud whose filter has been set to 1 cm above the table, i.e. only the upper half of the blocks are visible (see Figure 5.10d). This step is intended to optimise subsequent point cloud calculations.



(a) Original PCD



(b) PCD with camera (top) and board (bottom) reference system



(c) PCD with possible and final scale

(d) Filtered PCD

Figure 5.10: Pointcloud creation stages

In this way, the point cloud of the scene to be analysed will have been obtained. To identify each of the objects, a dataset has been created with a partial model of each part. For this case, a snapshot was taken in which only one of the pieces was located at a known point like P3 and with an orthogonal orientation. From the generated point cloud, we proceed to obtain in detail all those points that are part of the block to have a point cloud with the isolated object. Subsequently, as the distance between P3 and the origin of the coordinates of

the board is known, the part is translated to this point so that subsequent pose estimations can be measured for rotation and translation with respect to the origin. In this way, we have a point cloud with which we can identify the position and orientation of the block in any scene by means of the origin, values which can be sent to the virtual environment so that the robot performs the corresponding path for the manipulation of the object.

5.3.5. Object Segmentation

During the development of the implementation, a snapshot was taken in which only one of the pieces was located at the origin of the reference system of the table and with an orthogonal orientation. From the generated point cloud, we proceed to obtain in detail all those points that are part of the block to have a point cloud with the isolated object. In this way, we have a point cloud with which we can identify the block's position and orientation in any scene through the pipeline.

Figure 5.11: Isolated green piece PCD

The importance of having isolated the object with this certain position and orientation allows the transformation resulting from the pipeline to indicate the rotation and translation that must be performed to reach its current position from a reference system known by the robot, so that the values of this transformation matrix will be sent to the virtual environment so that the robot performs the corresponding path for the manipulation of the object. For each capture, two-point clouds will be worked together: the point cloud of the scene and the point cloud of the isolated object. The following process will be repeated for each object that has been segmented and isolated in a point cloud within the dataset itself.

5.3.6. Object recognition and pose estimation pipeline

At the beginning of an execution, the robot will be oriented by the controller to allow the capture of the scene at the robot workstation. As this is the first capture, there are no parts in the virtual environment, so at first it is necessary to determine which parts are in this station. This identification is done by segmenting the scene by colour, so it is only necessary to detect the existence of a set of points of the desired colour to indicate to the virtual environment that the identified block should be created. If so, the point cloud of the data set of the block in question will be taken to start the pose estimation process.

The process starts with a voxelisation of the point clouds to reduce the number of points by sacrificing part of the resolution in exchange for a significant reduction in computational cost, as well as redundant points that may cause errors in pose estimation. It is important to note that this step and the following ones will be carried out on both point clouds, as

having a significant imbalance in the number of points can lead to an uneven distribution of correspondences, which can lead to loss of precision in the pose estimation. Next, a series of points known as keypoints are extracted, i.e., a subset of distinguishable scene points that define the surfaces with a significantly smaller number of points. From these points, geometric patterns can be encoded considering the neighbourhood of each point to calculate descriptors. Subsequently, the descriptors of each point cloud are compared with each other considering each descriptor together with those around it, thus calculating pairings between both point clouds. With the information provided by this 33-dimensional vector showing aspects such as position, colour, and the relationship to its neighbours, point clouds can be compared to achieve a registration based on feature matching between the two-point clouds(see Figure 5.12).

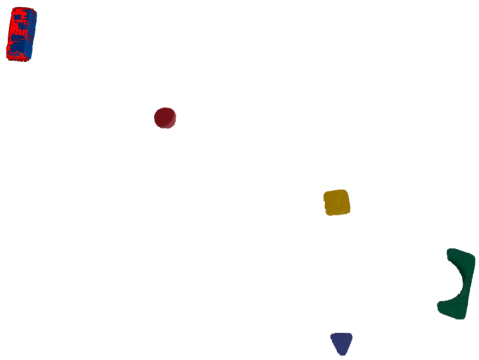


Figure 5.12: Transformation alignment after pose estimation

In this way, all the parameters about rotation and translation can be found in the Transformation Matrix, from where the parameters required from RobotStudio are obtained in a compact 4x2 matrix as seen in Figure 5.13 with the following values:

- Idx: Index to specify the type of parts we are dealing with, with 1 being the yellow cube indicator.
- Tx, Ty, Tz: Translation values with respect to the coordinate system
- S, Rx, Ry, Rz: Components of the quaternion representing the scalar value of the angle of rotation (S) and the vector defining the axis of rotation (Rx, Ry, Rz).

```
[[ 0.99999873, -0.00174533,  0.00115661,  -0.016  ]
 [ 0.00174533,  0.99999315, -0.00338068,  0.335   ]
 [-0.00115661,  0.00338068,  0.99999315,  0       ]
 [ 0.        ,  0.        ,  0.        ,  1.      ]]
```

```
[[+1.000, -0.016, +0.335, -0.000],
 [+0.874, +1.290, +0.127, +0.254]]

[[ Idx,   Tx,   Ty,   Tz],
 [   s,   Rx,   Ry,   Rz]]
```

Figure 5.13: Transformation matrix (Left) and Information Matrix for Robot Studio (Right)

Once the information matrix has been sent to RobotStudio, it will be able to create the given part in the same position and orientation while the pipeline continues to perform this

process of identification and pose estimation for each of the remaining parts in the dataset. At this point, once all the parts have been positioned in the virtual environment, the real robot moves the camera to capture the worker workstation scene to be analysed by the pose estimation algorithm with only estimate the pose of the parts that were on the previous workstation.

After completion of the pick and place task, these new positions shall be stored as the current positions while the robot is back at the user station waiting to be indicated that a new capture of the user table is required. Note that the subsequent operator table captures will take into account the previous position matrix to assess whether the piece has moved or not. If so, the pick and place task will be performed on those blocks handled. This is because each block, besides it is known the matrices of its current position in both stations, it has a flag to identify its change without the need for additional computation.

5.4. Robot programming

The Robot Controllers serve as a crucial component in the system, responsible for managing and controlling the behaviour of both the virtual and physical robots. These controllers are programmed via RobotStudio, which provides a virtual environment where users can design, simulate, and program ABB industrial robots. In addition to simulation, RobotStudio also enables offline programming, which means users can develop complete robot programs without the need for physical access to the actual robot.

5.4.1. Simulation Environment

RobotStudio offers a range of windows that streamline robot programming, simulation, and configuration. The Station tab serves as the central window, displaying the virtual robot station environment. The Controller tab simulates the robot controller's behavior, enabling program monitoring and modification. In the Program tab, users can write and manage robot programs using the Rapid programming language, a high-level, task-oriented language specifically designed for ABB robots. The Motion tab provides tools for defining and simulating robot motion. The Task tab facilitates the creation and management of complex task flows. With the Layout tab, users can arrange and configure the robot cell layout. Additional windows, such as the 3D view, I/O view, signal view, and message log, enhance visualization and diagnostics. These windows collectively empower users to efficiently design, program, simulate, and analyse robot systems within the RobotStudio environment.

As is shown in Figure 5.14, using RobotStudio, you can create a simulation of a real station by adding various components available in the RobotStudio libraries and external ones. These components include robot models, robot tables, grippers, cameras, and other peripheral devices. You can choose the specific model of the robot from the library that matches the real robot being used in the station. By placing and configuring these components within the virtual environment, you can replicate the physical setup of the station in the simulation.

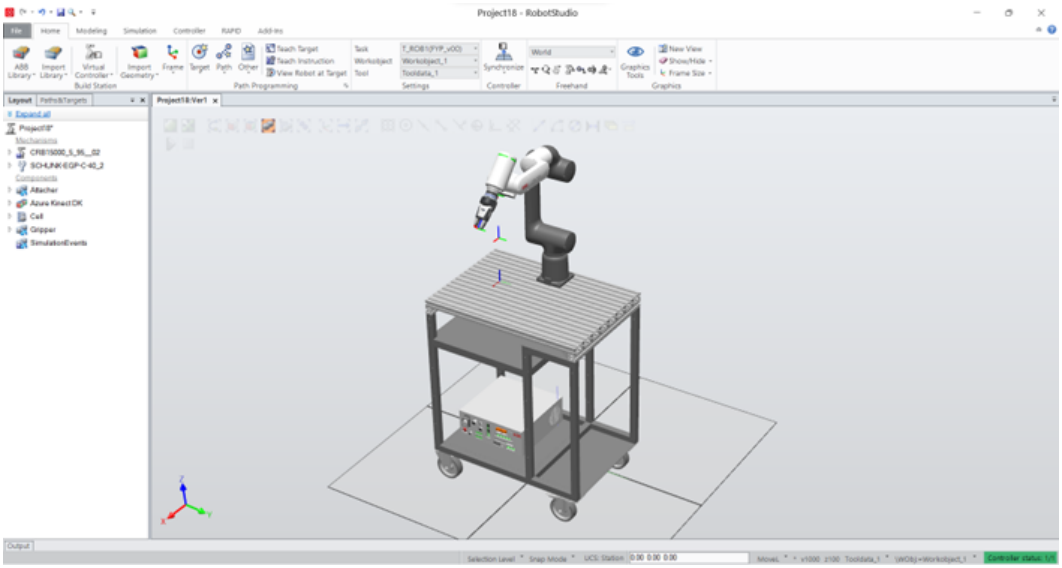


Figure 5.14: Main Tab RobotStudio Snapshot

To ensure accurate operation, the TCP of the robot and the work object need to be calibrated. The TCP calibration involves determining the precise location and orientation of the tool tip relative to the robot's coordinate system. This calibration ensures accurate positioning and movement of the robot end effector. Similarly, the work object calibration involves defining the position and orientation of the object that the robot interacts with in the station. Calibration is typically done using calibration procedures provided within RobotStudio, which guide the user through the necessary steps to accurately define the TCP and work object.

In the programmed simulation (see Figure 5.15), there are three main tasks:

- COMMS: This task involves communication between the camera analyser and the robot controller. Socket functions are used to acquire information processed by the Add-In the T_ROB1 task. The communication allows data exchange and coordination between the camera analyser and the robot controller.

- T_ROB1: This task controls the robot movements within the virtual controller. It assists the Add-In with visualization and execution of various procedures based on active signals. These procedures may include tasks such as piece recognition, data conversion, piece calibration, and more. T_ROB1 plays a crucial role in coordinating and executing robot movements based on the information received from the camera analyser and other components.

- DATA: This task shows throughout the Operator Window the information about the pieces that are tracked by the camera, XY and Z Rotation. Besides, it helps the controller to handle with the different conversions and updates for the pieces information.

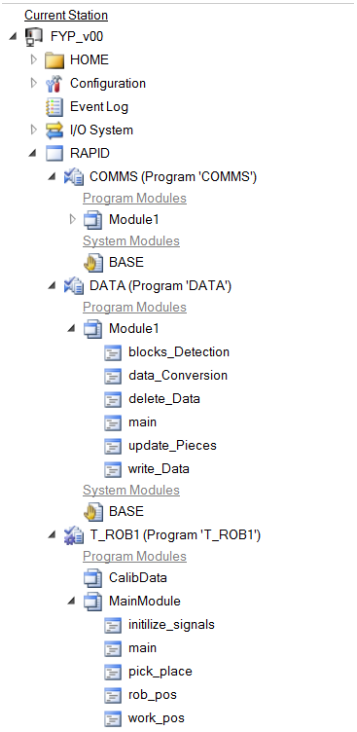


Figure 5.15: Task, Modules, and Procedures

5.4.2. Add-In

As it is mentioned before, some information for the visualization and signal activation is computed by an Add-in. Add-in refers to a custom software module or plugin that extends the functionality of the RobotStudio environment. It allows users to incorporate additional features, tools, or capabilities that are not native to RobotStudio by default. Add-ins in RobotStudio are created using programming languages such as C# and can interact with the RobotStudio application through its APIs (Application Programming Interfaces). These APIs provide a set of methods, properties, and events that allow developers to access and manipulate various aspects of the RobotStudio environment. Figure **??** it is shown an example of the RobotStudio APIs that are used.

```
ABB.Robotics.Controllers;
ABB.Robotics.Controllers.Discovery;
ABB.Robotics.Controllers.RapidDomain;
ABB.Robotics.Math;
ABB.Robotics.RobotStudio;
ABB.Robotics.RobotStudio.Environment;
ABB.Robotics.RobotStudio.Stations;
```

Figure 5.16: RobotStudio APIs

In summary, ABB.Robotics.RobotStudio is the software used for offline programming and simulation of ABB robots, while ABB.Robotics.Controllers allows for communication and control of ABB robot controllers. ABB.Robotics.RobotStudio.Stations API provides functionality for working with RobotStudio stations, enabling developers to interact with the virtual environment and manipulate station components programmatically, like a virtual robot workspace and contains components such as robots, parts, work objects, and programs. Subsequently, the implementation of this Add-in has facilitated the creation of a new tab called "Vision Assembly," which supports various tasks, including visualization, component creation, environmental localization, and signal activation for both the virtual and physical robots. Figure 5.17 showcases a program screenshot, illustrating the presence of different buttons that enable piece creation, worktable calibration, and connection management.

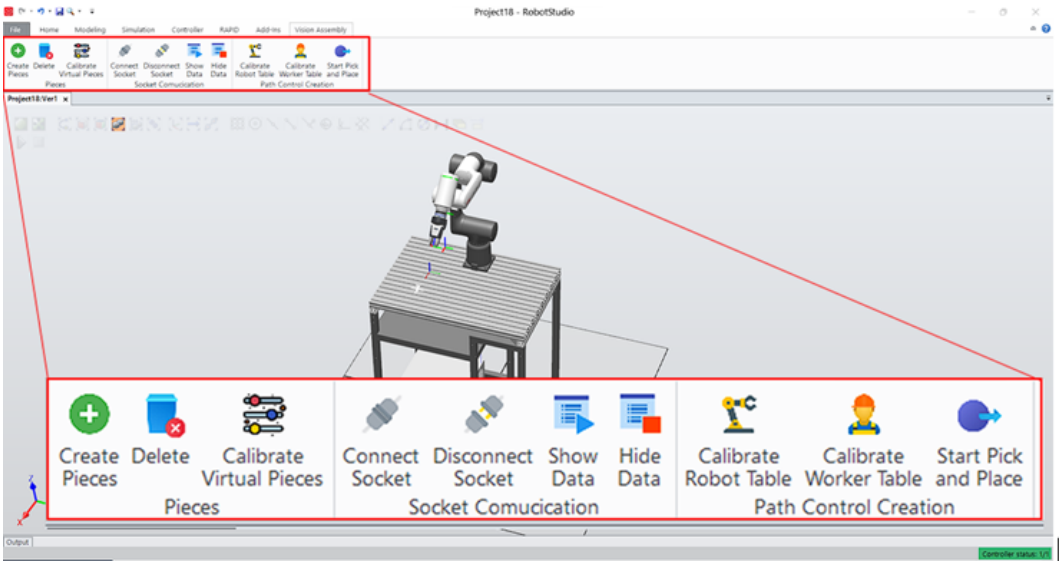


Figure 5.17: Add-In Tab Buttons

5.4.3. Virtual Robot & Physical Controller

As mentioned earlier, both the physical and virtual robots are controlled through their respective controllers. The controllers have identical programming, with minor signal modifications. This allows for the development of a versatile Add-in capable of replicating virtual actions in the physical environment, while accommodating signal adjustments. In our case, signal adaptation is required for functions such as opening and closing the gripper. By maintaining a consistent programming structure and making necessary signal adaptations, seamless integration between the virtual and physical robot controllers is achieved.

In this way, both the physical and virtual robots can operate simultaneously without the need for direct connection. This enables us to test different configurations of the robot and camera without relying on the real robot, providing a safer development environment for experimentation, subsequent result analysis, and implementation of new functions. By

decoupling the virtual and physical systems, we can conduct thorough testing, iterate on designs, and ensure the stability and reliability of new features before deploying them in the real-world setup. This approach enhances the overall development process, allowing for more secure and efficient experimentation, analysis, and implementation of enhancements.

6. Analysis and Results

The outcome of this thesis has been the implementation of a set of programs that, in conjunction with the described communication, enables the ABB GOFA 15000 robot and Microsoft Kinect Azure DK camera to perform a pick-and-place process in both real and virtual environments. These programs allow the user to visualize a digitally simulated environment that can be teleoperated, enabling the operator to interact with digital objects and the station itself. To demonstrate the functionality and utility of this project, several case experiments were designed and conducted.

6.1. Camera Calibration

Continuing with this aspect seen in the implementation, the calibration between the camera and the gripper is not carried out directly but through the board itself, on which both the point cloud of the scene and the segmented objects have as reference system the main corner of the table, thus allowing the resulting transformation to be a transformation relative to the same work object that is in the virtual environment, as well as the work object that has been defined in the station of the real environment.

As far as the calibration of the camera itself is concerned, no external calibration was finally carried out as the conditions in which this implementation was found did not require a need for greater precision. In addition, the SDK provided by Microsoft has allowed the obtaining of any parameter of the calibration of the Camera relevant to the project.However, as an experiment, a chessboard has been used to compare between the intrinsic components extracted through the SDK and those obtained through this method. By camping a series of images of the flat board with a pattern of known dimensions from different perspectives, a relationship can be found between the projections of the corners of the board on the image and the actual measurements, thus determining the intrinsic parameters of the camera. This process requires a sufficient number of images and from different perspectives, either by moving the camera or by moving the chessboard. For this we will use a chessboard of 9x6 squares of 3cm side and a function to make 15 captures with a waiting time between captures of 0.2 seconds.

When one or several captures are made without perspective changes, this measure cannot be accepted as it lacks sufficient information to determine its correctness. In first experiment, the 15 captures have been taken with both the camera and the table static. When the chessboard is reoriented to obtain 15 different random perspectives, the result (see figure 6.1) provides us with a lot of information.

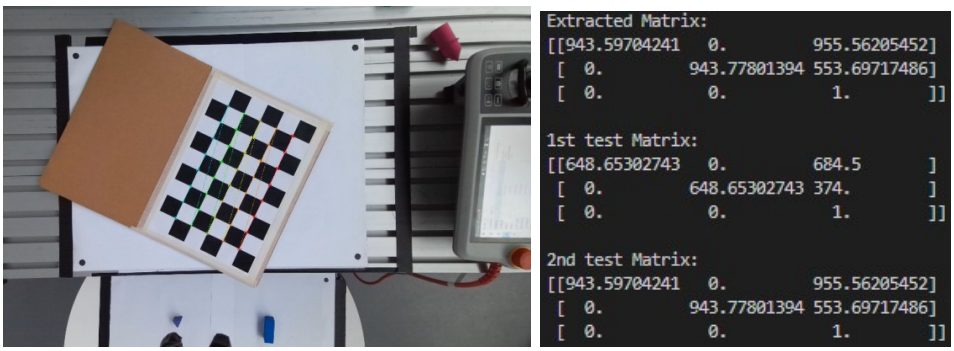


Figure 6.1: Calibration sample with chessboard (left) and obtained Extrinsic parameters (right)

Naturally, the first experiment is far from reality and should be discarded. On the contrary, the second experiment shows a better similarity since it indicates that the measured principal point (955.562, 553.697) agrees with the camera value (955.677, 552.134), while the measured focal lengths of x and y (943.597, 943.778) differ slightly (921.360, 921.394). This error may be due to the fact that the samples have been taken with an irregular background or that there is a scaling or distortion that has not been taken into account.

6.2. Camera Capture Settings

Through the Azure Kinect SDK, there are a number of features that can be specified or modified by the developer depending on the needs of the task. Some of these properties will keep their default value, such as setting the delay between colour and depth capture to 0, or no External sync since not more than one camera is being used. However, there are certain parameters that have been modified taking into consideration the workstation to be captured and the subsequent processing of the point cloud. Some of the concepts and values to be mentioned below can be seen in Appendix C.2:

- Frames Per Second (FPS): The captures used in the implementation consist of point snapshots, so, among the options of capturing at 5, 15 or 30 FPS, the one that offers a higher image quality versus a faster capture will be used, so FPS5 will be used.

- Colour Format: the colour spaces in which the snapshots can be taken are BGRA, YUY2 and YUV. While the latter two modes use compression techniques to reduce the colour information making them more suitable for streaming video, BGRA retains a higher fidelity of true colour. It also stores each channel independently, allowing quick conversion to RGB format by removing the Alpha channel and permuting the Blue and Red channels.

- Depth Mode: The Depth camera can capture distances with either NFOV or WFOV. NFOV mode allows a more detailed capture of a specific region while WFOV mode can capture a broader region of the scene at the cost of a lower resolution. At the same time, NFOV shows greater compatibility with the field of view that the RGB camera has, allowing less information to be lost in the transformation from depth to colour

while obtaining the RGBD image, which is why this will be the method used. The use of this mode also implies that the Camera must be placed at a distance of at least 0.5 m with respect to the objects to be captured, which will force the capture to show points that are not necessary, that is to say, any point that is outside the limits of the table. Moving the camera closer than this limit in order to have more useful points can lead to depth distortion problems.

- Color Resolution: The color camera allows captures at different resolutions, leading to a variance in the number of points in the pointcloud generated and the processing time, i.e. the higher the resolution, the higher the number of points and the longer the processing time. To determine the resolution to be used rigorously, the ratio between the two values will be considered and the z-score of this ratio will be calculated.

$$Z - Score = \frac{Value - Mean}{\text{Standard Deviation}}$$

The z-score shows how far a measure is from the mean in terms of standard deviations, with a positive result being a measure above the mean. For this case, a positive value for the points/time ratio indicates that more points are processed in less time, thus the resolution allows for more efficient processing. This is why the lowest resolution with a positive z-score will be taken to be the least time-consuming. As can be seen in Table 6.1, although there are high-resolution images with much higher z-scores, the captures will be taken at 1080p as this resolution is considered efficient and requires less time than the rest.

Resolution	**Capture and** Transform [ms]	**Create and** Filter PCD [ms]	**Points in PCD**	**Points/Time**	**Z score**
720p (16:9)	1689	755	715383	292,7099	-1,2897
1080p (16:9)	2140	1040	1952267	613,9204	0,0048
1536p (4:3)	2491	1690	2158102	516,1689	-0,3891
1440p (16:9)	2341	1866	2797886	665,0549	0,2109
2160p (16:9)	3429	3260	6526800	975,7512	1,4631
3012p (4:3)	3856	6500	10033911	968,8983	1,4355
		Mean	612,7211		
		Standard Deviation	248,1233		

Table 6.1: Capture Processing And Time Analysis Based on Resolution

6.3. TCP and Work Object Defining

Accurate definition of the Tool Center Point (TCP) and Work Object is essential for precise and efficient robot operations. The TCP determines the exact location and orientation of the tool, enabling the robot to perform tasks with accuracy. The work object defines the reference frame for objects, ensuring accurate perception and manipulation. These definitions enhance

robot performance, minimize errors, and enable reliable and safe operations. In RobotStudio, adding and defining a tool is a crucial step for accurate robot operations. In the project, the models of the gripper were added and then it was configured as a mechanism tool, you can specify its properties and define the TCP for precise positioning (see Figure 6.2). Associating the tool with the robot enables a seamless integration. For this purpose, a 3D part has been designed to facilitate the identification of the TCP.

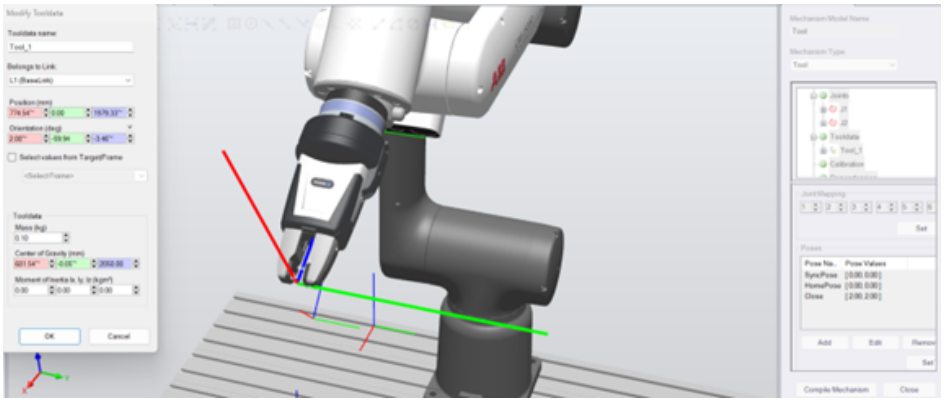


Figure 6.2: TCP Calibration Tool from RobotStudio

Once defined the TCP and imported to the real controller, it will define a work object using the 3-point definition method, follow these steps. Firstly, ensure the work object is properly positioned and the tool is attached. Access the work object definition settings through the appropriate interface. Select the 3-point definition method and specify three reference points by manually guiding the robot's end-effector (see Figure 6.3). Confirm the positions and allow the system to calculate the necessary parameters. Verify and fine-tune if needed. Finally, save the work object definition for future use.

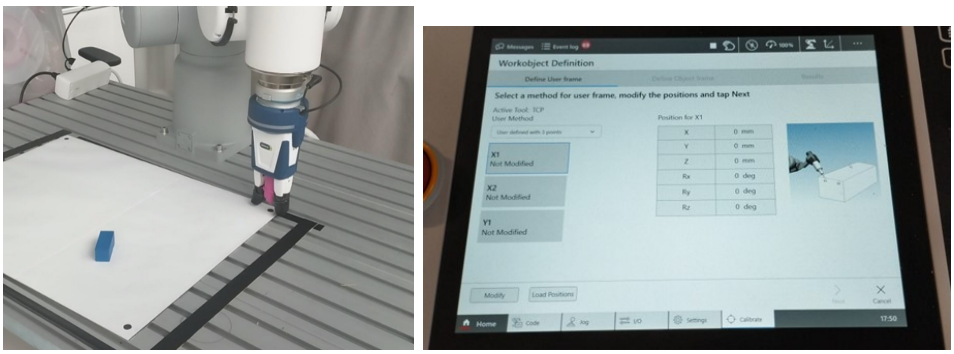


Figure 6.3: WorkObject station calibration (Left) and Workobkect Definition (Right)

6.4. Execution Program Example

Once all the programs are running, experimentation can begin. First, the connections between the programs discussed in Chapter 5.6, the communication is established: the Camera data analyser, the physical controller, and the virtual controller. With all the connections in place, the execution may proceed.

The Camera initiates its recognition programs, processing the gathered information. The results, including the recognized piece's location and orientation relative to the defined workspace origin, are then sent through a Socket to both the physical and virtual controllers. Upon receiving this information, the controllers and the Add-in process and handle the data. The operator, using the window created in the virtual robot program, takes charge of monitoring the experiment's progress. Then the operators' tasks are activating initially the Add-in in RobotStudio before running the programs in both virtual and physical controllers as well as the Visual Studio environment. Within the Add-in, as seen in Figure 6.4,the following steps are followed:

1. Create virtual pieces ("Create Pieces"): Various virtual piece models are generated in the virtual station.
2. Establish the connection between the controllers and the Camera ("Connect Socket").
3. Position the robot for robot table calibration ("Calibrate Robot Table").
4. Once all the piece data on the board is received, calibrate the virtual pieces ("Calibrate virtual pieces").
5. Proceed to calibrate the Worker Table ("Calibrate Worker Table"): There the worker change the position of the pieces in the Table meanwhile the camera is recognizing these changes.
6. After calibrating the Worker's table, the Robot performs various movements with the pieces, initiating the pick and place process on the robot table ("Start pick and place").

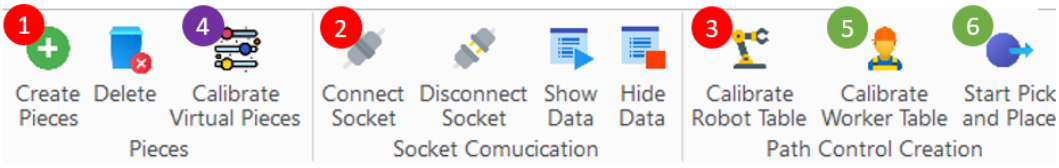


Figure 6.4: Add-In Buttons

This process can be repeated indefinitely through the recursion of steps 5 and 6. After completing all the previous steps, translations in XY and RZ of each piece can be shown/hidden on the operator's screen ("Show/hide data"). If any failures occur during the experiment, all programs must be restarted from the beginning by placing the pointer at the start of each program. Additionally, all existing pieces in the visualization should be deleted ("delete"),

and the process should be restarted from step 1. In a newer version of the Add-in, all these buttons have been deleted so all the process is made autonomously without the interaction of an extra operator. Otherwise, we include this old version to better understand the processes while the program is executing.

6.5. Time measuring

Analysing a series of runs in which the 5 pieces were found to be manipulated, we obtain, as shown in Figure 6.5, the following values:

- Group 1 - Initial settings: The initial set of functions such as initial configuration of the camera and virtual environment, updating the reference system and deleting parts from the previous run and establishing socket communication occurs in both programming environments simultaneously. The Visual Studio execution is shown in green while The RobotStudio execution is shown in blue.
- Group 2 - Data capture and PCD creation: Image capture (1689 ms), identification of the workobject coordinates (163 ms), creation of the point cloud (97 ms) and its processing and alignment (915 ms) follow, while RobotStudio is waiting for the transformations of each part. It highlights the power of Open3D for data processing and the need to reduce the number of points to facilitate the following process.
- Group 3 - Block recognition and pose estimation: In the process of identifying pieces by RGB filtering, the process takes 46 ms on average for all blocks. As for the pose estimation process, although the registration is 150 ms, the preprocessing increases the time. In turn, in order for all of them to have an average error of 9e-06, i.e. the average distance between each pair of points between the two pointclouds is less than 9e-06 meters, some of these processes are executed repeatedly by averaging the different transformations obtained until the resulting transformation allows the object to reach the threshold. Due to their different dimensions and more or less unique shapes, the process can be repeated 2-5 times for yellow (890 ms), 1-3 times for blue (600 ms), 1-2 times for red (400 ms), 1-2 times for green (480 ms) and 4-6 times for purple (1800 ms).
- Group 4 - RobotStudio update: After obtaining the matrix of each piece, it is communicated to RobotStudio to reposition the blocks in the virtual environment, while the robot moves so that the camera is in position to capture the operator's workstation.
- Group 2, 3 and 4: After a manual indication that the operator has finished handling the pieces, the groups above mentioned will be re-applied to analyse the operator's workstation.
- Group 8 - Path creation: After extracting the position and orientation values from the communication, the trajectories to be followed are determined.
- Group 9 - Pick and place: Finally, the routes to move each of the pieces are carried out. This can vary significantly depending on the number of pieces to be handled and their distances from the new positions.

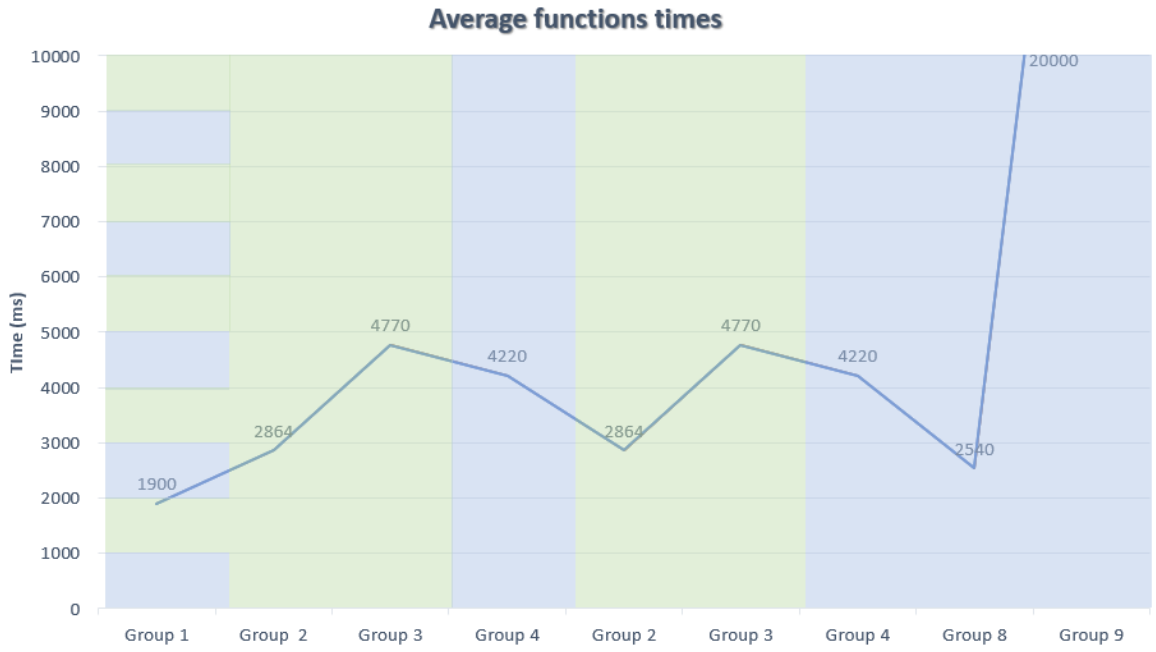


Figure 6.5: Average functions times

6.6. Result analysis

Once the initial success in the robot's performance is achieved following the described steps, it is important to continue the iterative process to further enhance its capabilities. This involves carefully analysing the obtained results and identifying any potential issues or areas for improvement. By closely monitoring the experiment and observing the robot's behaviour, valuable insights can be gained to guide the optimization process.One key factor in achieving optimal robot performance is camera calibration. The calibration process determines the intrinsic and extrinsic parameters of the camera, which are necessary for accurately interpreting the visual information it captures. Accurate perception is crucial in navigation tasks, enabling the robot to effectively detect obstacles and plan collision-free paths. Another key aspect that can contribute to the improvement of robot performance is the proper treatment of data related to the pieces being handled. The accuracy and reliability of the data play a significant role in the robot's ability to correctly identify and manipulate objects.

In summary, the integration of precise data treatment, accurate object recognition, reliable localization, and proper calibration of the TCP and work object leads to a significant improvement in the overall performance of the robot. These combined efforts enable seamless interaction with objects, resulting in enhanced efficiency, reduced errors, and successful execution of pick-and-place operations. By optimizing these aspects, robots can achieve the needed accuracy ensuring successful execution of pick-and-place operations.

7. Discussions

This chapter focuses on the solution and framework devised within this thesis, and it is divided into three subsections. The first subsection, labelled 7.1, examines the research and development methodology employed for the project. In subsection 7.2, the strategies employed to address the project's limitations situations that have been addressed during the development of the implementation of this project are discussed. Lastly, subsection 7.3 explores the project's objectives and its alignment with sustainability principles. Together, these subsections provide a comprehensive understanding of the final solution, the decisions taken and the framework developed in this thesis, laying the groundwork for the conclusion of the project.

7.1. Method

The DSRM was determined as the research methodology for this thesis given that its characteristics adapted to those required for a project focused on the design of an application. It has been possible to have a structure that has guided the stages of prior research, development of the implementation and subsequent evaluation in a rigorous manner and allowing the re-evaluation of the state in which the project was at each moment.

The aspects of problem identification, motivation and definition of objectives have made it possible to show the current situation of the industrial sector in terms of automation, operator training and the practicality of having a remote operation system with which to keep the operator away from potentially dangerous tasks. To this we added the investigation of the state of the art of certain concepts of current technology, determining those that were in such a state of development that they could be implemented in this system, such as the RobotStudio software, the selected robot and gripper, the Kinect Azure Sensor with its SDK, communication via TCP/IP protocol and the Open3D object recognition system. The software versions of all the programs used during this project can be found in Appendix B

The implementation, in addition to having been determined by the previous stages, has also provided feedback that has been able to help establish the intended objectives, limitations and assumptions that have been carried out which were not contemplated at the beginning. In turn, the resulting implementation, although starting from a properly delimited design, has also had a constant evaluation of how the various functions of the design finally implemented have led to the re-adaptation of other aspects. Some of these modifications can be the combination of code between C# and Python for reasons of library compatibility and time optimisation, the building of the simulation environment through models of the real components; like gripper, camera, table, the creation of an Add-In for RobotStudio to implement new features to the program; like multiple controller signal change, in live creation

of parts or visualization, the installation of the camera and its relationship with the robot and the established work area, and the use of contrasting colours for easy segmentation and recognition by the algorithm.

After the completion of the development of the implementation, the results of the final application in terms of time and accuracy have been evaluated in chapter 6, determining the degree to which the proposed objectives have been met and, finally, the communication carried out by means of a real-time execution of its operation and results.

7.2. Development and Results

In the development of the project, since a design was initially proposed to address the different objectives, some challenges and difficulties have been faced. The various decisions made can be grouped into three main categories and will be discussed according to the course taken in the implementation described in chapter 5: equipment, visual data analysis and virtual environment development.

When theoretical concepts are put into practice, the choice of equipment can be a relative constraint and a number of adaptations must be made. Among the robots that can be found at ASSAR are an IRB 2600 from ABB, two UR10e from Universal Robots, and two GoFa CRB 15000 from ABB. Since the other robots presented problems such as being larger than the required dimensions or being installed in a station with conveyors or without a worktop, it was decided to use one of the two GoFa after an evaluation of its characteristics. By including a work surface in both, it allowed several students to work with a collaborative robot without much effort.

The choice of the rgbd camera was simpler, as it was the only one provided by the centre. The attachment of the camera to the robot (i.e. eye-in-hand sensorisation) eliminated the need to install a fixed structure to capture both workstations or occlusion problems by the robotic arm. Finally, the choice of the gripper was a challenge involving several elements. As could be observed, those parts with a smaller size presented greater errors when reaching the match during the pose estimation. Bringing the camera closer to a lower height than the one used would have provided a higher number of points per piece, highlighting its unambiguous shape. However, this proximity generated some distortion, which could not be solved with the camera's own parameters or calibration, so it was decided to maximise the size of the blocks, taking the width of the gripper as a limit. Adapting its end with the pads offered a moderate flexibility when approaching the part, allowing a low percentage of error not to stop the execution due to collision. If possible, a gripper with a larger opening width would have been selected or even a vacuum gripper, more suited to tasks where there is only rotation in the z-axis.

Regarding software development, starting with the camera application, the official Microsoft "k4a" library for Python within the Azure Kinect Sensor SDK has been used. By including documentation and basic examples of connecting and extracting data from the camera, it was a great help to understand what information could be accessed and the de-

velopment adapted to the objectives of the project. Its computational cost has meant that after setting up the configuration at the start of the run, it takes captures only when the communication prompts it to do so.

After capturing the images, in order to use both images with the same resolution, the options of transforming the colour image with respect to the depth sensor's point of view or transforming the depth image with respect to the RGB camera's point of view are presented. Although the first option was initially chosen as it was desired to maintain accuracy at the cost of lower colour resolution, the number of pixels to generate a point cloud was insufficient for the pose estimation pipeline. By opting for the second option, although it created distance values for the new pixels from the original ones, causing a slight blurring at the edges of the blocks, upsampling occurred, making it easier to identify each side of the block more unambiguously.

Some aspects to highlight in the circle detection algorithm is the filtering of the image. A thershold has been chosen to remove the colours with the highest clarity (intensity values: 200-255). Although the dots are black (intensity values: 0-50), so are the metallic surface spots which, if they are black, present irregularities in their tone causing isolated regions. In relation to this, too aggressive blurring can curve regions such as the ones mentioned above too much, confusing the detection algorithm with false circles.

In this object recognition and pose estimation algorithm, the most important part is the registration based on feature matching, so the transformations must be adapted to this function. The reduction of points and extraction of keypoints has been done trying to obtain a correct performance in as many cases as possible. The number of points after voxelisation and after a slight keypoint extraction makes the pipeline functional against an exhaustive selection of robust keypoints. By starting with two point clouds from two different captures, plus other aspects such as partially blurred edges, there may be differences that cause keypoints that are not present in both sets. While this may result in a better alignment, these different points increase the average measured error of the alignment, which will lead to a rejection of the obtained transformation. Furthermore, this correspondence does not occur with high repeatability, so it is not guaranteed to work.

The resulting transformation is obtained after having reached a convergence close to 1 or after having performed 1000000 iterations. The time required is so low that it has been decided, for those occasions when a matching error above a threshold is obtained, to run iteratively starting from the previous transformation until the threshold is reached, sacrificing an increase in time to guarantee the matching. As already mentioned, the use of Add-Ins for the development of operations in the environment and the development of an interface has allowed for greater capacity in the virtual environment and allows for a more visual and understandable interaction for the user. However, for the time measurements for the final results, an automated version of the interface has been used.

Some features to highlight are the creation of parts. The environment is mostly made up of CAD models as RobotStudio allows the import/export of models. However, despite having the models of the blocks, it has been decided to use a series of functions for the creation of solid objects from basic geometry parameters (radius, height, width, angle,...), thus obtaining the previously defined blocks. Although the development has proved to be as functional as the import of models and shows a higher degree of complexity, initially it was initially aimed at the parameterisation of these values obtained from the visual analysis, that is, extracting distances and angles from the point clouds. A process that has been applied to place the reference system in the point cloud but entailed a higher complexity and a dependence on the resolution of the visual data.

The use of these 5 colors for each part has also allowed the upper and lower dimensions for the filtering by RGB values of the image to be expanded in such a way that different shades of the colour can be accepted without any ambiguity as to whether they belong to one colour or another. The lighting conditions in ASSAR provided homogeneous electric light throughout the plant, although indirect natural light caused changes in intensity. This solution was chosen over the installation of a fixed light source above the station, as it would include other objects in the vicinity of the robot, and could generate shadows on the parts or even on the robot itself, which could limit the robot to stricter working conditions.

With respect to RAPID, the programming of orthogonal trajectories has facilitated the non-collision with other parts. The movement of parts is performed on an XY plane located at 20cm with respect to the table, where no obstacle is expected to be encountered, to align with the x and y values of the coordinate and then descend to the z value to grab or release the block. Together with a reduction of the speed on approach and the zone radius applied to smooth the trajectory, it is intended to achieve for the route a balance between optimal and safe travel. This environment, being specialised in the management of signals and movement orders between the controller and the robot, does not present computational or temporal efficiency to perform other tasks simultaneously such as the operation of matrices or the creation of parts, making the implementation of Add-Ins a great tool for distributing the computational load between the computer and the controller so that the latter can execute the information already processed from the former.

Finally, the design that has been followed for communication has allowed this to be the only connection between the two development environments. In a normal execution, the absence of communication would keep Visual Studio waiting for a response that will not come, stopping the execution after a time limit. However, by replacing this communication with an assignment of the corresponding variable, the environment would achieve the expected results. This independence in the design of the communication has been a great help in facilitating the development, optimisation and debugging of both programs in parallel.

7.3. Sustainable Development

As discussed in Chapter 1.5, sustainability encompasses the economic, social, and environmental dimensions of a project, ensuring its ability to meet present needs without compromising the well-being of future generations. The implications of the project's outcomes on individuals and society have been examined within the context of each pillar of sustainable development.

From a social sustainability perspective, the integration of the Azure Kinect DK in ABB GOFA CRB 15000 for assembly process, like pick and place, tasks can have positive impacts. By automating repetitive and physically demanding tasks, it reduces the strain on human workers, leading to improved health and safety conditions. It also frees up workers to engage in more complex and intellectually stimulating work, promoting their professional growth and job satisfaction. This technology can also contribute to skill development, as workers can learn and operate alongside advanced robotic systems.

In terms of economic sustainability, computer vision-enabled in robots can enhance productivity and cost-effectiveness. By increasing the efficiency of pick and place operations, companies can reduce production time and minimize errors, leading to improved overall productivity. This can result in cost savings and improved competitiveness. Furthermore, the integration of computer vision technology can drive innovation and create new business opportunities, as companies develop expertise in this field and offer advanced automation solutions to clients.

Regarding environmental sustainability, the use of computer vision in robot environments can lead to resource conservation and waste reduction. By accurately recognizing and handling objects, robots can minimize material waste and optimize resource utilization. This reduces the consumption of raw materials and contributes to more sustainable production processes. Additionally, computer vision-enabled robots can enhance energy efficiency by performing tasks with precision and speed, minimizing energy consumption.

On the other hand, integrating computer vision into robotic assembly processes brings complexity in algorithm development and system calibration, requiring specialized expertise. This complexity arises due to the need for developing and fine-tuning computer vision algorithms, integrating sensors, and calibrating the system for optimal performance. However, high initial costs associated with implementing computer vision systems pose financial challenges for organizations. The expenses involved in procuring cameras, sensors, and processing hardware can be significant, making it difficult for smaller businesses or startups with limited financial resources to adopt such technologies. Furthermore, computer vision systems and robotic components require regular maintenance, calibration, and occasional repairs. Downtime during maintenance or system failures can disrupt assembly processes, impacting productivity and potentially increasing costs.

Overall, the integration of computer vision in robot environments for the assembly process has the potential to positively impact social, economic, and environmental sustainability. It improves working conditions, enhances productivity and cost-effectiveness, and contributes to resource conservation and energy efficiency. Otherwise, there are some drawbacks. By acknowledging and addressing them organizations can better navigate the integration of computer vision in robotic assembly processes, ensuring a balanced and sustainable approach to technology implementation.

8. Conclusions

As a conclusion of this thesis, the results will be considered in relation to the proposed objectives to determine the achievements and their implications as a proposed solution to the problems posed.

What has been achieved with this project is the combination of an object recognition system communicating with a virtual environment with the trajectory capability of an articulated robot, to mimic the state of the parts of an assembly station manipulated by a human in a real environment. Regarding the creation of trajectories, in addition to being able to operate the robot in the virtual environment in the desired way, it can extrapolate these orders to the real robot, allowing a more complete human-robot interaction and extending the possible applications of the designed solution.

In this way, through the exclusive use of the virtual environment it can have applications in training and adaptation of new operators to the assembly station, incorporation of a real robot greatly extends the possible applications such as robot assembly in hazardous environments teleoperated by a human in a safe space or, in non-automatable industry processes that require human supervision. It, therefore, has sufficient functions to serve as a model on which to develop a more advanced real application by extending its framework of delimitations.

In terms of the objectives achieved in the research stage, in addition to the skills obtained on the use of the diverse software and equipment, it has been possible to develop, thanks to the DSRM, a rigorous study of the current situation in the fields of collaborative robots, automatic modelling of parts, computer vision, use and processing of data from RGBD cameras and compatible software. This information has not only provided a more solid basis of expertise for the development of a feasible design but also, like the implementation itself, serves as documentation that compiles these concepts to lay the foundations for future studies with the aim of further optimising processes and tasks in the industry in a flexible manner by combining various fields of study.

Throughout the project, valuable insights have been gained in several technological and programming areas. A key lesson learned involved mastering communication between different devices through socket communication on a local network. This knowledge expanded the understanding of networked systems and their practical applications. Additionally, the project's focus on creating a RobotStudio add-in enhanced familiarity with C#, the programming language used. Exploring computer vision also proved instructive, revealing how cameras perceive and process their surroundings. This understanding facilitated the development of camera programming techniques to apply, such as automation.

To sum up, this thesis has complied all the necessary data and background to develop an implementation which has successfully combined an object recognition system with a virtual environment and the trajectory capabilities of an articulated robot, thus imitating the state of parts in an assembly station and allows for human-robot interaction.

9. Future Work

Considering the resulting project, the limitations initially established and the areas of improvement that can be found in part recognition, part grasping and processing time for recognition and image processing, future work on the project would focus on improving overall performance and addressing these challenges. Some potential areas of future work include:

- Advanced object recognition algorithms: Research and develop more sophisticated algorithms that can effectively identify and classify pieces in different configurations, considering variations in shape, colour, and texture. This could involve incorporating advanced techniques such as deep learning, convolutional neural networks, and feature extraction methods to enhance the system's ability to identify and classify pieces in various configurations. By training the system on a larger dataset and leveraging deep learning approaches, the accuracy and speed of recognition can be improved.
- Real-time image processing: Optimize the image processing pipeline to reduce the processing time and improve the system's responsiveness, enabling faster recognition and decision-making during robot operations as well as detecting possible errors during the execution such as incorrect positioning of the part due to dropping.
- Robot-Camera calibration. Involving the robot in this process can accurately obtain the intrinsic camera parameters and relate robot movements to the perceived information from the RGBD camera instead of establishing a work object manually (3-point method). Calibration between the end-effector and the camera by automatic movements of the robot would allow the establishment of a transformation matrix between the robot end-effector and the camera by collecting correspondences between the poses of the robot hand and the features observed in the camera images from the chessboard.
- Collision avoidance: Detecting potential collisions prior to movement can minimise the risk of accidents or damage to equipment and inform the operator of the need to reroute. RobotStudio has available features to identify potential collisions based on the robot's geometry, joint boundaries and object positions by simulating and analysing the robot's path and surrounding objects. In addition, some visual information can be displayed to users, enabling them to adjust the trajectory. Its adapted implementation to the project can detect whether or not it can perform the movement in case of a possible collision of the gripper with other parts when trying to grip one or if there is already a part in the target position.

- Advanced grasping: Extend the block reorientation capability to the X and Y axes. Enabling the robot to lay the block down or stand it upright would increase the flexibility and versatility of pick and place tasks, allowing for more complex assembly and handling tasks. This would involve the selection of a gripper adapted to grasp the blocks from multiple orientations like a vacuum gripper, the programming of multiple work objects on each piece to select the most appropriate one for the situation and a full 3D modelling of the block to determine from the camera its orientation in all 3 axes.

References

ABB. (2004). *RobotWare - Industrial Robot Controller Software.* Retrieved 09/03/2023, from https://library.e.abb.com/public/.

ABB. (2014). *RobotWare - The world's leading robot controller software.* Retrieved 10/03/2023, from fromhttps://library.e.abb.com/.

ABB. (2021). *Collaborative Robots - GoFa CRB 15000.* Retrieved 13/03/2023, from https://search.abb.com/library.

ABB Developer Center. (n.d.). *RobotStudio Overview.* Retrieved 17/03/2023, https://developercenter.robotstudio.com/api/robotstudio/articles/Concepts/Appearances/Appearances-Topic.html.

Agency, E. E. (2019). *Sustainability transitions : policy and practice.* Publications Office.

Asada, H., & Slotine, J.-J. (1991). *Robot analysis and control.* John Wiley & Sons.

Ben-Ari, M., Mondada, F., Ben-Ari, M., & Mondada, F. (2018). Robots and their applications. *Elements of robotics*, 1–20.

Blank, A. G. (2006). *Tcp/ip foundations.* John Wiley & Sons.

Bohuslava, J., Martin, J., & Igor, H. (2017). Tcp/ip protocol utilisation in process of dynamic control of robotic cell according industry 4.0 concept. In *2017 ieee 15th international symposium on applied machine intelligence and informatics (sami)* (pp. 000217–000222).

Brad Stephenson. (2019). *Microsoft's New Azure Kinect DK Camera.* Retrieved 14/03/2023, https://www.onmsft.com/news/microsofts-new-azure-kinect-dk-camera-is-now-available-to-buy-online/.

Cedmonds. (2022). *Find then open the azure kinect device. learn.microsoft.com.* Retrieved 13/03/2023, from https://learn.microsoft.com/en-us/azure/kinect-dk/find-then-open-device?source=recommendations.

Code, V. S. (2019). *Visual Studio Code.* Retrieved from http://mentorthis.s3.amazonaws.com/upload/files/2022/06/55yIf7PjAppzmBhYxk1L_06_8b52883468172443a3960cb347a3a4f5_file.pdf.

Colleges, M. C. (2021). *What is sustainability?* Retrieved from 12/03/2023 https://www.maricopa.edu/about/sustainability.

Dai, Y., Xiang, C., Qu, W., & Zhang, Q. (2022). A review of end-effector research based on compliance control. *Machines*, *10*(2), 100.

Darwish, W., Li, W., Tang, S., Wu, B., & Chen, W. (2019). A robust calibration method for consumer grade rgb-d sensors for precise indoor reconstruction. *IEEE Access*, *7*, 8824–8833.

El Makrini, I., Merckaert, K., Lefeber, D., & Vanderborght, B. (2017). Design of a collaborative architecture for human-robot assembly tasks. In *2017 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 1624–1629).

Forsyth, D., & Ponce, J. (2012). *Computer vision: A modern approach. always learning.* Pearson.

Fu, K., Peng, J., He, Q., & Zhang, H. (2021). Single image 3d object reconstruction based on deep learning: A review. *Multimedia Tools and Applications*, *80*, 463–498.

Gao, M., Ruan, N., Shi, J., & Zhou, W. (2022). Deep neural network for 3d shape classification based on mesh feature. *Sensors*, *22*(18), 7040.

Geerts, G. L. (2011). A design science research methodology and its application to accounting information systems research. *International journal of accounting Information Systems*, *12*(2), 142–151.

Goertz, R. C. (1949). *Master-slave manipulator* (Vol. 2635). Argonne National Laboratory.

Guizzo, E. (2019). By leaps and bounds: An exclusive look at how boston dynamics is redefining robot agility. *IEEE Spectrum*, *56*(12), 34–39.

Han, M., Zhang, Z., Jiao, Z., Xie, X., Zhu, Y., Zhu, S.-C., & Liu, H. (2022). Scene reconstruction with functional objects for robot autonomy. *International Journal of Computer Vision*, *130*(12), 2940–2961.

He, R., Rojas, J., & Guan, Y. (2017). A 3d object detection and pose estimation pipeline using rgb-d images. In *2017 ieee international conference on robotics and biomimetics (robio)* (pp. 1527–1532).

Holubek, R., Delgado Sobrino, D. R., Košťál, P., & Ružarovskỳ, R. (2014). Offline programming of an abb robot using imported cad models in the robotstudio software environment. In *Applied mechanics and materials* (Vol. 693, pp. 62–67).

Iddan, G. J., & Yahav, G. (2001). Three-dimensional imaging in the studio and elsewhere. In *Three-dimensional image capture and applications iv* (Vol. 4298, pp. 48–55).

International Federation of Robotics. (2021). *International Federation of Robotics.* Retrieved 13/03/2023, from `https://ifr.org/robot-history`.

Klette, R., Koschan, A., & Schluns, K. (1998). Three-dimensional data from images. *Springer-Verlag Singapore Pte. Ltd., Singapore.*

Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital twin in manufacturing: A categorical literature review and classification. *Ifac-PapersOnline*, *51*(11), 1016–1022.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Lefranc, G., López, I., Osorio-Comparán, R., & Peña, M. (2022). Cobots in automation and at home. In *2022 ieee international conference on automation/xxv congress of the chilean association of automatic control (ica-acca)* (pp. 1–6).

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*, 91–110.

Malone, B. (2011). George devol: a life devoted to invention, and robots. *IEEE Spectrum Magazine. https://spectrum. ieee. org/automaton/robotics/industrial-robots/george-devol-a-life-de voted-to-invention-and-robots.*

Microsoft. (2020). *Azure Kinect DK.* Retrieved 05/03/2023, from `https://azure.microsoft.com/en-us/products/kinect-dk/`.

Nari Shin. (2020). *3D hand-eye calibration for logistics automation.* Retrieved 17/06/2023, `https://blog.zivid.com/3d-hand-eye-calibration-for-logistics-automation`.

Nguyen, A., & Le, B. (2013). 3d point cloud segmentation: A survey. In *2013 6th ieee conference on robotics, automation and mechatronics (ram)* (pp. 225–230).

Nikula, R.-P., Paavola, M., Ruusunen, M., & Keski-Rahkonen, J. (2020). Towards online adaptation of digital twins. *Open Engineering*, *10*(1), 776–783.

O'Regan, G., & O'Regan, G. (2013). George devol. *Giants of Computing: A Compendium of Select, Pivotal Pioneers*, 99–101.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, *24*(3), 45–77.

Pérez, L., Rodríguez, Í., Rodríguez, N., Usamentiaga, R., & García, D. F. (2016). Robot guidance using machine vision techniques in industrial environments: A comparative review. *Sensors*, *16*(3), 335.

Räsänen, I. (2020). *Virtual reality in industrial robot monitoring* (B.S. thesis).

Robotics, A. (2007a). *Gofa crb 15000.* Retrieved 20/04/2023 from `https://www.inf.u-szeged.hu/~groszt/teach/Robotika/Segedanyag/3HAC032104-en.pdf`.

Robotics, A. (2007b). *Operating manual robotstudio. västerås, sweden.* Retrieved from `https://www.inf.u-szeged.hu/~groszt/teach/Robotika/Segedanyag/3HAC032104-en.pdf`.

Rokhim, I., Ramadhan, N. J., & Rusdiana, T. (2021). Image processing based ur5e manipulator robot control in pick and place application for random position and orientation of object. , 124-130. doi: 10.1109/ISMEE54273.2021.9774170

Rusu, R. B., & Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *2011 ieee international conference on robotics and automation* (pp. 1–4).

Schaller, M. (2016). The empirical benefits of conceptual rigor: Systematic articulation of conceptual hypotheses can reduce the risk of non-replicable results (and facilitate novel discoveries too). *Journal of Experimental Social Psychology*, *66*, 107–115.

SCHUNK. (2018). *Co-act egp-c 40-n-n-gofa: Collaborating gripper for small components.* Retrieved from `https://schunk.com/se/en/gripping-systems/parallel-gripper/co-act-egp-c/co-act-egp-c-40-n-n-gofa/p/000000000001468548`.

Shahria, M. T., Sunny, M. S. H., Zarif, M. I. I., Ghommam, J., Ahamed, S. I., & Rahman, M. H. (2022). A comprehensive review of vision-based robotic applications: Current state, components, approaches, barriers, and potential solutions. *Robotics*, *11*(6), 139.

Spong, M. W., & Mohammadi, A. (2022). Robot modeling and control. *IEEE CONTROL SYSTEMS*, *1066*(033X/22).

Telang, S., Rout, B., & Pande, A. (2018). Implementation of cascade classifier technique for gesture control of an abb industrial manipulator. In *2018 2nd international conference on inventive systems and control (icisc)* (pp. 748–753).

UNDP. (2015). *Sustainable development goals. united nations development programme.* Retrieved from 02/03/2023 `https://www.undp.org/sustainable-development-goals`.

Valle, M. G. (2016, September 10). *El lenguaje de programación c.* Retrieved from `https://www.academia.edu/28385406/El_lenguaje_de_programaci%C3%B3n_C`.

Venable, J. R., Pries-Heje, J., & Baskerville, R. L. (2017). Choosing a design science research methodology.

Wang, L., & Mohammed, A. (2012). Vision-assisted and 3d model-based remote assembly. In *International conference on innovative design and manufacturing.*

Wiley, V., & Lucas, T. (2018). Computer vision and image processing: a paper review. *International Journal of Artificial Intelligence Research*, *2*(1), 29–36.

Wu, A., Xiao, H., & Zeng, F. (2019). A camera calibration method based on opencv. In *Proceedings of the 4th international conference on intelligent information processing* (pp. 320–324).

Zahavi, A., Afrange, F. A., Haeri, S. N., Ajeevan, U., & Liyanage, D. C. (2018). Abb yumi© high-speed pick and place game in action. In *29th daaam int. symp. on intelligent manufacturing & automation* (pp. 1216–1221).

Zhou, Q.-Y., Park, J., & Koltun, V. (2018). Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*.

A. Work Breakdown and Time Plan

Activity	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17	W18	W19	W20	W21	W22
Literature Review																			
Basic Learning																			
Report Writing																			
Camera Calibration-testing																			
Build Simulation Environment																			
Mid Term Presentation																			
Software Synchronisation																			
Optional Objectives																			
Verification and Validation																			
Final Oral Presentation																			

Table A.1: Original Time Plan

Activity	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17	W18	W19	W20	W21	W22
Literature Review																			
Basic Learning																			
Report Writing																			
Camera Calibration-testing																			
Build Simulation Environment																			
Mid Term Presentation																			
Software Synchronisation																			
Optional Objectives																			
Verification and Validation																			
Final Oral Presentation																			

Table A.2: Final Time Plan

B. Software Versions

Group	Name	Version
Robot Programming	Robotstudio	2023.3.1
	RobotWare	7.7.1
	Wizard	1.4.1
Add-In	VisualStudio 2022	17.5.5
	RobotStudio SDK	2022.3.1
	PC SDK	2022.3.1
	.NET Framework	4.8
Camera Connection & Image data analysis	Visual Studio 2022	17.5.5
	Python	3.9.0
	Pip	23.0.1
	Open3D Library	0.17.0
	OpenCV Library	4.7.0

Table B.1: Software version used

C. Specifications

C.1. GOFA CRB 15000

Robot version	Reach (mm)	Payload (kg)	Armload (kg)
CRB 15000	950	5	No armloads
Number of axes	6		
Protection	IP54		
Mounting	Any angle, including table mounting, wall mounting, and ceiling mounting		
Controller	OmniCore C30		
Customer power supply	24V/2A supply		
Customer signals	4 signals (for IO, Fieldbus, or Ethernet)		
Tool flange	Standard ISO 9409-1-50		
Functional safety	SafeMove Collaborative included All safety functions certified to Category 3, PL d		

Table C.1: – GOFA CRB 15000 Specifications (ABB, 2021)

Axis movement	Working range	Axis max. speed
Axis 1 rotation	−180° to 180°	125°/s
Axis 2 arm	−180° to 180°	125°/s
Axis 3 arm	−225° to 85°	140°/s
Axis 4 wrist	−180° to 180°	200°/s
Axis 5 bend	−180° to 180°	200°/s
Axis 6 turn	−270° to 270°	200°/s

Table C.2: – GOFA CRB 15000 Movement (ABB, 2021)

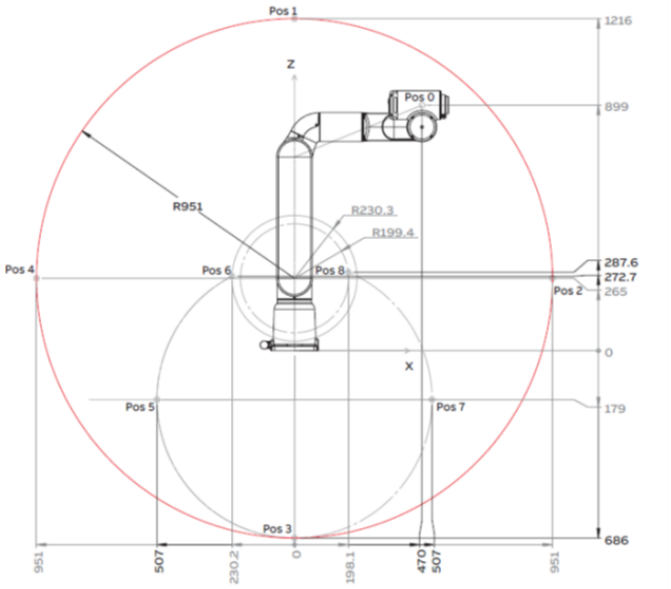


Figure C.1: Working Range Side View (ABB, 2021)

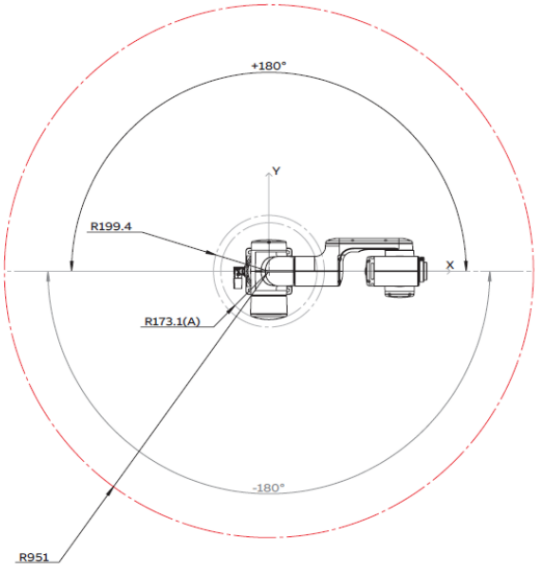


Figure C.2: Working Range Top View (ABB, 2021)

C.2. Azure Kinect DK

In the following tables and figures, these abbreviated terms are used:

- NFOV: Narrow field-of-view depth mode
- WFOV: Wide field-of-view depth mode
- FOV: Field-of-view
- FPS: Frames-per-second
- FoI: Field of Interest
- IR: Infra-Red

Mode	Resolution	FoI	FPS	Operating range*	Exposure time
NFOV unbinned	640x576	75°x65°	0, 5, 15, 30	0.5 - 3.86 m	12.8 ms
NFOV 2x2 binned	320x288	75°x65°	0, 5, 15, 30	0.5 - 5.46 m	12.8 ms
WFOV 2x2 binned	512x512	120°x120°	0, 5, 15, 30	0.25 - 2.88 m	12.8 ms
WFOV unbinned	1024x1024	120°x120°	0, 5, 15	0.25 - 2.21 m	20.3 ms
Passive IR	1024x1024	N/A	0, 5, 15, 30	N/A	1.6 ms

Figure C.3: Depth camera supported operating mode (Microsoft, 2020)

Operating range*: 15% to 95% reflectivity at 850nm, 2.2 W/cm /nm, random error std. dev. 17 mm, typical systematic error $<$ 11 mm + 0.1% of distance without multi-path interference. Depth may be provided outside of the operating range indicated above. It depends on the object's reflectivity.

Depth Mode	IR Pulses	Pulse Width	Idle Periods	Idle Time	Exposure Time
NFOV Unbinned NFOV 2xx Binned WFOV 2x2 Binned	9	125 us	8	1450 us	12.8 ms
WFOV Unbinned	9	125 us	8	2390 us	20.3 ms

Figure C.4: Depth sensor raw timing (Microsoft, 2020)

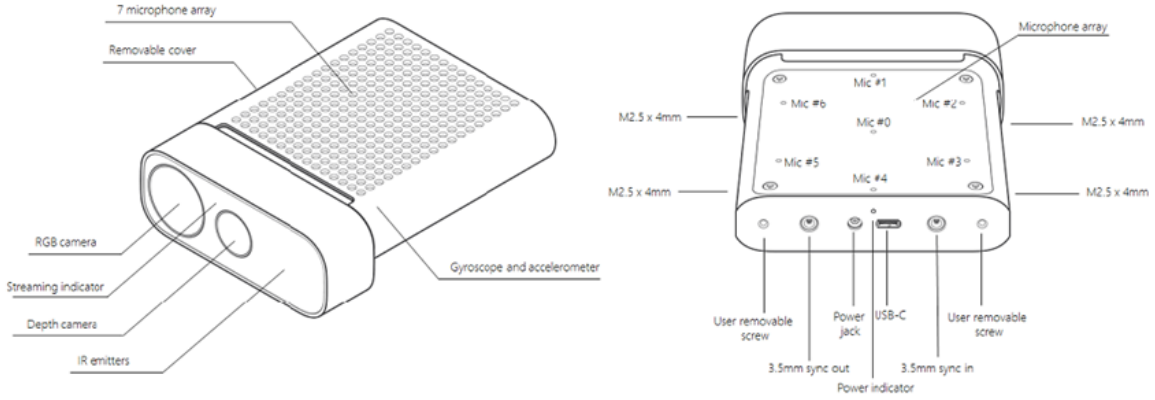


Figure C.5: Azure Kinect hardware integrations (Microsoft, 2020)

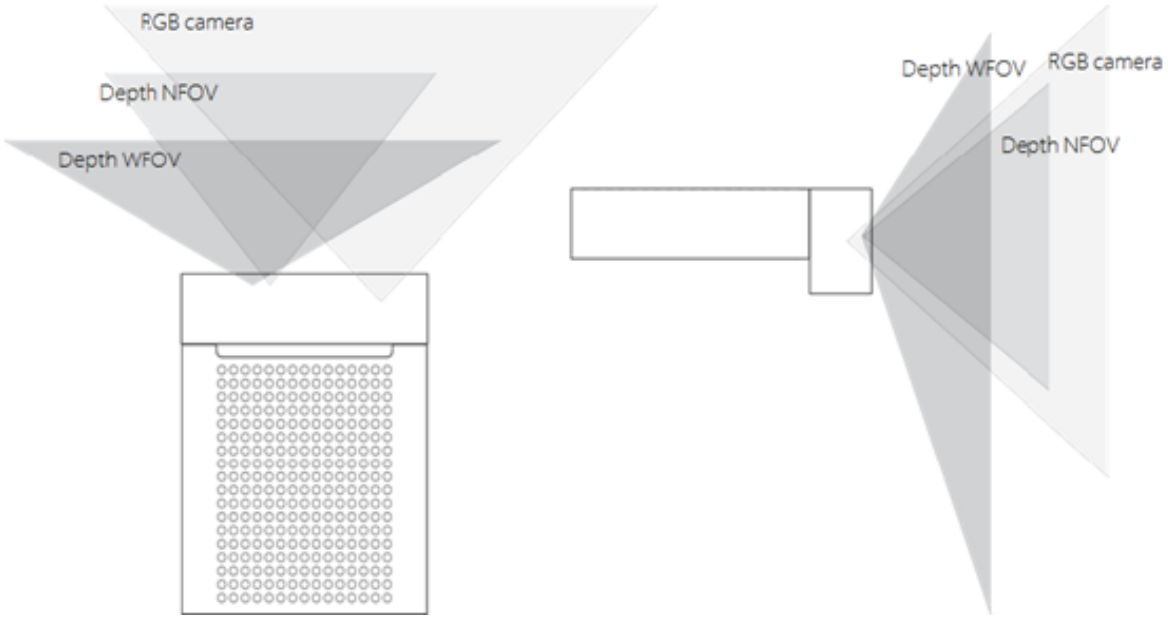


Figure C.6: Camera fields of view (Microsoft, 2020)

C.3. Co-act EGP-C 40

Description		Co-act EGP-C 40-N-N-GoFa
ID		1468548
General operating data		
Compatible robot		ABB GoFa (CRB 15000)
Robot flange		Standard flange
Integrated sensors		yes, inductive in two directions
Dimensions X, Y, Z	[mm]	93.8, 90.2, 123
Mechanical operating data		
Stroke per jaw	[mm]	6
Min./max. gripping force	[N]	35/140
Min./max. force per jaw	[N]	17.5/70
Recommended workpiece weight	[kg]	0.7
Max. permissible finger length	[mm]	50
Max. permissible weight per finger	[kg]	0.08
Repeat accuracy	[mm]	0.02
Closing/opening time	[s]	0.2/0.2
Weight	[kg]	0.62
Min./max. ambient temperature	[°C]	5/55
IP protection class		30
Cable connector/cable end		2 x M8
Cable length L1	[mm]	70
Cable length L2	[mm]	175
Electrical operating data		
Nominal voltage	[V DC]	24
Nominal current	[A]	0.2
Max. current	[A]	2
Controller electronics		integrated
Communication interface		digital I/O
Number of digital I/O		4/2

Table C.3: Technical data Co-act EGP-C for ABB