Integration and development of a collaborative robot assisting in assembly of dummy phones

Master Thesis

Mads Riis Thomsen

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg

AALBORG UNIVERSITY
STUDENT REPORT

Robotics
Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg
http://www.aau.dk

Title:
Integration and development of a collaborative robot assisting in assembly of dummy phones

Project Period:
01/09/2022 - 05/01/2023

Participants:
Mads Riis Thomsen

Supervisors:
Dimitris Chrysystomou
Chen Li

Page Count: 37

Date of Completion:
4th January 2023

Abstract:

This report will showcase a collaborative robot with the capabilities of picking up items when instructed with a full sentence, and the capabilities of assisting the user with assembling dummy phones. This project is a part integration of a previous project, which the author were part of development for, of a simulated robot system capable of using Named-Entity Recognition (NER) to extract task and object information from a sentence and then executing said task on said object. The prototype developed in this project is able to pick up object. Additionally, the system is also capable of using colour detection techniques to determine the progress of an assembly, and is able to determine the next part need for the assembly in order to assist the user.

Preface

This project report is made by Mads Riis Thomsen, in the 10th semester of Robotics, under the Department of Electronic Systems at Aalborg University. The project was made over the course of four months starting at September 1st and was supervised by Chen Li and Dimitris Chrysostomou.

This report is targeted towards people with a technical background and thus explanations of certain technical terms, assumed to be of basic knowledge, will not be given.

The author wants to thank Aalborg University for providing guidance, equipment and knowledge and thus making this project a possibility.

[illegible]

Mads Riis Thomsen

Reading guide

Citing before a period indicates that the citation applies to the previous sentence whereas a citing after a period indicates that the citation applies to the whole paragraph.
References are ordered by order of being mentioned, with numeric citation style. Bibliography entries contain information in the following order; author(s), title, miscellaneous info, URL and URL date in the end (if available online). Dates are in standard date format of year-month-day.

This report is focused around Aalborg University in Denmark and thus inside the EU. Therefore, there might be legislative aspects outside the EU, which are not considered.

Abbreviations list

Abbreviation	**Definition**
HMI	Human-Machine Interface
NLP	Natural Language Processing
NER	Named Entity Recognition
CNN	Convolutional Neural network

Contents

Chapter 1

Introduction and Background

1.1 Human robot collaboration

Industry 4.0 is the current manufacturing trend which intends to implement more automation, machine-machine-communication, internet of things and more. Among these, the concept of cyber-physical systems.[1]

With industry 4.0 taking off, it has brought an exceptional amount of collaboration between humans and robots. Before the 4th industrial revolution, the robots and humans existed separately from each other, with robotic manipulators being kept in cages as to not harm human workers. Later, the world saw the introduction of new robot manipulators, who had been designed with touch sensors, slower speeds and more rounded design. These were named cobots. This bridged the world between the robots and the humans, and they could now collaborate together to perform tasks.[2]
The human-robot interaction (HRI) has been a popular research topic and has seen a lot of development through out the years. One of these developments are the addition and integration of cameras to give the robot visual perception. With the great development of Convolution Neural Networks (CNN) over the past decade, robotic visual perception has been seen a lot of improvements.[3]

With this it could be possible for a robot to observe a process or task done by a human and learn from it. This could result in quick and easy pivoting when there is a change in production, or it could be used to assist the human worker in the assembly.[3]

1.1.1 Previous work

Previously, the author had worked on a project (P8). This project was developed in a simulation environment, thus it has not been integrated into a real life setup. This section will briefly describe the overall project, and will be describe even further in Chapter 2.

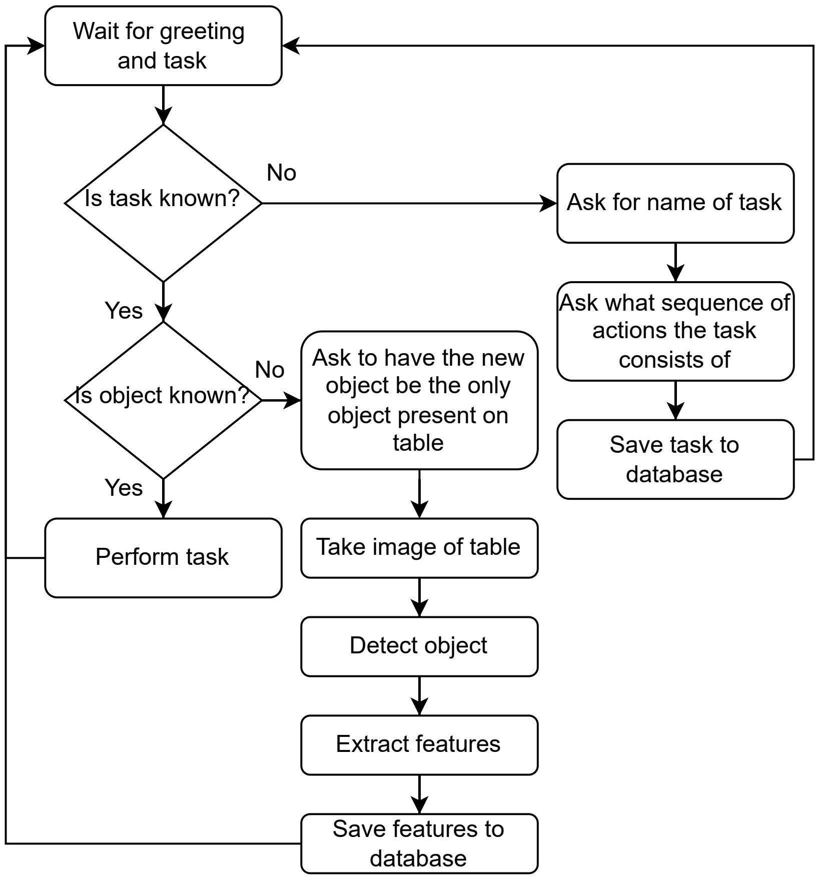

Wait for greeting and task
Is task known?
No
Ask for name of task
Ask what sequence of actions the task consists of
Save task to database
Yes
Is object known?
No
Ask to have the new object be the only object present on table
Yes
Perform task
Take image of table
Detect object
Extract features
Save features to database

Figure 1.1: A rough flowchart of the logic of the previous P8 project

P8 project description

The P8 project, formally titled "A Deep-Learning Based Collaborative Robot Cell With Natural Language Instruction and Visual Semantic Reasoning", was developed by the author and other project members in spring 2021.
The project aimed to create a robot system that was able to detect objects on a worktable, learn new objects placed on the worktable, recognise speech and process the speech (NLP). This was developed in a simulation environment, using a UR5 as a robot platform. Due to lockdowns at the time, the project was never integrated into a real life platform.

The goal of this project (Master project) is to integrate the previous project into a real life system. This, however, has its own complications. Especially the previous project ended up consisting of multiple different modules including: object detection, feature extraction, grounding, speech-to-text, text-to-speech, entity extraction, command building and robot controller. This is a lot of different modules which all contain their own difficulties for integration. As such, this thesis will examine the project more deeply in Chapter 2 in order to decide on what modules to focus on integrating during this project. A diagram of all the modules and how they interact with each other can be seen in Appendix A.

The system would start off by waiting for a verbal user greeting, task and object. A task consists of a sequence of actions, and the robot would know simple actions coded into the robot such as: move to position, open gripper, close gripper, etc. If the task was not known, the user could teach the robot the new task by giving it the name of the task and the sequence of actions the task consists of. For object, it had a database of all the objects it already knew, which consisted of the objects name and its features. If the object is not known the user is asked to clear the table and place the unknown object on the table. The robot would then take an image to the table and detect the object lying on the table. It would then extract the feature of the object and save the objects name and features to the database. As flowchart of this can be seen in Figure 1.1.

1.1.2 Goal and initial problem formulation

The goal of this project is to integrate the previous P8 project onto a real life robotic system, as well as developing a module for the system that is able pick up relevant parts needed for assembly depending on the state of the assembly. An example of this would be if the current state of the assembly is placing the PCB into the assembly, the system would recognise this and pick up the PCB for either handing it over to an operator or even placing it into assembly itself.

Based on this goal, an initial problem formulation is formed to be:

- How can a robotic picking system be integrated onto a physical robot and be used to assist in assembling a product?

Chapter 2

Previous project

This chapter will dive into the previous project and the different technologies it used, in order to figure out which ones to focus on when integrating.

2.1 Vision

This section is about the visual object recognition. First, the system will use background subtraction in order to get blobs for each object present on the worktable. The system will then crop out each blob in the image, which will then go through feature extraction. The feature extraction is done in a neural network method. The model used is the MobileNetV2. The training had 2 stages to it, first the training was to be able to classify 5 different object present on the work table, and secondly it was trained using triplet learning witht he classification layer removed. The feature space was reduced to be of size 3 as there would only be 5 object present on the table, meaning if the size was greater it would likely overfit and be sensitive to noise.

When the system is looking for an object, crop out an object and extract its features. The features extracted are then compared to the features present in the "Grounding Database", which contains the currently known objects, their features and their name. It will then classify the object to the object it closely resembles in the database.
The system can also learn new objects. This assumes that the foreign object is the only object present on the table, meaning it is the only object the system will detect. It will then extract the features of this object and add it to the database along with the specified name.

2.2 Grounding

The grounding module is responsible for finding the commonality between a spoken word and its meaning. An example would be if the user asks the robot to find the blue cover, then

the module would be responsible for figuring out what object is the "blue cover".
It does this by using the features from the feature extractor, and then check the database as described in Section 2.1. However, there is also the case of where the requested object appears multiple times on the worktable. In such cases, the request either wants any of the objects or a specific one. If the user wants any of them the robot will just grab the first one it detects. However, if the user wants a specific item it is requested with a "spatial relation" word. This is words such as "to the left of". In such cases, the grounding module has logic that looks for the specific item requested. This is done by first getting every object which matches the classification, then it will analyse each of those and return the one that matches the spatial relation the best. An example would be: The user requests "Find the blue cover above the black cover", the robot then searches for all the covers and classifies them. If there are more than one blue cover, it will compare the position of those to see which ones matches the spatial relation. In this case, it would also have to get the position of all the black covers to figure out the correct blue cover to find.

Along with object grounding, there is also task grounding. This is to connect the spoken task to a sequence of actions the robot has to take. The database also has entries for tasks, which consists of the name of the task and its sequence of actions. If a task is not known, the user can teach the robot the new task by providing with the sequence of actions the task consists of. The learned task is then saved to the database, and can then also be used to teach new tasks. An example would be, if you first teach the robot the task of picking up an item, then you would be able to teach the robot a task that has picking up items as one of the tasks acions.

2.3 NER

Text-to-speech (TTS) and speech-to-text (STT) capabilities are present on the robot, in order to give and receive verbal instructions. These modules are both using Microsoft Azure, and they convert audio to a string of text or vice-versa. The spoken input is then processed in the NER module, which analyses the string of text to extract the intent behind the words spoken. This is done in a Deep Learning model.

This model is trained using the DistilBERT model, with a dataset consisting of 333 sentences. All of these sentences have the words labelled with associated entities, such as "object", "location", "take" etc. This is so the model can learn what words and context refer to what intent. The performance on the trained model can be seen in Table 2.1.

Precision	Recall	F1	Accuracy
0.9799	0.9669	0.9733	0.9746

Table 2.1: The performance of the trained NER on test dataset.

2.4 Robot interface

Once the coordinates for pick up are found, they are sent to the robot controller which tells the robot to do the correct set of actions
The interface is done through ROS MoveIt! This framework makes it easy to interface with the UR5 and would have the added benefit of simplifying a transitions another robot.

Chapter 3

Equipment

3.1 Franka robot

The Franka robot arm is a 7 DOF collaborative robot. Besides the additional degree of freedom, the robot is similar to the UR5 robot used in the previous project. This makes it so the project has an easier time being ported over to the Franka arm.
The robot arm can be seen in Figure 3.1

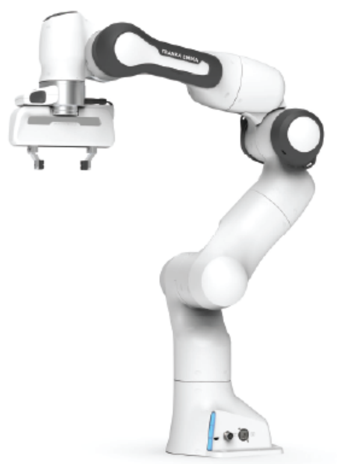


Figure 3.1: The Franka Panda robot arm. [4]

3.2 Intel RealSense Camera

The RealSense D415 is an active IR stereo depth camera by Intel. The D415 is well suited for high accuracy depth applications, such as the bin-picking problem. It provides the highest depth quality per degree when compared the RealSense D435.

3.3 Dummy phone parts

The objects the robot will be manipulating are the
AAU Smart Lab dummy phone parts. These are 4 parts that can be assembled together to resemble a mobile phone. For this project, a white front cover and a blue back cover are used, and the fuse parts of the dummy phone are pre-installed on the PCB. An image of a phone with black covers can be seen in Figure 3.2.

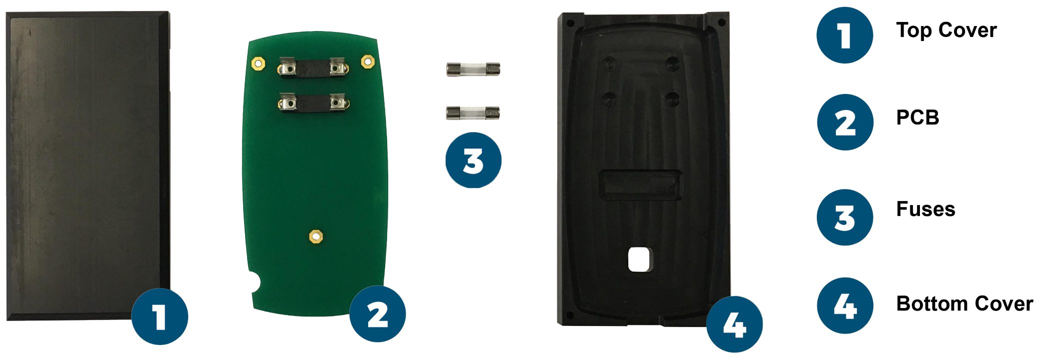


Figure 3.2: The parts for the dummy phone with black coloured covers. For this project 1 is a white color, 4 is a blue color and the two fuses in 3 are pre-installed on the PCB in 2 [5]

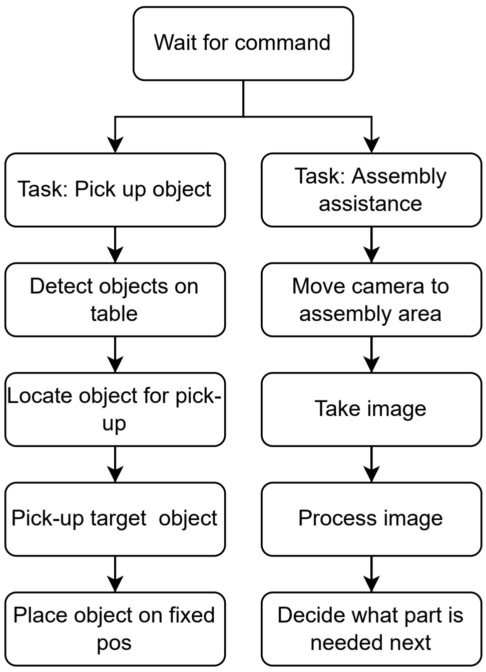
Chapter 4

Requirements and goals

4.1 Ideal vision for the final product

The ideal end vision of the final product of this project will be integrating enough of the previous P8 project, such that it is capable of executing a pick up task. As well as the addition of a visual reasoning module. The new module should be able to look at an partial assembly of the dummy phone, and then figure out what part is needed next in the assembly process. An example of this would be:

- Assembly process is to place the PCB part into the back casing part
- The robot sees a back casing part in the assembly fixture
- The robot recognises that the next part in the process is the PCB part
- The robot will locate and pick up the PCB part



Wait for command
Task: Pick up object
Task: Assembly assistance
Detect objects on table
Move camera to assembly area
Locate object for pick-up
Take image
Pick-up target object
Process image
Place object on fixed pos
Decide what part is needed next

Figure 4.1: General flowchart of the vision for the final product

Ideally the system would also be able to perform the actions the P8 system implemented, such as verbal communication, object and task learning, and deal with stochastically placed parts.

4.2 Problem approach

With the previous section in mind, it would is useful to come up with an approach to the implementation. The implementation can be split up into two phases: An integration phase and a development phase. The integration phase focuses on the integration of the P8 project onto the Franka robot and has to find ways to deal with a new robot, environment and workstation. The development phase is about the development and implementation of the visual reasoning idea explained in Section 4.1.

4.3 Delimitation

Due to limited time and resources the ideal vision for the final product will not be feasible to develop. As such, it has to be chosen what parts of the development to focus on more and what parts to focus less on, with the knowledge that the parts that are focused less on are not ideal solution.

For the integration phase, most of the focus will be on integrating enough of the P8 project to be able to execute a pick up task. This will require simple interfacing with the different modules of the P8 project, meaning a lot of the capabilities will not be integrated fully or at all. It has been decided that the part will also be placed in static locations, in order to simplify the robot control aspect of the system. The object and task learning capabilities will not be touched, as these features are very different than the general tasks the robot is capable of doing. Finally, the verbal interfacing using speech-to-text and text-to-speech will also be removed from the product, in order to avoid errors of a noisy environment.

The development phase will be limited in the sense that it will be a first idea solution. This means it likely will not be the best solution to the implementation the assembly assistance in this context. However, after implementing it and seeing its performance it leaves room for improvement and optimisation. Additionally, the assembly assistance module is developed over a period of time where access to the lab was restricted, thus the prototype of this module would not be fine tuned for the robotic system but still be implemented in the project.

4.4 Requirements

4.4.1 User Requirements

User requirements are the first requirements that are drawn and identified from the research performed in chapter 2. Where possible, this section will expand on the rationale for each requirement. User requirements are set from the perspective of a non-technical user, designed to represent what they are looking for in a solution. They are not intended as numerical, testable benchmarks, but rather a more general guiding principle for the design. Because they are not as easily testable, and often subjective, they should have been tested together in a multi-part test of the robot’s capabilities called the "Acceptance Test", rather than testing each requirement individually. However, these Acceptance Tests were not performed due to lack of time.

1. The conversation flow should be guiding the user through the process such that it is not necessary to have training on how to use the system.
   The conversation flow should be easy to use such that a user does not need specific keywords to trigger an action. The management of a company using this solution would prefer to spend as little time training workers on the use of the system, and if the system mimics natural conversation (meaning it uses a dialogue to instruct the user with the possibility of asking for/using clarifications, similar to how humans often instruct each other), then the user will be able to learn to work with it through experimentation and conversation.

2. The system should be able to communicate to the user when the user asks for something that cannot be performed.
   The worker needs to be explained why the robot was unable to accomplish a task in the event that this happens. It could for example be because the robot misunderstood the user, or because the worker assumed something about the workspace that was not true.

3. When additional input is needed, the system should be able to convey this information to the user.
   The system should be able to query the user for clarification if information is missing for the system. This could be that the user asks the system to pick up but forgets to specify the object to pick up.

4.4.2 Performance Requirements

Performance requirements are quantitative requirements, refined from the user requirements into specific, testable benchmark numbers. For each requirement, we will outline briefly what will be required in order for that requirement to be considered successful. These tests are covered in chapter 6. Notable about the thresholds for the requirements are that they have been chosen while keeping in mind that the system is a prototype. Ideally, striving for getting everything to a 100% success rate would be the goal, but this will not be realistically possible.
Furthermore, it should be mentioned that the thresholds for each requirement are based on what the group deems realistic based on a combination of the delimitation in section 4.3, the related works presented in chapter 2 and the initiating experiments/discoveries when investigating each approach used in the project.

The requirements are:

Integration

1. The NER and task grounding system should be able to extract the correct task 80% of the time.

The NER and task grounding should be able to extract the correct task 80% of the time when the input detects the pick up task. Furthermore, the system should be able to ask for clarifications when it fails to extract the correct task, allowing the user to use a different word/sentence structure. While only the pick up task is integrated in this project, the NER and task grounding modules should be able to recognise the tasks.

2. The NER and command parsing system should be able to extract the object entities 80% of the time.
   The NER and command building system (meaning building a structured representation of the task/object) should be able to extract the correct object(s) 80% of the time, when the objects are known and the input detects the correct words.

3. The system should be able to gracefully recover from incorrect user input errors, by informing the user and performing error correction in at least 90% of possible cases.
   If an error in the pipeline occurs the system should gracefully be able to recover from it. Here gracefully, means that the system does not just crash and that it keeps running. It should then inform the user what went wrong and perform error correction. This has to be done in 90% of the possible cases.

4. The system should, 90% of the time, be able to correctly identify whether the user wants the system to execute a task, learn a new task or learn a new object.
   The system will ask the user what they want it to do. 90% of the time, the system should then be able to correctly identify whether the user wants the system to execute a task, learn a new task or learn a new object. While the learning a new task or object commands are not integrated in this project, the NER system should be able to recognise the commands.

5. The system should, 90% of the time, be able to correctly move to and grasp a specified object.
   During a pick up task, the robot is told to move to a specified object and grasp it. It must be able to do this correctly 90% of the time.

Assembly assistance

1. During the assembly task, the system should be able to detect the current progress off the assembly 90% of the time. *When asked to assist with the assembly process, the system must be able to discern the current progress of the assembly using computer vision methods. The system should at least be able to correctly discern the assembly progress 90% of the time.*

2. During the assembly task, the system should be able to correctly figure out what part is needed for the next step in the assembly process. *After the system has recognised the progress of the assembly it must be able to correctly discern what part is needed for the next step of the process.*

3. During the assembly task, the system should be able to correctly pick up the next object needed for the assembly 90% of the time. *After the system has figured out the next part*

needed for the next step of the assembly, it must be able to pick up the needed part 90% of the time.

4.5 Final Problem Formulation

Following the findings in the previous sections of this chapter, it is possible to formulate a problem statement which this project will set out to answer:

- How can a Human-Machine Interface be integrated in an industrial setting to help with assembly of products?
  - How can a robot system be integrated on the Franka Panda arm?
  - How can a robot system developed for simulation be integrated into a physical environment?
  - How can a robot system assist a user in assembly of a dummy phone?

Chapter 5

Implementation

5.1 The setup

The setup where the integration will be performed is a platform containing the equipment described in Chapter 3. The platform was used in another project, and contains two pillars or frames in the corners of the platform. A package, which will be described in detail Section 5.2, has the platform modelled in the robots workspace, meaning the robot will not compute any trajectories that collides with the platform. The setup can be seen in Figure

5.2 Franka robot integration

This section explain how I went about integrating the previous P8 project onto the Franka panda arm, as well as some of the complications and how I dealt with them

5.2.1 Robot controlling

In the previous P8 project, the robot used in the simulated environment was a UR5 robot, which means some changes are needed in order to interface with a Franka panda arm. Previously, another student group had used the Franka panda arm for a project, in which they also developed a interfacing package for the Franka panda arm. [1]

The package uses MoveIt! for controlling the robot, and interfacing with the package itself is done via ROS services. This creates simple commands used for controlling the robot. For integrating, this means that the robot controller modules had to be modified, as it used to use ROS actions to interfacing with the UR5.

The package comes with its own challenges, such as needing 3 PCs in order to interface with the Franka panda arm:

[1] `https://github.com/HuchieWuchie/AAU_franka_moveit`

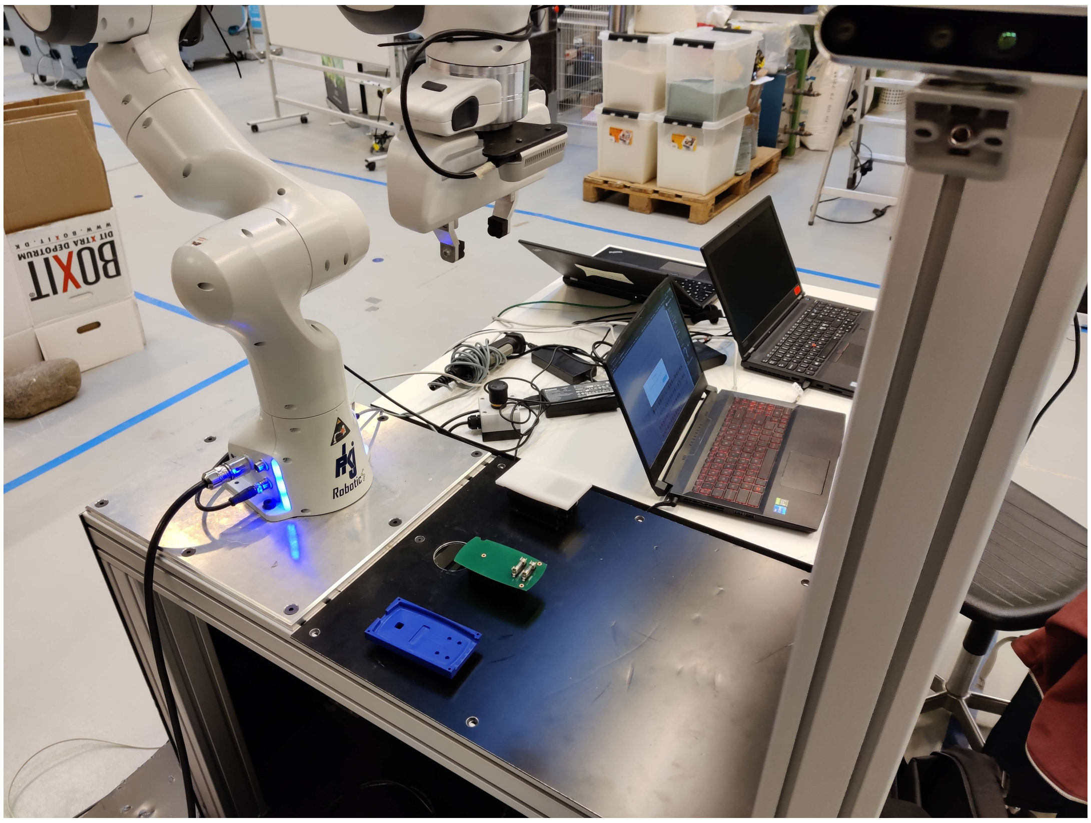


Figure 5.1: The setup at the lab in Fib 14.

- A PC for connecting directly to the Franka panda arm which the arm requires (Interface PC)
- A PC connected via network running the AAU franka moveit package (Franka PC)
- A PC connected via network running the developed program. This PC is also connected to the camera. (ROS PC)

This diagram in Figure 5.2 visualises the setup.
The reason for having a Franka PC and a ROS PC instead of combining them, is because of computing power. Especially if the system uses a neural network it will use up a lot of resources, thus splitting the workload into 2 PCs is a better option.

In the P8 project, the target objects and positions were passed from `dialog_flow.py` to the `robot_controller.py`. This code was responsible for connecting an communicating with a ROS action server which then had all the needed kinematic parameters and calculations needed for the UR5 robot. The code in `robot_controller.py` has been changed to interfacing with the "AAU franka moveit" package through its services.

Using the services there are two ways to make the robot move to a desired position. One way is to generate a `Pose` ROS message, which the package will use moveit to move the robot to. This is done with the `moveToPose(Pose)` service. The other method of controlling is to create

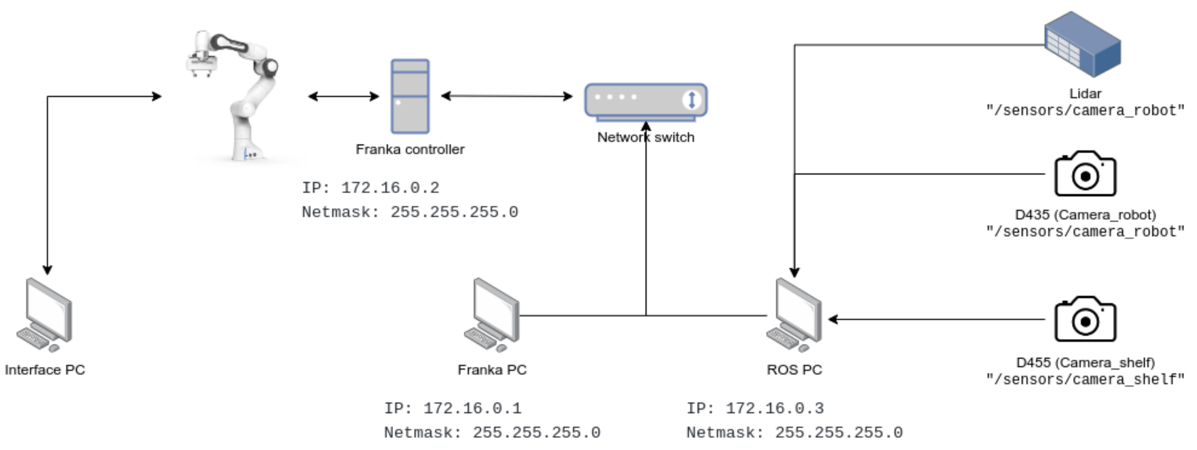


Figure 5.2: Chart showcasing how the devices should be connected using the AAU Franka MoveIt! package [6]

pre-assigned joint values in the package itself. As the package uses move it, it is possible to create groups in the SRDF file, which is good if a project has static poses. This is done with the `moveToNamed(group_state_name)` service.

In order to make integration more simple, it has been decided that parts will have static positions in the workspace. As such, static positions will be added to the AAU franka moveit package. The specifics on the static part placements will be explored in Section 5.3.2

5.3 Camera

As the P8 project was in a simulation environment, the camera used was a simulated one. On the setup there are two cameras present. One is present on the frame of the worktable, another is placed on the hand of the robot. This is very different from the P8 project, as that had the camera placed above the workspace. Since the camera used is an Intel RealSense camera the `realsense2` library is used for interfacing.

However, whenever an image had to be taken in the P8 project the robot had to move out of view. As such, now the robot will use the camera placed on the robot hand, and the robot will instead move in a position such that the camera can get a similar image from the P8.

5.3.1 Speech and NLP

A big difference from the P8 project and this one is the working environment. In the P8 project, it was developed at the home of the group members, which were more quiet environments. Additionally, there were also access to better microphones at our home offices.

This presents a problem when integrating into a factory environment. When communicating verbally with the robot, there might be a lot of noise and background chatter that the robot will pick up on. This could potentially result in the robot getting the wrong command. Two possible solutions were thought of when trying to fix this problem.

The best solution, but also the most time consuming, would be to develop a way for the robot to keep track of the main speaker and isolate their speech from the noise. This would be an ideal solution as the system would retain its verbal conversation capabilities.

However, due to time constraints, a more quick-fix solution was chosen instead. This was chosen with the full knowledge that it would not be an ideal solution and remove the verbal aspect of the conversations. The solution proposed would remove any kind of text-to-speech and speech-to-text capabilities and replace them with a console interface. This means the only may of communicating with the robot would be via text through the console. The input and output of the console would be the same as when communicating verbally, but the it would just be written instead of spoken.

5.3.2 Part placements

As mentioned in Section 4.3, the placement of the dummy phone parts are in a static position on the platform. The specific parts used for the project is a white front casing, PCB and blue back casing. The 3 parts are placed near the base of the robot, as can be seen in Figure 5.3.
The parts are also placed upon smaller "platforms", in order to elevate them and reduce the risk of the robot crashing its gripper into the table. Ideally these platforms would be designed for the specific parts. However, due to lack of time a quick solution had to be made, and random objects around the lab were used as platforms instead. This makes setting up the parts more uncertain and take more time, due to the nature of the objects. The objects can be seen in Figure 5.4.
It does not take a lot to figure out that this was a last minute solution. It will work for the purposes of this project, but special made fixtures definently have to be made later on.

5.4 Errors encountered in integration

This section will highlight some of the errors and complications encountered in the integration process, and how they managed or fixed.

5.4.1 Franka robot

While the package is useful and easy to use, it is not perfect. As mentioned in its documentation, the robot will sometimes encounter an error when launching the program or when moving. One case is for when the launching the program, the robot will not be able to move

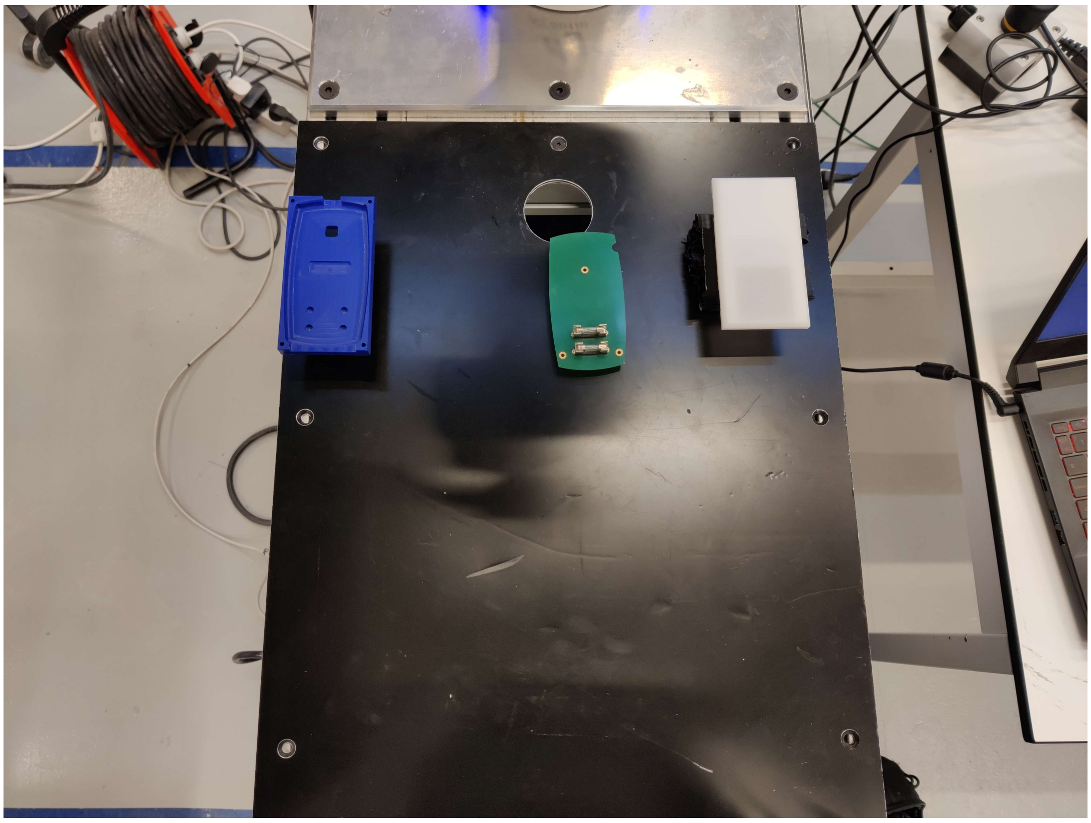


Figure 5.3: How the parts are placed on the worktable. The Franka robot is located at the top of the image

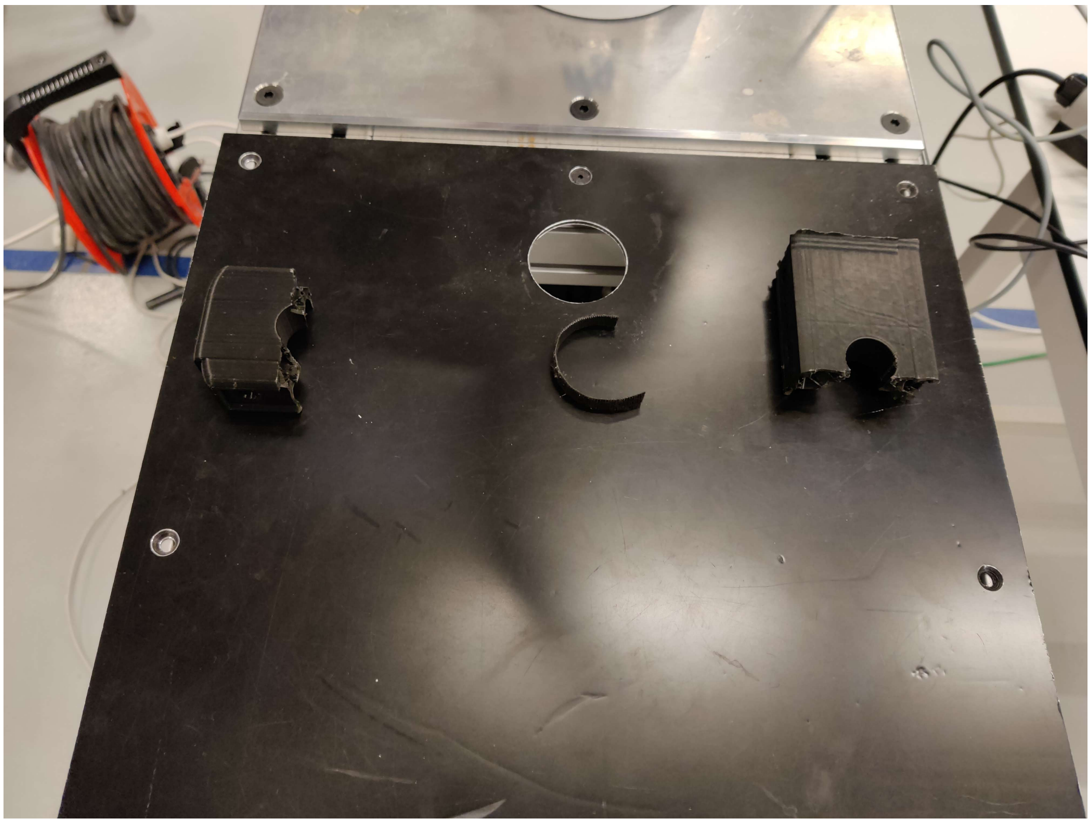


Figure 5.4: The rudimentary platforms for the phone parts

due to a reflex error. This error should occur when the robot encounters a self collision, but it is usually a bug as has only occurred when the robot was in a safe position. Resetting the program and pressing a button on the robot itself usually fixes the error.

Another, and more troublesome error, is a "`GOAL_TOLERANCE_VIOLATED`" error. The documentation notes that this error is likely due to constraints that needs to be fine tuned in MoveIt!. Experiences from integration shows this error to more likely occur when grasping objects. This suggests it is likely a gripper problem, due to it being a binary state of being fully open or fully closed, but when grasping an object the gripper will not fully close, which the robots does not like.

Trying to make the errors less likely to appear, the fully closed position of the gripper was changed to have a width of a bit smaller than the phone parts. This makes the amount of goal tolerance violated be smaller than if it was fully closed. Additionally, a configuration file was modified to allow for more tolerance for the gripper. This resulted in the error to be less likely to occur, but they still happen from time to time. In order to fully fix this error, more time needs to be spend with the robot to fine tune the tolerance levels.

5.4.2 Vision

A big difference from the P8 project to this project is the interfacing with the camera. The P8 project used a ROS Action client/server solution to interface with the camera. This was due to the simulation environment used, Webots, which required this method to get an image from the simulation environment. As mentioned, now the program will interface directly with the camera, bypassing ROS in terms of interfacing with it.
Due to this change, a bug in the P8 code was discovered. In the main logic of the code it would initialise the camera twice, which would work for a ROS Action client/server solution. However, when interfacing with the `realsense2` library only one camera stream can be initialised at a time. This would cause errors in the code as it would try to initialise the camera twice, this was easily fixed and worked perfectly after the fix.

5.5 Assembly assistance module

This section will describe the development of the new module, Assembly Assistance, and how it works. As mentioned in Section 4.3, this module was developed over the Christmas holiday when there was no access to the lab and subsequently the robot. As such, work has been done where it was possible, and a basic prototype of the module has been developed from home. This also means some of the equipment used are technically better that the equipment on the robot, such as the phone camera and lighting setup. After the project is finished, implementing the module on the robot system would be a good next step of the process.

5.5.1 How the assembly works

In Section 3.3 the phone parts themselves were explained a bit. For the assembly of the phone, for this project, the process can be broken down into steps:

- Step 0: No parts placed
- Step 1: The blue bottom cover is placed
- Step 2: The PCB is placed in the blue bottom cover
- Step 3: The white cover is snapped onto the assembly

When trying to figure out what part is needed for each step, it goes as follows: Step 0, the blue bottom cover is needed. Step 1, the PCB is needed. Step 2, the white top cover is needed. Step 4, no part is needed and the assembly is finished.

5.5.2 Module process

The idea for this module is to implement a task command sentence, such as "help me assemble this product", where the robot will then take an image of the assembly area. It will then analyse the image, which would then output the current progress of the assembly. Ideally, this module would have used a CNN implementation, however due to time constraints a more rigid and constrained solution was found using more classic computer vision methods.

A general flowchart of the module can be seen in Figure 5.5
After the module has found the next object needed for assembly, the main controller would then execute a pick up command for the specified object.

Image processing

This part will describe the details of the image processing, for the module to be able to classify the step of the assembly. What the modules specifically does, is perform colour detection on the image. It looks for blue, green and white colours, due to the colours of the objects. This does make the solution very specific to this project and does not allow for much flexibility in the assembly. An example would be that you can not swap a part out for another colour.

A flowchart of the process done on each colour, meaning blue, green and white, can be seen in Figure 5.6.
The module uses OpenCV for the image computations. First it converts the RGB image to an HSV image, as it is a useful colour space to work with when performing colour detection. Upper and lower bounds for thresholds of the three colours are created in HSV format, which dictates what pixels are considered to be the specified colour. The thresholds are as shown in Table 5.1.

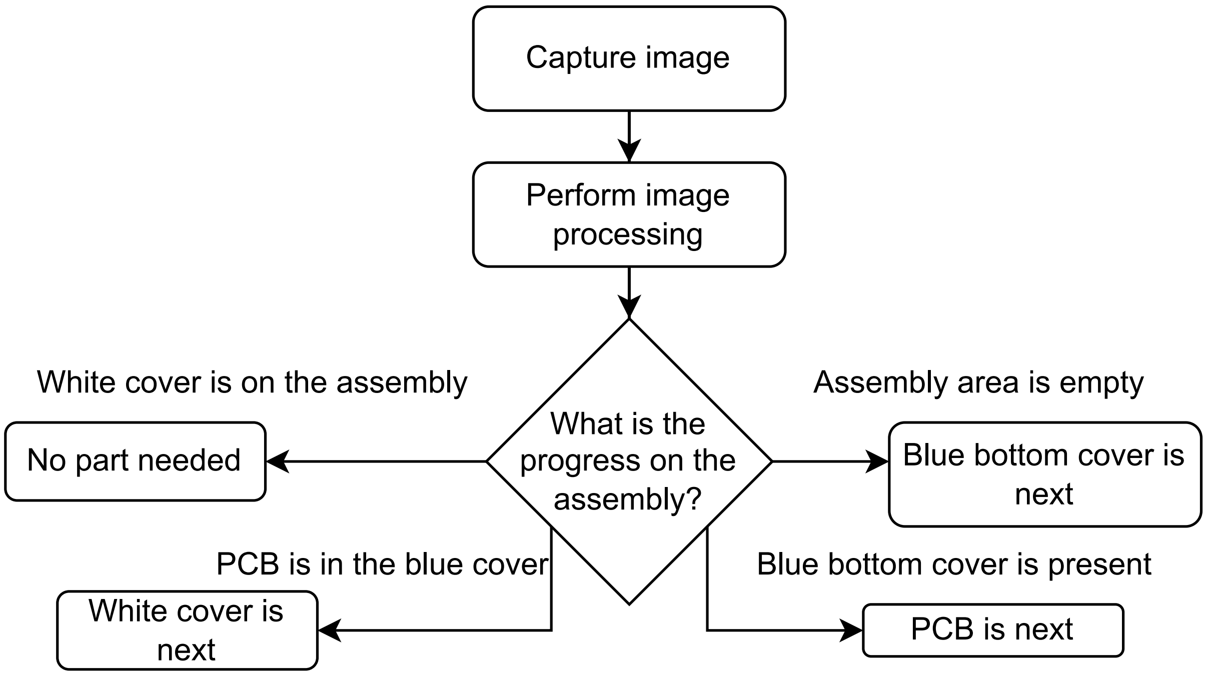


Figure 5.5: A general flowchart of how the assembly assistance module.

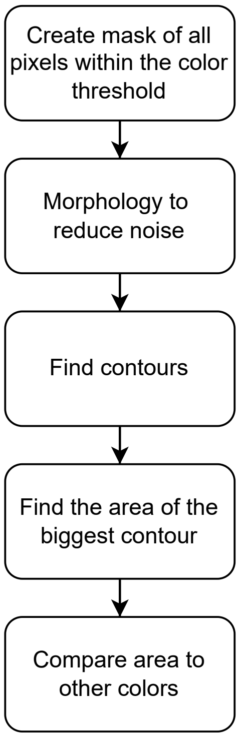

Create mask of all pixels within the color threshold
Morphology to reduce noise
Find contours
Find the area of the biggest contour
Compare area to other colors

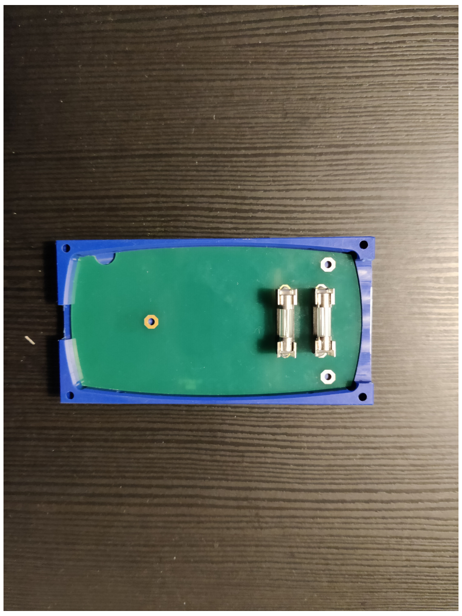
Figure 5.6: Flowchart of the image processing of the assembly assistance module

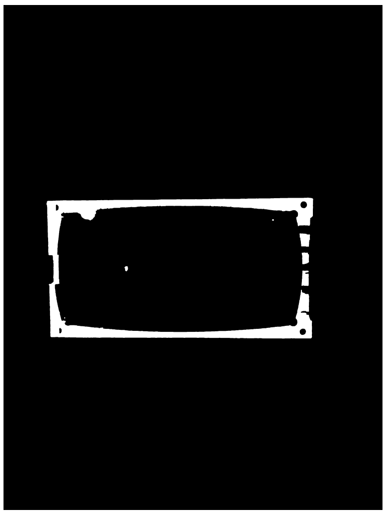
	H	S	V
Blue	[110,130]	[50,255]	[50,255]
Green	[25,102]	[55,255]	[72,255]
White	[0,255]	[0,70]	[185,255]

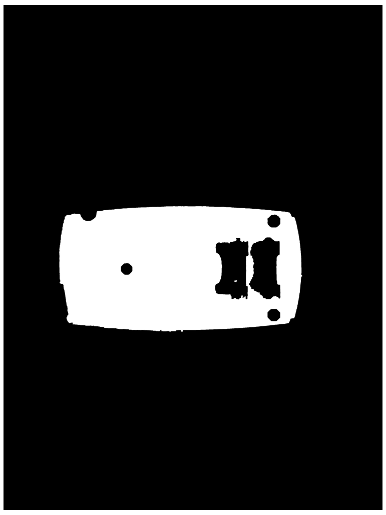
Table 5.1: The values presented in the table are the values that OpenCV accepts, which is [0,179] for H and [0,255] for S and V

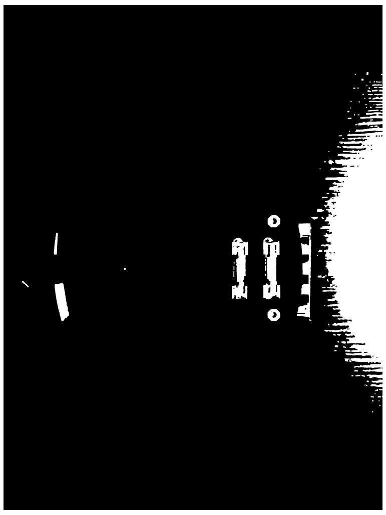
These thresholds are what what decides the performance of the colour detection, meaning it is important that they are set to correct values for the project. With the images the module was developed on, the values are good. In Chapter 6 the module will be tested on different images to see how it performs.

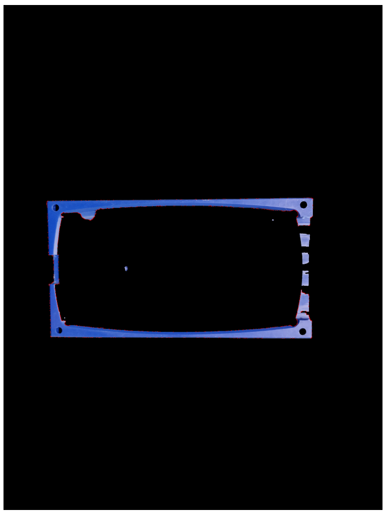
After the colour masks have been found, the masks will be noise reduced using morphological opening and closing operations from OpenCV. Then the OpenCV function `findContours()` is used to generate contours of the blobs in the thresholded image. This will return a list of all the contours found, but the interesting contour is the contour with the largest area, which will be the area of the largest object present in the image if the colour thresholds are correct. Images of the process can be found in Figure 5.7:

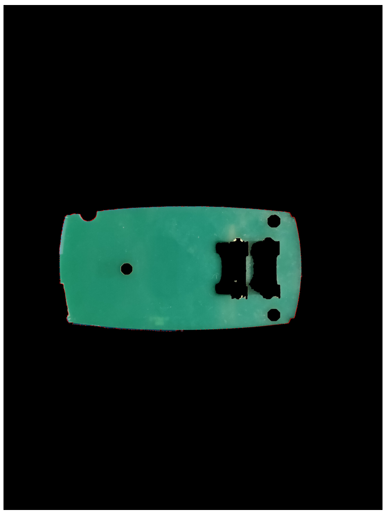

(a) A


(b) B1


(c) B2


(d) B3


(e) C1


(f) C2

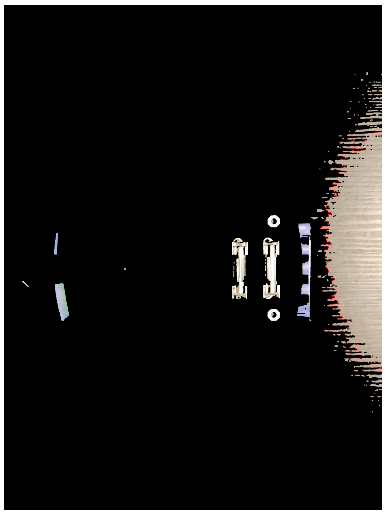

(g) C3

Figure 5.7: Figure A is the image that is being process. Figures B1-3 are the segmented images. Figures C1-3 are the masks with the largest contours drawn.

Chapter 6

Testing

This chapter contains all the tests carried out after the prototype was considered finished after the integration. The tests are based on the requirements laid out in section 4.4.2. Due to restricted access to the lab at Fib 14, not all of the tests have been carried out in time, so only the tests that have been able to been performed from home have been done. The tests that are not performed will still be described in how they would have carried out. A demo video of the robot performing a pick up task and the results for all the tests can be found in the footnotes [1] [2] The tests have the structure as described her:

- **Requirements**: The specific requirement(s) being tested.
- **Focus**: The area of the system this test is focusing on and the purpose of the test.
- **Setup**: The specific environment set up, such as objects in the scene, starting positions and whether part(s) of the system are disabled or bypassed. A "normal setup" in this project means that at least one of each item is arranged on the table in a random location but ensuring they do not touch. For all tests except the full system test, errors originating from modules not being tested at that moment will be ignored, manually corrected or the test run will be repeated. This ensures that the modules are tested in an isolated state.
- **Procedure**: The description on how the test will be performed.
- **Expected Outcome**: How the system is expected to behave if it were to pass the test.
- **Test Requirements**: What specific metrics must be exceeded for the test to pass.
- **Results**: The numerical results of the test, an assessment and the binary final result for the test.

[1]Demo video: `https://drive.google.com/file/d/11rP1y7Rmp6BI1RMQKZ8hadszgyBZE31f/view`

[2]Test results: `https://docs.google.com/spreadsheets/d/1YN279k4O3oiciDD6BHKxFNkuYUEYzDUpwlLrRLDQTjE/edit?usp=share_link`

6.1 Test case 1 - Task extraction

- **Requirements**:
  - Performance requirement 1: The NER and task grounding system should be able to extract the correct task 80% of the time.
  - Performance requirement 4: The system should, 90% of the time, be able to correctly identify whether the user wants the system to execute a task, learn a new task or learn a new object.
- **Focus**: This test will test whether the NER and grounding modules are working as expected.
- **Setup**: <Insert some images, if consistent setup, use that>
- **Procedure**: The system will get a command input that wither specifies a task or a learning command. The test will be preformed 20 times. 10 will contain a task command and 10 will contain a learning command.
- **Expected Outcome**: While the full system only has a pick up task integrated, the NER and grounding modules should be able to discern the correct command and tasks.
- **Test Requirements**: The test is considered passed if the system extracts the correct tasks and commands 80% of the time.
- **Results**: The results are as follows:

	Task	Learn
1	YES	YES
2	YES	YES
3	YES	YES
4	YES	YES
5	YES	YES
6	YES	YES
7	YES	YES
8	YES	YES
9	YES	YES
10	YES	YES

Table 6.1: The task column indicates the successes and failures when a task command is given, the learn column indicates the successes and failures when the learn command is given.

As can be seen the results indicate a perfect test score of s 100% success rate. This is an improvement from the P8 project which had 90% success rate. A reason for this could be due to not having a Speech-to-Text integration, as that was a big failure point on the P8 project. Regardless, this test is considered passed.

6.2 Test case 2 - NER Object extraction

- **Requirements**:
  - Performance requirement 2: The NER and command parsing system should be able to extract the correct object entities 80% of the time.
- **Focus**: This test will test whether the NER and command parsing systems are able to correctly extract the object entity from a given sentence input.
- **Setup**: <Insert some images, if consistent setup, use that>
- **Procedure**: The system will start and will be given a command of performing any task on an object. This object entities given to the system will be: the white cover, PCB and the blue back cover. Each entity will be tested 10 times.
- **Expected Outcome**: The system is expected to recognise the object entities of the white and blue covers correctly, but the PCB is expected to be more difficult due to experiments from the P8 project.
- **Test Requirements**: The test will be considered passed if the overall success rate is at least 80%
- **Results**: The results are as follows:

	White top cover	PCB	Blue back cover
1	YES	YES	YES
2	YES	YES	YES
3	YES	YES	YES
4	YES	YES	YES
5	YES	YES	YES
6	YES	YES	YES
7	YES	YES	YES
8	YES	YES	YES
9	YES	YES	YES
10	YES	YES	YES

Table 6.2: White top colour column indicates success and failures when giving the system a white cover object. PCB is for when giving the system a PCB object. Blue bottom cover is for when giving the system a blue bottom cover object.

As can be seen in the results, these also show a success rate of 100%, which is again an improvement from the P8 project likely due to the same reasons. This test has been considered passed.

6.3 Test case 3 - Error recovery

- **Requirements**:
  - Performance requirement 3: The system should handle errors from incorrect user input, by informing the user 90%.
- **Focus**: This test will test whether the error handling is working as expected.
- **Setup**: <Insert some images, if consistent setup, use that>
- **Procedure**: The system will start, and be given an incorrect user input. This can be words or entities the system does not know, tasks or objects the system has not learned. This will be done a total of 10 times.
- **Expected Outcome**: For unknown sentence inputs, the system is expected to inform the user that it does not understand the input and will ask for the user to input a new sentence. For unknown objects or tasks, the system is expected to inform the user that it does not know of the object or task and then ask if the user wants the robot to learn it.
- **Test Requirements**: The test is considered passed if the success rate is at least 90%
- **Results**: This is one of the tests which were not able to be completed due to restricted access to the Fib 14 lab during the holidays. The physical setup was needed, as various failure points needed to be addressed in order for this test to have any meaning.

6.4 Test case 4 - Movement and grasping

- **Requirements**:
  - Performance requirement 5: The system should be able to correctly move to a specified position and grasp an object 90% of the time.
  - Performance requirement 3: During an assembly task the system should be able to correctly pick up the next object needed for assembly 90% of the time.
- **Focus**: This test will test whether the system is capable of moving to an object and grasping it, without encountering and error.
- **Setup**: <Insert some images, if consistent setup, use that>

- **Procedure**: The system will be asked to perform the pick up task, 10 times for each object, for a total of 30 times. One of the requirements tested here does specify during an assembly task, but the grasping part of the assembly task uses the pick up procedure, so the requirement is still valid.
- **Expected Outcome**: The system is expected to correctly perform the task 99% with an occasional MoveIt! error being expected.
- **Test Requirements**: The test is considered passed if the system performs the pick up task without encountering an error with a total success rate of at least 90%.
- **Results**: This is the second and last test which was not able to be completed due to lab restrictions during the holidays. As this test is mainly about whether or not the Franka integration was successful, it would not have made sense to do a simulation test instead.

6.5 Test case 5 - Assembly process progress recognition

- **Requirements**:
  - Performance requirement 1: During an assembly task, the system should be able to detect the current progress of an assembly 90% of the time.
  - Performance requirement 2: The system should be able to correctly figure out the next object needed for an assembly procedure 90%.
- **Focus**: This test will measure the assembly assistance module, mainly focusing on the vision aspect.
- **Setup**: <Insert some images, if consistent setup, use that>
- **Procedure**: The system will be asked to perform an assembly assistance task. The system will the take an image of the assembly, process it and classify it as a certain step in the process, then based on that classification determine what part is necessary for the next step.
- **Expected Outcome**: The system is expected to correctly classify the assembly step and figure out what part is needed
- **Test Requirements**: The test is considered passed if the system classifies the correct step 90% of the time, and figures out the correct object for the next step of the assembly.
- **Results**: The results are as follows:

	Progress presented	Progress recognised	Correctly recognised?
1	Step 1	Step 1	YES
2	Step 1	Step 1	YES
3	Step 1	Step 1	YES
4	Step 2	Step 2	YES
5	Step 2	Step 2	YES
6	Step 2	Step 1	NO
7	Step 3	Step 3	YES
8	Step 3	Step 3	YES
9	Step 3	Step 3	YES
10	Step 3	Step 3	YES

Table 6.3: The first column indicates what step in the assembly process that was presented to the robot, the second column indicates what step the system recognised it as, and the final column indicates whether the individual test attempt was a success or not.

As can be seen from the results, there was a failure point when the robot was presented an assembly at step 2, but the system classified it as from being at step 1. This means the biggest contour from the blue segmentation was bigger that the biggest contour from the green segmentation. Due to lighting variations the threshold was not robust enough to deal with that change in the environment. There are several fixes for this, both continuing in the classical computer vision direction or switching the direction to involve a CNN. These ideas will be discussed more in Chapter 7. It is very likely that if the test had been conducted at the lab, that there would have been more failures due to big changes in lighting and camera. Regardless, this test has been considered passed.

6.6 Test summary

The overall results from all the tests can be seen here:

Test case #	Test case name	Results
1	Task extraction	PASSED
2	NER Object extraction	PASSED
3	Dialog error recovery	INCONCLUSIVE
4	Movement and grasping	INCONCLUSIVE
5	Assembly process progress recognition	PASSED

Table 6.4: Tables showing which tests were passed, failed and inconclusive

As can be seen, majority of the test cases have been deemed passed. However, due to circumstance test case 3 and 4 were unable to be completed in time, and this it is not known whether they can be deemed as passed or not. From the development phase, it is likely that test case 3 would have been failed, due to the Franka MoveIt! bugs described in Section 5.2. Another thing to note, is that test case 5 is likely to perform much worse when applied to the robot system at the lab, due to big changes in both environment and camera equipment. It is likely that a new approach has to bee thought of when it comes to the assembly recognition system.

Chapter 7

Discussion

This section will make an assessment of the prototype based on the results from Chapter 6. It will start off with discussing the implementation and system performance, and then move on to feasibility and future improvements.

7.1 Summary of implementation

In section 4.1, a vision for how the final prototype was idealised, and this project set out to develop a physical implementation of this ideal prototype idea.
This was broken down into involving two phases: an integration phase, and a development phase. The integration phase was about integrating the previously worked on P8 project onto the Franka Panda robot arm, and the development phase was about developing a module for an assembly assistance feature.

7.1.1 Integration phase

The biggest challenge in the integration phase was converting the system to be compatible with the Franka Panda robot arm. It ended up being an integration that relied on the assembly parts being placed on fixed locations on the worktable. Additionally, only the pick up task was implemented, meaning all the additional tasks the P8 project was capable of were not integrated to the system. The Franka integration was done using the AAU Franka MoveIt! package, which included a simple interfacing method to the Franka arm using ROS services. It did come with its own problems, such as the robot sometimes stopped moving, or some fine tuning was needed for the gripper.

Other challenges included camera integration, both in terms of the P8 code not having proper handling of a physical camera connected to the PC, and in terms of different environment conditions and camera parameters. Additionally, the language interfacing system with the robot was reduced to being a console interfacing system, due to the noisy environment a production workshop. The NER system was still integrated properly, meaning the console inputs would just be what you would say to the robot and it would handle it as it was expected to.

7.1.2 Development phase

This phase sought out to develop a module to the system that was able to detect the progress of an assembly, and then be able to decide what part was needed for the next step of the assembly. This was developed during the Christmas holidays, meaning there was no access to the robot, as such it would have to be developed with regards to images taken from home.

The solution involved colour detection of an image with an assembly in progress. Depending on what colour area was the biggest it would decide what the progress of the assembly was on. This solution was very tailored to this specific project, meaning if other coloured parts of the phone were used instead it would not work as expected. Additionally, the colour thresholds were also tailored to the images taken from home, which was done in completely different lighting environment and using an different camera than the ones at the robot system.

7.2 System performance

In chapter 6 various tests were performed on the system, in order to evaluate the performance and to determine if it lived up to the requirements decided in section 4.4.2. In total, 5 tests were constructed, but only 3 of them were able to be completed due to some of the tests requiring the test to be performed on the physical system at the lab.

Test case 1 and test case 2 were about the performance of the NER system, in order to evaluate the performance of this modules after changing the method of input. The results were great for this part, as it had a success rate of 100%, which was even better than the tests conducted in the P8 project. The P8 project had a success rate of 95%, while still meeting the requirements, is lower success rate than the tests made in this project. A big reason for this is using a console input, rather than using speech-to-text input. The speech-to-text system used in the P8, while still good, was not perfect. Combining that with microphones not being perfect either, even in a quiet environment, meant that more failures were prone to happen. The draw-back with using the console input is that at that point you might as well remove the NER aspect and just just normal commands.

Test case 5 was about the assembly assistance module. The test was conducted on images taken from home, at different times of the day. The tests had a 90% success rate, which makes the test be considered passed. However, it is very likely that this module will perform much worse on the environment at the robot, due to change in camera and lighting. In order to make it work in the context, some camera parameters might need to be adjusted, as well as parameters for the colour detection threshold.

Test case 3 and 4 were not conducted as they require access to the physical robot in order to properly test them. The expectations of these tests are that test case 3 would be a successful test, as not much has been changed to the error recovery actions during the integration part of the implementation, meaning the best metric of performance for this test is the P8 project's test on error handling, which had the test be passed. Test case 4 is expected to be passed, while still encountering some amount of errors, due to the few bugs present in the AAU Franka MoveIt! package. This expectation is based on the work done during the integration phase, and becoming familiar with how the movement and grasping works for the system.

7.3 Feasibility

From the previous sections in this chapter, only some of the essential parts of the system got integrated. For this product to be considered fully finished it must be fully integrated, such that all the modules work on the system. This requires more fine tuning to the camera and the robot control, which would expand the capabilities of the system so it can do more than pick up parts on static positions.
Additionally, the assembly assistance module was developed away from the lab environment, so more work would have to be put into to that module for it to be fully implemented on the system.

The prototype does demonstrate that integrating the P8 system onto a different robot than what it was designed for is possible, and it also demonstrates one possible road to follow for implementing the capability of assisting with the assembly of a product.

7.4 Future development

There are a lot of improvements to be made to the prototype. Many of them involve adjusting various parameters to make the system more reliable, such as the position tolerance for the robot control.

As mentioned previously, some of the tasks the P8 project was capable of were not integrated. This would be a good idea to try to integrate, as it would make the system more like the previous project. This would require re-adjust some camera parameters, as the camera on the physical system is very different than the one in the P8 simulated system. If this would be done, then the system would be able to use its object detection capabilities and feature extraction capabilities to its full extend.

Another thing to improve on is more stochastically placed phone parts, so the system is not restricted to static locations. This would need some changed in the robot controller code, as at the moment the static locations are hard coded into the AAU Franka MoveIt! package. The original P8 code already has the ability to convert image coordinates into world coordinates, but some work would have to be done in order to convert those coordinates into robot poses to be sent to the MoveIt! controller.

The assembly assistance module is a module that has a lot of improvements to be made. While the performed tests indicated it performed well, this was with the understanding that it was not in the laboratory environment. A lot of work would have to be done on the colour detection part for the system to perform reliably in that scenario. It would also be smart to use another method of deciding what step the assembly is on. At the moment, it decides it based on the size of the largest contour for each colour. Instead of using the largest contour, a way could be to count the amount of pixels of each colour mask. An even better method would be to develop and implement the module with a CNN that has been trained on images of all the steps. This would have a somewhat high development time, but the end result would also be a lot more robust.

Finally, the interfacing with the system also has to be looked at one more time. The P8 system used a verbal communication interface, while this project used a console interface. While the console interface is more reliable, it defeats the one of the purpose of the P8 project, which was to develop a robotic system that you interfaced with using spoken language.

Chapter 8

Conclusion

This project set out to integrate a previous project onto a physical robot system, while also developing a feature to assist the user in assembling dummy phones. The problem formulation were constructed, which the project sought out to answer:

- How can a Human-Machine Interface be integrated in an industrial setting to help with assembly of products?
  - How can a robot system be integrated on the Franka Panda arm?
  - How can a robot system developed for simulation be integrated into a physical environment?
  - How can a robot system assist a user in assembly of a dummy phone?

The proposed solution was an integration of the previously worked on P8 project, which was developed for a simulation environment on a UR-5 robot, and was integrated on a Franka Panda robot arm system. The integration on the Franka Panda arm was done using the AAU Franka MoveIt! package, which was a package created for simple interfacing with the Franka Panda arm. The integration was also done using static locations for the phone parts, which was a different approach from the P8 project. During the integration process the several challenges came up, such as interfacing with the camera and various bugs in the AAU Frank MoveIt! package. The camera challenges were errors there were not noticed in the P8 development, due to it having a different camera interfacing method in the simulation, but were fixed in integration. Additionally, a module to assist in assembly of module phone were also created. This method used colour detection methods to detect the amount of colour in an image, which dictated which step the robot recognised the assembly to be at. It was then able to decided what part to pick up based on the step recognised.
The integration did not include all of the capabilities of the P8 project, due to time and manpower, but it shows that it is possible and more work needs to be put into integration in order for the full P8 system to be integrated. The assembly assistance capabilities were also developed at a period of time where access to the robot was limited, to the specific parameters are tuned for a different environment and camera. Thus, more work needs to be done of fine tuning the module on the robot.

Bibliography

[1] Xi Vincent Wang et al. 'Human–robot collaborative assembly in cyber-physical production: Classification framework and implementation'. In: *CIRP Annals* 66.1 (2017), pp. 5–8. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2017.04.101. URL: https://www.sciencedirect.com/science/article/pii/S0007850617301014.

[2] L. Wang et al. 'Symbiotic human-robot collaborative assembly'. In: *CIRP Annals* 68.2 (2019), pp. 701–726. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2019.05.002. URL: https://www.sciencedirect.com/science/article/pii/S0007850619301593.

[3] Hongyi Liu et al. 'Deep Learning-based Multimodal Control Interface for Human-Robot Collaboration'. In: *Procedia CIRP* 72 (2018). 51st CIRP Conference on Manufacturing Systems, pp. 3–8. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2018.03.224. URL: https://www.sciencedirect.com/science/article/pii/S2212827118303846.

[4] Pomo robotics. *Franka Emika Panda arm*. URL: https://www.pomorobotics.com/wp-content/uploads/2021/10/Franka-Emika-Panda-square.png (visited on 2023-04-01).

[5] Jens F Buhl et al. 'A Dual-arm Collaborative Robot System for the Smart Factories of the Future'. In: *Procedia Manufacturing* 38 (2019), pp. 333–340. URL: https://doi.org/10.1016/j.promfg.2020.01.043.

[6] Albert Daugberg Christensen. *AAU Franka MoveIt!* URL: https://github.com/HuchieWuchie/AAU_franka_moveit (visited on 2023-04-01).

Appendix A

Previous project module diagram

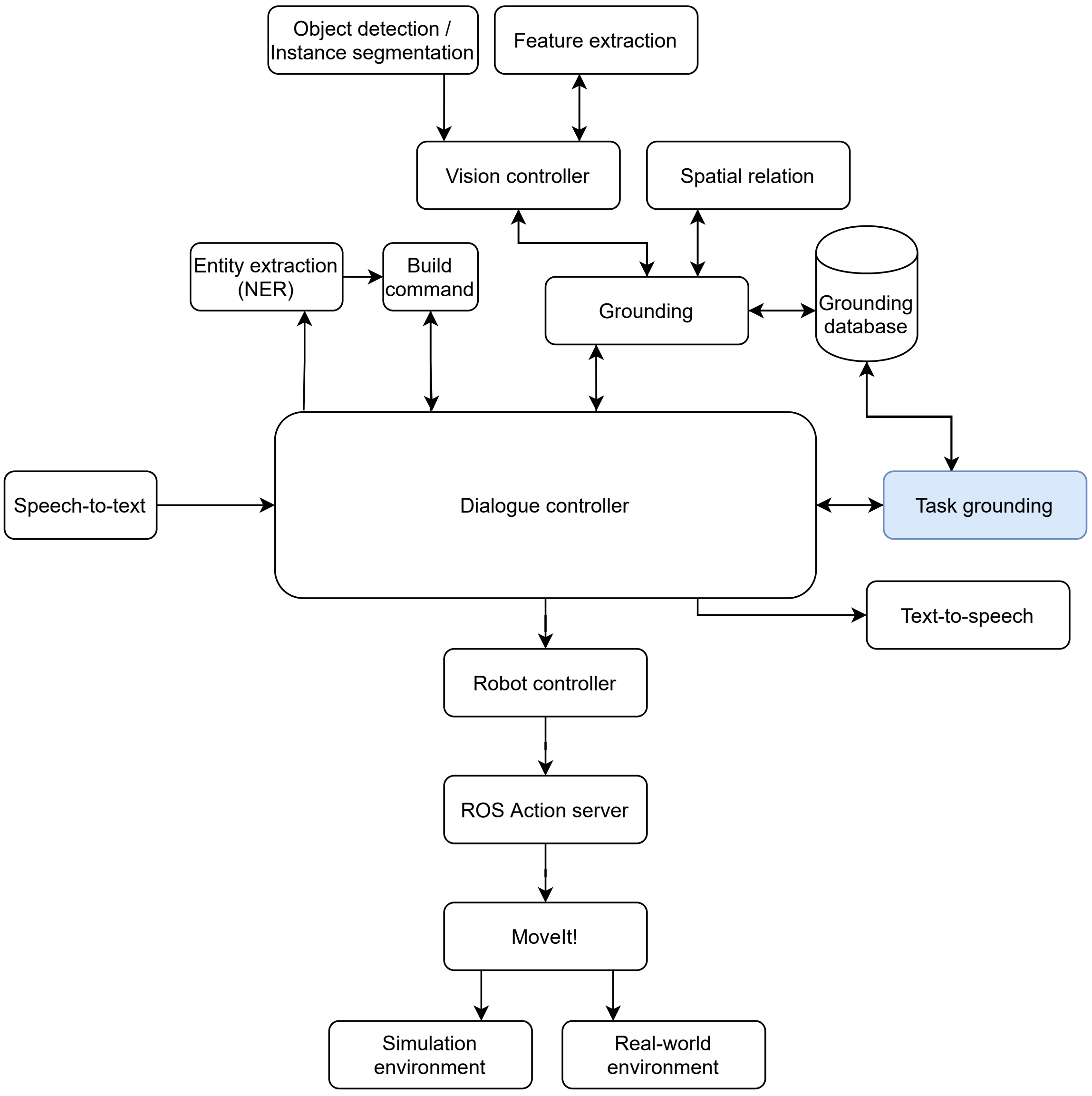


Figure A.1: Diagram of the different modules in the P8 project and how they connect to each other.