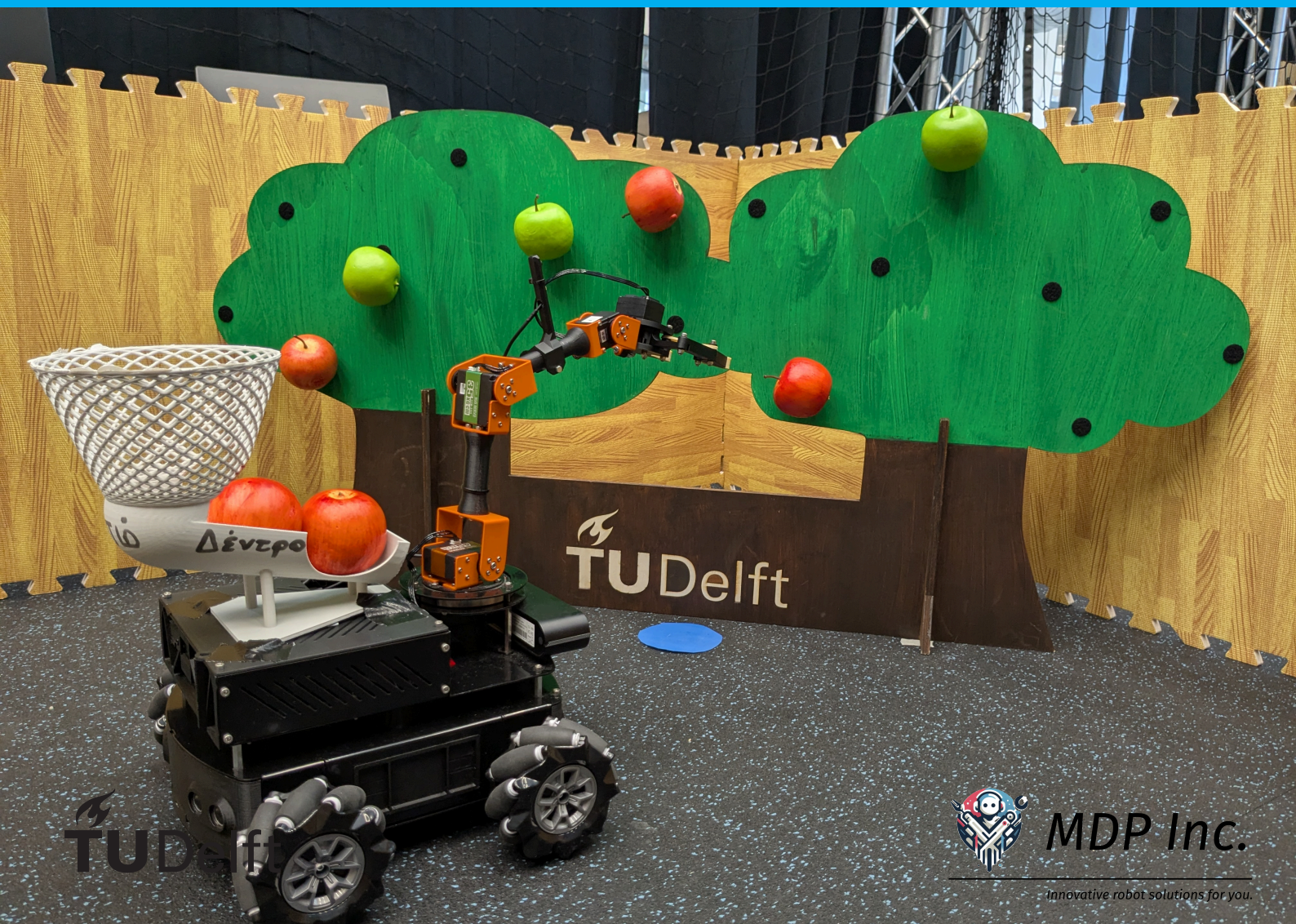
Apple-picking Robot

MDP Inc. - Group 11

RO47007 Multidisciplinary Project – 2025

A. Kourris, D. Kritharoulas, M. Saravanan, N. Termote,
T.J. van der Weij



Preface

Template version: 2025.1

Welcome to the MDP

Dear students, we are excited to welcome you to the Multi-disciplinary project 2025! The MDP is the finishing milestone of your first year, allowing you to bring the theory that you have learned during the first year into practice and giving you time to reflect on your specialization and second year. In this project, you will be a member of a 5-person development team for the fictitious consultancy firm *MDP Inc.*, a Robotics consultancy firm of the Department of Cognitive Robotics that provides initial innovative robot solutions to their clients. Our new client, the United Nations, has tasked *MDP Inc.* to provide robot solutions to their real-world problem. Each team from MDP Inc. will be working on creating a unique solution to solve this problem. During the development, the teams will be supported by the MDP staff, be provided with a mobile robot innovation platform, MIRTE Master, and a flexible workspace has been reserved for each team; please have a look at Brightspace to find more information.

Learning outcomes of the MDP

The MDP is a unique course that focuses on developing your practical skills as well as providing you with space to grow personally. These skills are detrimental for your career after graduation, but also for the final part of your study. The learning objectives for this course are divided over two domains (see also description in TU Delft Study Guide):
1) Knowledge, insight, judgment and skills: By the end of the course, you should be able to:

LO 1.1: Define a robotics use case with its functional and non-functional requirements.

LO 1.2: Design robotic solutions by integrating knowledge of system engineering, current opportunities for robots, trends in robotics, and societal aspects.

LO 1.3: Produce a robotic solution that meets the design specifications and considers sustainable, safety, ethical, economical and other relevant societal aspects.

LO 1.4: Use functional architecture for planning and communicating robot software.

LO 1.5: Communicate the multidisciplinary robot solution orally, in writing, and in code documentation according to robot software development guidelines.

2) Transferable and interpersonal skills: By the end of the course, you should be able to:

LO 2.1: Formulate learning goals for your own personal development.

LO 2.2: Reflect on your competencies (e.g., Teamwork, Leadership, Entrepreneurial thinking, Strategic multidisciplinary problem-solving), their development, and your personal learning goals.

LO 2.3: Apply structured feedback to improve your performance or the performance of others.

LO 2.4: Apply structured multidisciplinary software project management, with the use of different team roles and responsibilities.

Overview of the MDP

The project consists of 5 phases as seen in Table 1 below. In each phase, you will have a set of activities to complete and report on them at the end of each phase. As a team, you will analyze the client's problem and provide a solution to it and present that to the client during the project. After organizing yourself as a team, you will analyze the client's problem and formalize it together with its requirements and how you are planning to address it. You will first design and build something very simple (that works), and over time work from here to get to your final project goal. In the MDP, you will likely work both in simulation and with MIRTE Master, and present your final results to our client,

Table 1: MDP overview with different phases.

Weeks	Phase		Main activities
3.8-3.9	0	Project preparations	Attend introduction, Meet your team, get to know the assignment, ROS install, and GitLab check
4.1	1	Initiation & Planning	Getting organized and planning of your project
4.2-4.3	2	System design	Initial system design and realization
4.4-4.6	3	Implementation & validation	Further design, implementation and robot testing
4.7-4.8	4	Project completion	Finalizing your project and final reporting

including a demonstration. During your project, you will be writing a report using the fixed template that you are reading at this moment. We co-designed the template with industry partners to mimic common elements that development teams have to report, e.g., used for certification of the robot. In each phase, you will complete the content of the required chapters and submit it before the phase's deadline. The teaching team will provide you with feedback on the completed chapters. We recommend you to incorporate this feedback in the final version of the report.

In addition, following on from the RO47000 *Vision and Reflection* course, you will be working on a personal reflection assignment throughout the project. Besides developing your practical robot skills, you will grow personally during the project, e.g., managing a team, presenting your ideas or realizing your strengths. This personal growth will be detrimental to your future career and also the last part of your study, e.g., which topic suits you best for your master thesis. You will individually report your reflection in a separate assignment twice, at the beginning and at the end of the project. For the reflection, you will work with the MDP reflection template that works similarly to this report template.

Learning material

We have invited you to our Brightspace course page that contains all required information on the MDP. Here, you will find the two required templates, for the main report and for the reflection. To support your development throughout the MDP, we have created various self-learning videos, e.g., how to create Gantt charts or how to use your MIRTE Master robot. We recommend you to watch the videos together with your group and discuss the content and how to apply it to your work. We will have selected in-person workshops to discuss the content together, e.g., a systems engineering session to discuss formal system design principles.

Interaction and questions

The MDP is a highly interactive project-based course. You are required to closely work with your fellow group members to realize your robot solution for the client. On top of this, you will likely encounter general problems on software, using the MIRTE Master, working with ROS or similar. During the scheduled group working sessions, we encourage you to work together with your team but also to interact with other teams on common problems. The MDP is *not* a competitive course and will give you the opportunity to collaborate between teams as you will likely encounter later on in your professional life. We have also created a discussion forum on Brightspace that you should use to discuss questions that are relevant for other groups. **Important:** The MDP teaching team will not answer any of such general-interest questions received via mail. Please post those questions on Brightspace so that everyone can see them and participate in the discussion.

Working style

The success of your project depends heavily on your group's working style. Every member will need to contribute to your project. In the beginning of the MDP, we recommend you to sit down and discuss and plan how you want to work together as a group (this will also be part of your reflection deliverable). For instance, you can start with questions such as: when do you want to meet? how do you want to communicate within the group? how do you organize the group? what kind of team rules do you want to apply? Non negotiable is that you treat everyone in your group with respect and follow the TU Delft Code of Conduct. The MDP teaching team does not tolerate discrimination, harassment, uncomfortable and unacceptable behaviors or similar. If you encounter such situations (personally or from other teams),

please do not hesitate to contact the MDP teaching team; we will treat the conversation anonymously.

The MDP is a highly collaborative and time-intensive course. Given the course's 5 credits, you are expected to commit around 18 hours per week to your project. Make sure to properly plan your schedule and properly communicate with your group. It is your group's freedom but also responsibility to organize yourselves and find the best working style on your own.

Deadlines and assessment

All deadlines and deliverables are summarized on Brightspace. **Important:** *continuously consult the MDP's Brightspace page for the latest information on deadlines and deliverables.* Please note that deadlines are final and no extensions will be given to individual groups.

The assessment in the MDP is two-fold. We apply knock-out criteria that you need to fulfill to complete the course but you will not receive a grade for. These include actively participating in all mandatory sessions and presentations (mid and final presentations), submitting all required deliverables on time, and participating in the buddy check.

The final grade is based on your end report, your final presentation and robot demonstration, as well as your final reflection report. We apply specific grading rubrics to determine your grade. You can find the rubrics that we use for assessment on Brightspace. We encourage you to read the rubrics and discuss questions with the MDP teaching team during one of the group-staff sessions. We would like to emphasize that grades are not assigned per group, but per individual student. For this reason, we require you to specify who did what in your deliverables. Thus, you should also distribute the workload fairly among your fellow group members.

Working principles of this document

This document contains the structure of your main report in which you describe the development of your robot solution to the client. The content of the document is inspired by company practices you will likely encounter in your professional life later on. Each chapter in this report corresponds to a specific development phase in the MDP and requires you to focus on a specific part of your robot solution, e.g., the formal system design or the validation. Thus, in each phase, you and your group will complete the required chapter and submit the document on time on the specified deadline. Important: You don't have to complete chapters that are required at later deadlines.

Each of the chapters has a fixed structure that you need to adhere to. We have put instruction text in blue to describe what your group is required to fill in each of the chapters/sections. Carefully read the instructions and remove them before filling your sections. Important: make sure to adhere to the instructions, they are there to help you and to focus on what specifically you need to report, e.g., when we ask you to highlight 3 examples, please only provide 3 examples. Similarly, figure placeholders serve as examples and should be replaced with actual figures. Before submitting your report, make sure to proofread and spell-check your document. You also need to stick to the required page limits and are not allowed to modify the template (e.g., font colors, font sizes, etc.). Submitted reports that do not stick to the requirements, e.g., not using the official font, will be desk-rejected. For the assessment, we would like you to specify who wrote which section in the document. Therefor, please add the names of the authors at the end of the corresponding sections.

Policy on use of generative AI

You are allowed to use generative AI tools (like ChatGPT or GitHub Copilot) to help with your work in this course. However, you must clearly mark any part of your project that was created with AI. For each use, explain why you used it, how you used it (e.g., what kind of prompts or tasks), and how you verified that the result works as intended. Use AI to support your learning — not to skip it. Be honest and reflective in your use.

Change log

Sometimes, your work in later chapters influences content that you have written earlier. You are allowed to make changes to earlier sections. To help us assess your current section and provide valuable feedback, we would like you to color the changes in the color tudelft-warm-purple available in this

template for easier reference for us. Moreover, please refer in your later chapters to the changes so that we are aware and have a look at them.

Last but not least, we wish you a successful MDP and a lot of fun developing your solution and growing personally. The 8 weeks will be filled with a lot of ups as well as some minor downs. Together as a team, you can navigate these challenges and come up with creative solutions to those challenges. We are very much looking forward to what you will create during this year's MDP!

Your MDP Teaching Team

Contents

Abstract

To support small-scale apple farmers facing labor shortages, this project aimed to develop an affordable autonomous apple-picking robot aligned with UN Sustainable Development Goal 2 (Zero Hunger). The proposed solution uses the MIRTE Master platform equipped with a modular design, LiDAR, camera and sonar sensors. The robot autonomously maps its environment, navigates safely to trees, detects ripe apples, and performs grasping and placing actions using a robotic arm. Object detection and SLAM algorithms were implemented alongside planning and control methods such as MPPI. Validation tests confirmed the robot's ability to classify apples, localize objects, avoid obstacles, and communicate with users via a GUI. While limitations remain due to hardware malfunctions and integration time constraints, the core functionalities proved effective. The solution demonstrates the feasibility of a cost-effective and semi-autonomous harvesting system tailored to small-scale farms, offering a promising step toward more sustainable and accessible agricultural automation. (Termote)

A. Kourris, D. Kritharoulas, M. Saravanan, N. Termote, T.J. van der Weij
November 19, 2025

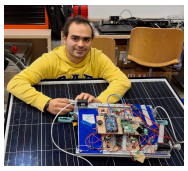
1

Start of the Project

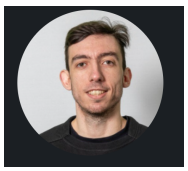
1.1. Meet the team - Everyone

Dentro:
Hello, we are team Dentro! (pronounced as */ðen.dro/*; meaning *tree*) We are a group of passionate robotics students aiming to take robots out of research labs and factory production lines and into real-world agricultural applications.

Nils Termote:
My name is Nils. I am 24 years old and I am from the Netherlands. In this project I will be focusing on the machine perception aspect of our solution and I will be mostly working together with Tjerk. I will personally focus on creating packages for orchard and apple localization while also being responsible for the structure of our weekly meetings by creating meeting agendas and serving as the chairman.



Antreas Kourris:
My name is Antreas. I am 23 years old and I come from Nicosia, Cyprus. I will be working on the pathfinding and controls aspects of the project. My main goal is to ensure that the robot can navigate in the field safely, reliably, and optimally. Aside from my technical responsibilities, I am also responsible for organizing our file-sharing and task management platforms.



Dionysios Kritharoulas:
My name is Dionysios. I am 25 years old and come from Athens, Greece. My main role is focused on the Human-Robot Interaction (HRI) aspect of the project. I will work on implementing a GUI that farmers can use on their phones or laptops to receive valuable information about the robot's status and mission. Moreover, I will aim to enable the robot to interact with the user (e.g., through gesture recognition or buttons in the GUI) and respond appropriately to their commands. Aside from my technical responsibilities, I will also be responsible for code review and documentation on GitLab.

Mukil Saravanan:
I am Mukil Saravanan, an enthusiastic robotics student from India. In this project, I will be investigating different control strategies, including designing and evaluating Non-linear Model Predictive Controller (NMPC) and Model Predictive Path Integral (MPPI) for the mobile manipulator. These target optimally navigating amidst obstacles in the presence of uncertainty while ensuring robustness. Furthermore, I am responsible for keeping track of the deadlines of all the deliverables throughout the journey.

Tjerk van der Weij:
My name is Tjerk and I am a 25-year-old student from The Netherlands. I will be focusing on the Machine Perception side of this project together with Nils. My responsibilities lie in data processing and visualization of the whole sensor suite. Also, I will work on creating and training an effective computer vision model to detect the tree, basket, and differentiate between ripe and unripe apples. Next to the technical part, I will be responsible for proofreading and spellchecking the report before submission.

1.2. Ways of working

As a team we have agreed to meet at the project tables on Tuesday, Wednesday, and Thursday from 8:45 to 12:45 as minimum, making exceptions if a team member has other obligations like lectures at that time. If more time is needed, we have decided to meet at different times during the week. Task delegation was split up as discussed in section 1.1. Nils and Tjerk are working mainly on perception, Antreas and Mukil are working on the path planning and control, and finally Dionysios is working on the Human Robot Interaction aspect of the project. For online communication, we are using WhatsApp, which enables quick communication. Code sharing and collaboration will be done through GitLab. We have a Microsoft Teams Channel to share and store non-code files. Finally, through Teams, we use the integrated Microsoft Planner to keep track of pending tasks that need to be completed. (Kourris)

1.3. Problem definition

Our client is the United Nations which is an international organization founded in 1945 to promote peace, security, human rights, and development across the globe. Our client wants us to focus on SDG 2 (Zero Hunger) [1] to end hunger, achieve food security, improve nutrition, and promote sustainable agriculture. The goal emphasizes that everyone has access to sufficient and nutritious food all year round, especially vulnerable populations. It also supports small-scale farmers, equitable access to land, technology, and markets, and resilient agricultural practices that adapt to climate change and protect ecosystems. Small-scale farmers face labor shortages [4] and challenging working conditions [2] which makes it difficult to harvest their apples and compete with large agricultural companies. Current automation solutions can cost up to $\$500000$ per unit [3], which a too high investments for small-scale farmers. (Termote)

1.4. Proposed solution

Practically, our task is to program a robot that can pick ripe apples on command, place them in our selling basket, and provide updates on our orchard and harvest. Our goal is to address labor shortages and difficult working conditions by developing an autonomous robot capable of operating for extended periods. Also, the robot needs to be a cost-effective solution to stay within the budget of small-scale farmers. Therefore our solution consists of a modular system with easy replaceable 3D printed parts, and open-source software allowing for low cost maintenance. The MIRTE Master robot uses a SLAM algorithm to map its environment and localize itself within it. Obstacles in the map are identified using a LiDAR sensor, while a camera sensor, combined with object detection models, is used to locate and classify trees, apples, and the selling basket. The robot can pick up several apples at a time and place

it in its carrying basket. The robot navigates to a tree using planning and control algorithms, where it employs an object detection model to distinguish ripe from unripe apples. The coordinates of ripe apples are then used to guide the robot's gripper for picking. Afterward, the robot moves to the basket and gently drops the apples inside the selling basket. Ultrasonic sensors detect nearby moving objects, such as humans and other animals, and integrate this information into the motion planning to ensure safe navigation. Two graphical interfaces are provided: one for the farmer, showing a global map, task progress, error messages, battery status, and start/stop controls; and a more technical interface for developers, featuring depth maps, detailed error reports, full robot state, and sensor visualizations for performance evaluation. (Van der Weij)

1.5. Requirements

In this section, we describe all the requirements that our proposed solution should meet. We divide the requirements into two categories: mandatory and optional. The mandatory requirements are necessary for the robot to operate, while the optional requirements provide additional benefits for the client. (Kritharoulas)

1.5.1. Mandatory requirements

As part of the mandatory requirements, the robot should be able to map its surroundings and localize itself. It must also be capable of detecting trees and potential obstacles. Furthermore, it should navigate towards the tree or basket while avoiding the static obstacles. If it detects a farmer or an animal on its way, it has to slow down and give them priority. Another crucial thing is to be able to distinguish between ripe and unripe apples, move its arm towards a selected apple, and grasp it using the gripper. Finally, it should be able to drop an already grasped apple into the basket. Regarding human-robot interaction, the robot must be able to communicate its location on the map and its sensor data in a visually friendly manner to the farmer. Additionally, frequent updates about the robot's mission status (e.g., whether the mission has been completed successfully) should be communicated to the farmer. Finally, when the robot attempts to grasp an apple, it should be able to detect if a human hand or an animal is nearby and stop its motion to ensure safety. For a list of mandatory requirements, see Table 1.2. (Kritharoulas)

Reference	Short description	Specialization
MR1	Map and localize	Planning, Perception
MR2	Recognize and locate tree and obstacles	Perception
MR3	Navigate towards tree/basket while avoiding obstacles	Planning
MR4	Detect any farmers or animals in the robot's path	Perception, HRI
MR5	Slow down and give priority to farmers/animals if they are detected	Planning,HRI
MR6	Discriminate between ripe and unripe apples	Perception
MR7	Grasp ripe apples	Planning
MR8	Drop apples to the basket	Planning
MR9	Communicate robot's location, sensor data and mission status to the farmer	HRI
MR10	Detect human hand or animal when trying to grasp an apple and stop arm's motion	HRI

Table 1.2: List of mandatory requirements in this project (Kritharoulas)

1.5.2. Optional requirements

The robot can also be designed to navigate towards the tree/basket in a way that minimizes battery usage. For the apple-grasping task, a more intelligent behavior involves the robot aiming to grasp the apple that requires the least amount of effort in terms of joint torque. Another optional requirement is to incorporate a failure recovery mechanism that clears the constructed map and restarts the mapping and localization process when the robot gets stuck. Regarding the human-robot interaction aspect, the

robot can be able to communicate information to the farmer regarding its battery level and the number of ripe apples it has successfully placed in the basket. Additionally, it can detect the farmer's hand gestures indicating whether the robot should stop or approach, or receive start/stop commands from the farmer through a web-interface. For a list of the optional requirements, see Table 1.3 (Kritharoulas).

Reference	Short description	Specialization
OR1	Minimize battery usage while navigating	Planning
OR2	Grasp the ripe apple that minimizes total joint torque	Planning
OR3	Clear map and restart when robot gets stuck	Planning
OR4	Follow start/stop farmer commands through a web-interface	HRI
OR5	Communicate information to the farmer regarding battery level and number of apples in the basket	HRI
OR6	Detect farmer start/stop hand gestures and comply accordingly	Perception,HRI

Table 1.3: List of optional requirements in this project (Kritharoulas)

1.6. Project plan

Name	Begin date	End date	Resources
Phase 1	4/21/25	4/25/25	
Phase 2	4/28/25	5/8/25	
Local Path Planner	4/28/25	5/8/25	
Analytical Robot Model	4/28/25	5/5/25	Mukil
MPC Prototyping	4/28/25	5/7/25	Antreas
Manipulator Controller	5/6/25	5/8/25	
RRT*	4/28/25	4/30/25	Antreas
Dataset Creation	4/28/25	4/30/25	Nils,Tjerk
Interface with all sensors	4/28/25	4/30/25	
Perception Protoyping	5/1/25	5/7/25	Nils,Tjerk
GUI Design	4/29/25	5/7/25	Dionysios
List user requirements	4/29/25	4/29/25	Dionysios
Create React JS front end	4/30/25	5/6/25	Dionysios
Intergrate with ROS2	5/7/25	5/7/25	Dionysios
Reseach on SLAM	4/28/25	5/8/25	Antreas,Mukil,Nils,Tjerk,Dionysios
Report Writing	4/28/25	5/7/25	Antreas,Mukil,Nils,Tjerk,Dionysios
Client Presentation	5/8/25	5/8/25	
Report 2 due	5/8/25	5/8/25	
Buddycheck 1 due	5/9/25	5/9/25	
Phase 3	5/13/25	6/4/25	
Global Path Following	5/13/25	6/4/25	Antreas,Mukil
Waypoint generation	5/13/25	5/16/25	Antreas
Controller for robot base in ROS2	5/13/25	5/15/25	Mukil
Mobile Manipulator Controller	5/19/25	5/23/25	Antreas,Mukil
Dynamic Obstacle Avoidance	5/29/25	6/4/25	Antreas,Mukil
Detect Objects	5/13/25	5/21/25	Nils,Tjerk
Train object detection model	5/13/25	5/16/25	Tjerk
Intergrate Model with ROS2	5/19/25	5/21/25	Tjerk
SLAM	5/22/25	5/29/25	Nils
Sensor Fusion	5/22/25	5/29/25	Nils,Tjerk
User GUI	5/23/25	5/29/25	Antreas,Dionysios
Human / Animal Detection	5/13/25	5/28/25	Dionysios
Dataset Augmentation	5/13/25	5/13/25	Dionysios
Vision Model Training	5/14/25	5/14/25	Dionysios
Integration with ROS2	5/15/25	5/28/25	Dionysios
Navigation with Static Obstacles	5/26/25	5/26/25	
Navigation with Dynamic Obstacles	6/5/25	6/5/25	
Complete Navigation and Mapping	6/5/25	6/5/25	
Report 3 due	5/14/25	5/14/25	
Gitlab Documentation due for review	5/14/25	5/14/25	
Buddycheck 2 Due	5/14/25	5/14/25	
Phase 4	6/5/25	6/17/25	
Prepare Presentation	6/5/25	6/13/25	Antreas,Mukil,Nils,Tjerk,Dionysios
Report Writing	6/5/25	6/17/25	Antreas,Mukil,Nils,Tjerk,Dionysios
Testing and evaluation	6/5/25	6/13/25	Antreas,Mukil,Nils,Tjerk,Dionysios
Documentation Feedback due	6/6/25	6/6/25	
Final Presentation	6/16/25	6/16/25	
Final Report due	6/18/25	6/18/25	
Final GitLab documentation due	6/16/25	6/16/25	
Individual Reflection Report due	6/17/25	6/17/25	

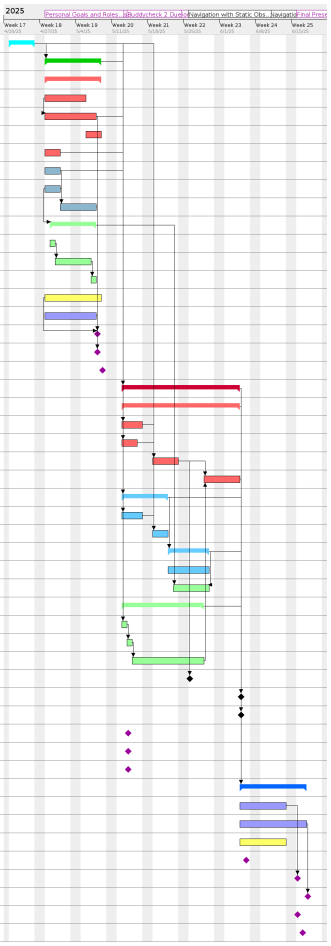


Figure 1.1: Gantt chart of the project (Kourris)

The Gantt chart can be seen in Figure 1.1. The color coding of the Gantt chart is as follows: Yellow means group brainstorming or research, Purple is for report writing and documentation, Orange is for path planning and controls, Blue for perception, Green for human-robot interaction and finally Black represents milestones. (Kourris & Saravanan)

2

Functional Architecture

2.1. Functional hierarchy

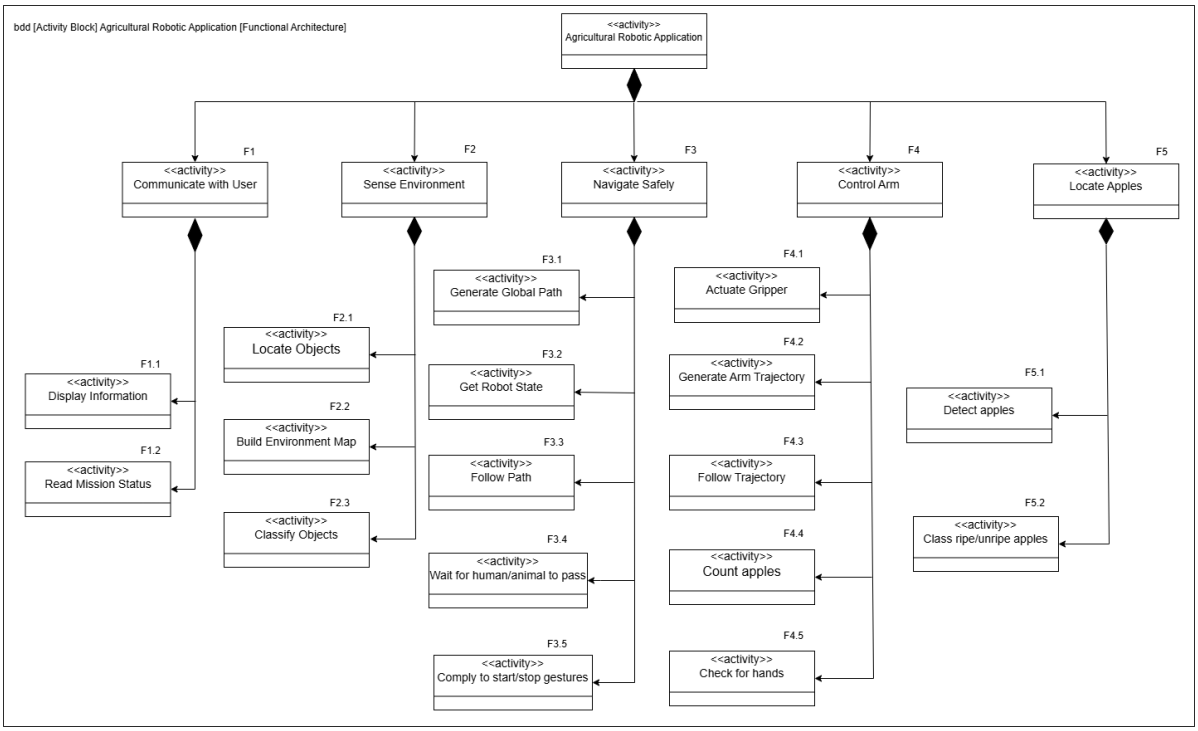


Figure 2.1: Functional hierarchy of the system (Kritharoulas)

The first level of the functional architecture consists of five main functions. F1 includes all the activities required for the robot to communicate with the user, which in this case is the farmer. F2 refers to all tasks related to environment sensing using the available sensors on the robot. F3 encompasses the activities necessary for the robot to navigate safely from a given point A to point B. F4 refers to the control of the arm and finally, F5 includes all the functions that enable the robot to locate apples within the scene. See Figure 2.1 for an overview.

One level lower, there are two subfunctions related to communication with the user. Specifically, F1.1 computes the information on robot location, state and battery percentage that is displayed to the user, while F1.2 refers to how the robot keeps track of the mission status. These two functions both relate to the requirement MR10 as described in Section 1.5. Regarding environment sensing, F2.1 refers to the robot's ability to detect objects in the scene. By 'objects', we mean anything that may be present

in the environment, including trees, baskets, static and dynamic obstacles, as well as humans and animals moving around. This function fulfills requirements MR2 and MR4 as stated in Section 1.5. F2.2 refers to the process of building a map of the environment which the robot can use to localize itself and navigate to its goal and fulfills requirement MR1. F2.3 corresponds to assigning one of the three classes (red apple, green apple or tree) to a detection, which can be assigned to the requirements MR2 and MR6. Regarding safe navigation, five main subfunctions are considered. F3.1 is responsible for generating the global path on the map from start to end, while F3.2 handles extracting the robot's state, including its x and y coordinates and orientation. F3.3 refers to the local planner, which is responsible for following the previously generated global path, while F3.4 is responsible for slowing down the robot and prioritizing potential humans or animals that are detected and may block the robot's movement. Finally, F3.5 refers to the robot's ability to start or stop based on the farmer's hand gestures, giving priority to its intention instead of focusing solely on navigating toward the goal. All the subfunctions of F4 are fulfilling the requirement with reference MR5. For the arm control part, F4.1 is responsible for actuating the gripper, which means either opening or closing the gripper to grasp an apple, and relates to requirement MR7. The function F4.2 generates an arm trajectory from the base to the apple that must be grasped, and F4.3 implements the controller that follows that trajectory. These two functions also fulfill, together with F4.1, requirement MR7 and MR8 of grasping and dropping the apples in the basket. F4.4 is responsible for counting the number of ripe apples still attached to the tree which is associated to requirement MR9 and specifically sharing the mission status. F4.5 is responsible for detecting any human hands near the apples through the gripper's camera. Finally, regarding apple localization, F5.1 is responsible for detecting the apples on the tree, while F5.2 is responsible for classifying them as ripe or unripe. Both of these function refer to requirement MR6. (Kritharoulas & Van der Weij)

2.2. N2 chart of your system

See Table 2.1 for the N2 chart of our system. (Van der Weij)

Table 2.1: N^2 chart of our proposed robotic system (Van der Weij)

• sensor readings	• model detection bbox		• start switch ON			• model detection bbox		• hand detection	• gripper ON				Inputs/Outputs
F2.1 Locate objects	• object location	• object location		• obstacles					• obstacles				
	F2.2 Classify	• class									• class		
		F2.3 Map	• map								• map		
			F3.1 Global path	• path									
		• robot's state		F3.2 Follow path	• detected human/animal	• reached goal			• robot's state		• robot's state		
					F3.4 Wait for human								
						F5.1 Locate apples	• apple location		• apple location				
							F5.2 Classify ripe/unripe		• apple class				
								F4.5 Check for hand	• no hand				
		• robot's state							F4.2 Arm trajectories	• arm trajectory			
										F4.1 Actuate gripper	• apple count		• apple +1
											F1.1 Read status	• GUI data	
												F1.2 Display mission status	

2.3. Functional Flow of your system

See Figure 2.2 for the Functional Flow of our system. Please note that this loop is only for picking up and delivering one apple. (Kourris)

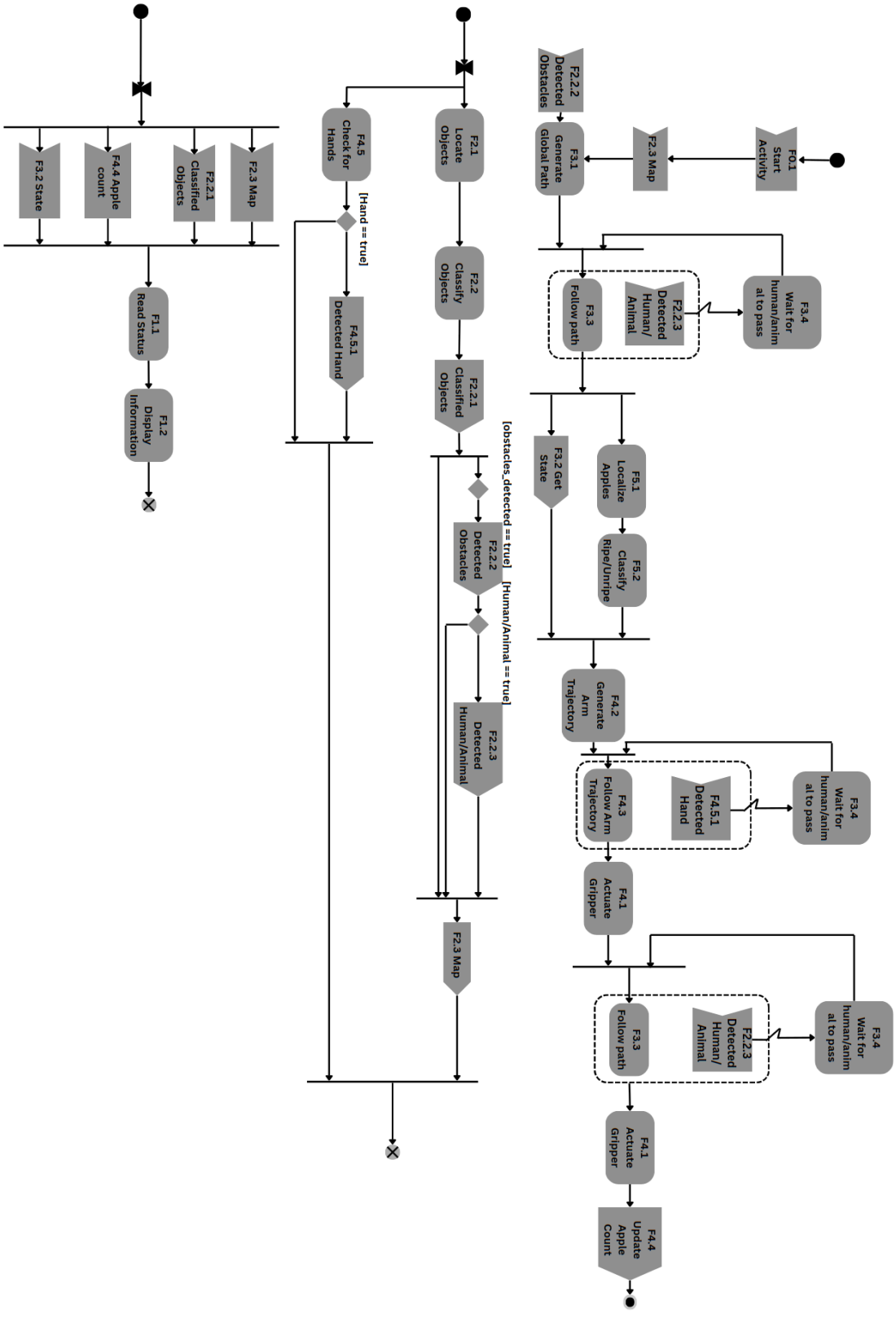


Figure 2.2: Functional Flow Chart for our proposed robotic system (Kourris)

3

Description of the Robot Software

3.1. Nodes Overview

In this chapter an overview of our physical system will be given. Figure 3.1 shows the main data flows and connections between all nodes and Figure 3.2 shows all functions allocated to all nodes. (Termote)

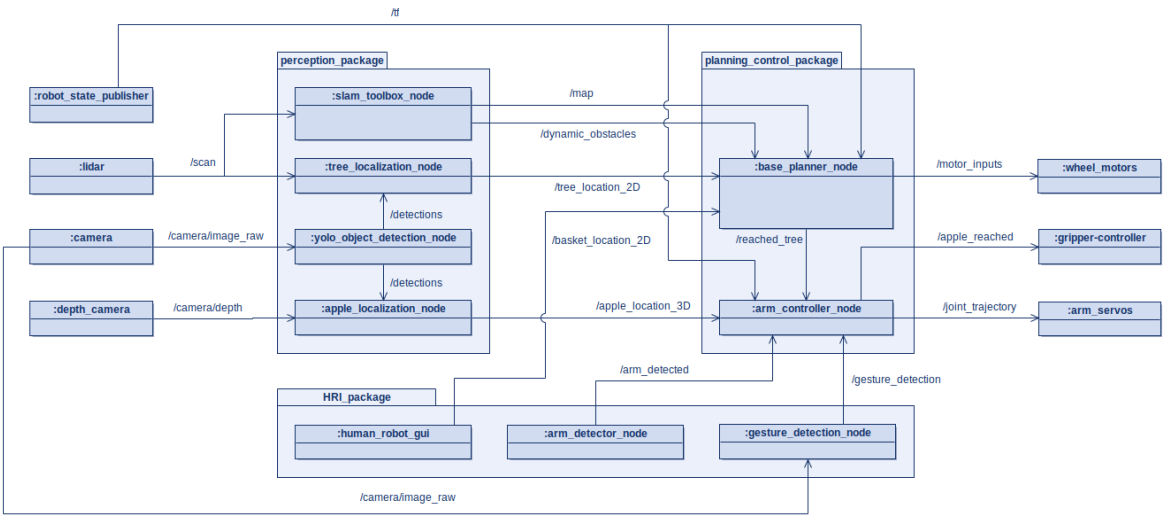


Figure 3.1: SysML Internal Block Diagram of our proposed system (Termote)

3.2. Design choices

An overview of the nodes and how they correspond to the different functions can be seen in Figure 3.3 (Kourris).

3.2.1. F1 Communicate with user

Both sub-functions of this module are allocated to the *human_robot_gui_node*. This node is responsible for reading the robot's state, including position, map, apple count, and sensor data, and presenting this information to the user in real time. Assigning these functions to a dedicated GUI node ensures a clear separation between user interaction and autonomous behaviors, which improves modularity and supports system usability.
Requirements addressed: MR9 (Communicate robot's location, sensor data, and mission status to the farmer) (Kourris & Kritharoulas)

3.2.2. F2 Sense Environment

The *tree_localization_node* identifies and determines the position of trees within the robot's surroundings. It integrates image data from the base camera with LiDAR point cloud information to precisely locate tree trunks in space. This function is handled by a dedicated node to keep tree detection separate from other perception tasks, ensuring cleaner communication interfaces and a modular system design. Accurate tree positioning is crucial for subsequent processes like mapping, planning, and apple detection, making this node a fundamental component of the perception pipeline. The *slam_toolbox_node* is responsible for real-time SLAM, creating and updating a global environmental map that includes static obstacles such as trees and walls. By assigning this task to an independent node, the mapping process remains distinct from other operations like control and object recognition, improving system stability and maintainability. This design allows the map to be updated autonomously as the robot navigates, ensuring a consistent global reference for navigation and localization functions. Together, these nodes enhance the robot's environmental awareness, a key requirement for safe and autonomous operation. Their structured allocation highlights a focus on modular development and functional separation.
Requirements Addressed: MR1 (Map and localize), MR2 (Recognize and locate tree and obstacles) (Kourris & Termote & Saravanan)

3.2.3. F3 Navigate Safely

To ensure safe and autonomous navigation, the *base_planner_node* manages both local and global path planning. This node integrates the necessary planning functions for the robot to determine and follow viable paths through its environment while avoiding static and dynamic obstacles. Centralized planning logic with a single node allows for a clear separation between navigation, perception, and control components. This architectural approach promotes modularity and simplifies future updates to planning algorithms. The global planner computes a collision-free route from the robot's current position to its target destination, factoring in static elements from the mapped environment. Meanwhile, the local planner enhances real-time adaptability by continuously adjusting the path in response to moving obstacles and environmental changes. By integrating both planning layers within the same node, the system optimizes their coordination, improving responsiveness and reliability during operation. This structured design supports the robot's need for safe, efficient, and adaptable navigation in dynamic environments.
Requirements Addressed: MR3 (Navigate towards tree/basket while avoiding obstacles), MR5 (Slow-down and give priority to farmers/animals if they are detected) (Kourris & Saravanan)

3.2.4. F4 Control Arm

Once the robot's base reaches its target location, it must interact with its environment to carry out the task of apple picking. The *arm_controller_node* is responsible for this functionality, generating and executing motion trajectories for the robotic arm, actuating the gripper, and tracking the number of apples successfully picked. Target positions are received from the navigation or perception systems, and the node calculates and executes the necessary movements for grasping. This node integrates all manipulation-related functions into a dedicated component, ensuring modular control independent from navigation and perception logic. Additionally, the node handles state updates, such

as apple count, which are shared with the GUI to keep the user informed about task progress. The *hand_gesture_detection_node* detects if a person's hand is also reaching the same apple. If a hand is detected, the robot stops, allowing the person to safely interact within the robot's environment.

Centralizing all manipulation tasks within the *arm_controller_node* simplifies coordination across planning, motion execution, and feedback monitoring, while ensuring consistent and predictable computational load.
Requirements Addressed: MR7 (Grasp ripe apples) MR8 (Drop apples to the basket), MR10 (Detect human hand or animal when trying to grasp an apple and stop the arm's motion) (Kourris)

3.2.5. F5 Locate Apples

The detection and localization of apples are managed by two specialized nodes: *apple_localization_node* and *gripper_camera_node*. The *apple_localization_node* identifies ripe and unripe apples utilizing RGB images and determines their three-dimensional positions by integrating image-space detections with depth data from the primary camera. This process enables the robot to accurately select apples for grasping. Upon approaching a designated apple, control is transferred to the *gripper_camera_node*, which processes images from a monocular camera attached near the gripper. This refinement ensures precise localization, facilitating accurate end-effector positioning during the final grasping phase.

By employing both a wide-field depth camera for comprehensive environmental awareness and a close-range monocular camera for fine-grained localization, the system achieves an optimal balance between global detection and precision. The division of responsibilities across distinct nodes promotes modularity and simplifies the design, configuration, and optimization of each detection stage.
Requirements Addressed: MR6 (Discriminate between ripe and unripe apples) (Kourris)

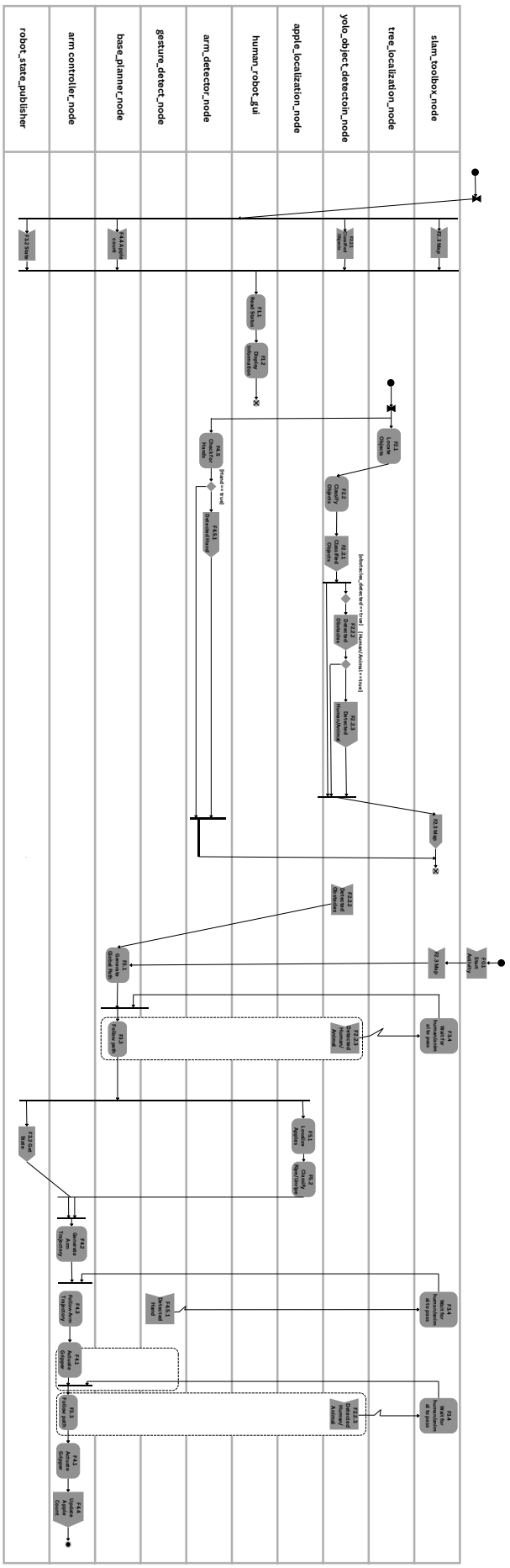


Figure 3.2: Functional Flow Chart of our proposed system with node swim lanes (Termote)

Functions		Nodes									
Id	Function	slam_tool_box_node	tree_localization_node	yolo_object_detection_node	apple_localization_node	human_robot_gui	arm_detector_node	gesture_detection_node	base_planner_node	arm_controller_node
F1	**Communicate with the user**					X				
F1.1	Display Information					X				
F1.2	Read Mission Status					X				
F2	**Sense Environment**	X	X	X	X		X			
F2.1	Locate Objects		X							
F2.2	Build Environment Map	X								
F2.3	Classify Objects			X						
F3	**Navigate Safely**	X								
F3.1	Generate Global Path								X	
F3.2	Get Robot State									
F3.3	Follow Path								X	
F3.4	Wait for human/animal to pass	X					X		X	
F4	**Control Arm**							X		
F4.1	Actuate Gripper									X
F4.2	Generate Arm Trajectory									X
F4.3	Follow Trajectory									X
F4.4	Count Apples									X
F4.5	Check For Hand							X		
F5	**Locate Apples**			X	X					X
F5.1	Detect Apples			X	X					
F5.2	Classify Ripe/Unripe Apples			X	X					

Figure 3.3: Table of functions and corresponding nodes (Kourris)

4

Validation of the Robotic Solution

4.1. Test procedures

In Table 4.1 the tests performed in this project can be found (Kourris & Saravanan).

Reference	Short description	Function
T1	Build a map of the environment	F2.2
T2	Classify trees, ripe and unripe apples	F2.3, F5.2
T3	Estimate position of objects in the environment	F2.1
T4	Move robot base to a specific location on the map	F3.1, F3.2, F3.3
T5	Avoid dynamic and static obstacles	F3.4
T6	Move end effector to specific position and orientation	F4.2, F4.3
T7	Stop the manipulator from colliding with a human	F4.5
T8	Grab apples and place them in the basket of the robot	F4.1
T9	Detect farmer's start/stop hand gestures and comply accordingly	F3.5

Table 4.1: List of conducted tests in this project (Kourris & Saravanan)

4.1.1. Test T1: Build a map of the environment

We tested the F2.2 function. Maps were created both autonomously and using tele-operation to control the robot. Mapping was tested both in the demo arena and in larger areas. The aim was to see if the mapping algorithm could correctly capture all the static obstacles. The maps were accurate, and in autonomous mode, the robot did not collide with any objects. Autonomous exploration was also tested in different areas. The aim was to observe whether the robot could map the closed environment in finite time. (Kourris & Saravanan)

4.1.2. Test T2: Classify trees, ripe and unripe apples

We tested functions F2.3 and F5.2. The object detection model was initially tested on a subset of the dataset that was not used for training, and then on the actual cameras. These tests were performed under varying distances and lighting conditions. The aim was to see how accurately and robustly the model could classify the objects. Multiple numerical evaluations (like bounding box regression loss, classification loss and mAP50 score) were performed to check the efficiency of classifications. Visual model evaluation can be found in Appendix A. (Kourris & Saravanan & Van der Weij)

4.1.3. Test T3: Estimate position of objects in the environment

We tested the function F2.1. The tree localization node was tested at different robot positions and orientations such that the tree is in the field of the camera. Within the detected tree, 3D positions of a ripe apple was cross-checked manually with a measuring tape. The aim was to evaluate whether the

localization algorithm could locate the tree and the red apple precisely. (Saravanan)

4.1.4. Test T4: Move robot base to a specific location on the map

We tested functions F3.1, F3.2 and F3.3. The map server node was tested to check whether the mapped environment (Occupancy Grid Map) was loaded properly in RViz. Once the map is loaded, the Adaptive Monte Carlo Localization (AMCL) node checks whether the robot can localize itself within the map. It does this by observing the introduction of local costmaps, global costmaps and AMCL particles in RViz. Sometimes, the initial location of the robot with respect to the map frame was manually provided. When a 2D navigation goal is provided by the user, the global and local paths generated by the planner are evaluated for path completeness. Eventually, the location of the robot is checked with the provided goal location for minimal localization error. The robot could reach the goal position within a reasonable error bound. (Saravanan)

4.1.5. Test T5: Avoid dynamic and static obstacles

We tested the function F3.4. Once the 2D navigation goal is provided either manually or computed autonomously from the detected tree location, the following was observed. At first, the global and local costmaps were observed such that the robot does not go inside inflated obstacle configurations. It is to be noted that the global costmap is used for avoiding static obstacles that are already mapped, while the local costmap is used for avoiding dynamic obstacles in the environment. This is done by checking whether the current sensor readings are reflected in the local costmap as the robot moves in the environment. Subsequently, global and local paths were evaluated for non-collision with static and dynamic obstacles, respectively. Experimenting with different environments (for static obstacles) and slow-walking humans (for dynamic obstacles) resulted in collision-free navigation. Specifically for dynamic obstacles the robot had to navigate back forth from the starting position to the tree 5 times with a human in the arena and 5 times with another robot. (Saravanan & Kourris)

4.1.6. Test T6: Move end effector to specific position and orientation

We tested functions F4.2 and F4.3. A 3D position and orientation were given to the manipulator. We measured the deviation of the end effector's final position and orientation from the preset one. To accurately evaluate the manipulator, this test was performed ten times. (Kourris)

4.1.7. Test T7: Stop the manipulator from colliding with a human

We tested the function F4.5. A 3D location of an apple is computed and set as the goal position for the manipulator. To test this function, a human arm is used to block the robot's path. The purpose of the test is to evaluate whether the robot can stop by setting its velocity to zero upon detecting the obstruction, and then resume its movement once the obstruction is cleared. Additionally, the performance under different lighting conditions was tested, and the position accuracy of keypoints of a slow-moving human arm was evaluated. The human arm detection pipeline was evaluated for detection speed and accuracy. (Saravanan & Kourris)

4.1.8. Test T8: Grab apples and place them in the basket of the robot

We tested the function F4.1. This test was meant to test the ability of the gripper to grab and hold on to apples. The gripper was given a known apple location and had to navigate to it and grab the apple. (Kourris & Kritharoulas)

4.1.9. Test T9: Detect farmer's start/stop hand gestures and comply accordingly

We tested F3.5 to verify the robot's response to start/stop hand gestures. In a static environment, the robot remained still until it detected a thumbs-up gesture through its gripper camera, then successfully moved toward its goal. Subsequently, upon detecting an open-palm (stop) gesture, it halted as expected, confirming proper functionality. (Kritharoulas)

4.2. Validation results

4.2.1. Test T1: Build a map of the environment

When tested T1, the robot met all the requirements and was able to map the closed environment without deviation. It is to be noted that the robot sometimes deviates from the true representation for a short amount of time, especially when the environment has too few features (mostly empty; without any obstacles). However, loop closure effectively corrected these errors upon revisiting known areas, ensuring the final map was accurate and consistent. (Saravanan)

4.2.2. Test T2: Classify trees, ripe and unripe apples

When evaluating T2, after 50 epochs the algorithm achieved a bounding box regression loss of 0.44, which means the model has reasonable accuracy in assigning bounding boxes. A classification loss of 0,24 on the validation set shows that there is high accuracy in labeling the three different classes. Furthermore the Mean Average Precision at IoU threshold of 0.5 reached 0.99, meaning detections were almost always correct and complete. Thus, test T2 was declared as passed successfully. However, more inference test can be conducted in varied lighting conditions. (Saravanan & Van der Weij)

4.2.3. Test T3: Estimate position of objects in the environment

In Test T3, the robot successfully detected the tree and accurately identified the target objects. Object classification and localization were consistent throughout the test. However, a small but consistent bias was observed in the distance measurements to the detected objects. (Saravanan)

4.2.4. Test T4 & T5: Navigate robot base while avoiding obstacles

During tests T4 and T5, all requirements were met. The algorithm successfully loaded the map and computed the robot's initial location relative to the map frame. While autonomous localization was initially imprecise, manually providing an approximate start point improved performance. The robot refined its position as it moved and received new sensor data. Path planning and obstacle avoidance worked consistently, allowing the robot to reach its target. Minor issues included occasional sensor delays due to network congestion and difficulty navigating narrow paths caused by obstacle inflation. Overall, both tests were successful. (Saravanan & Kourris)

4.2.5. Test T6: Move end effector to specific position and orientation

While conducting test T6, small deviations were measured for both position and orientation. Specifically, there was a position error of $\pm$ 2.5cm and for the orientation an error of $\pm$ 5 degrees. These errors were considered small enough that a low-level controller could compensate for them. Even though the manipulator was not totally accurate, the results were considered acceptable. (Kourris)

4.2.6. Test T7: Stop the manipulator from colliding with a human

Test T7 passed successfully. It was tested under different lighting conditions and varying positions of human intervention in the camera's field of view. We observed that the keypoints of the hand was accurately detected in real-time with occasional reduction in total keypoints. However, at all times, detected keypoints were sufficient to trigger a command to the manipulator to pause the motion. When the human arm went out of the camera's field, the manipulator could continue its path without any jerk and other suboptimal behavior. (Saravanan)

4.2.7. Test T8: Grab apples and place them in the basket of the robot

Test T8 was considered inconclusive. While the gripper was able to navigate to the right location due to a hardware malfunction, the gripper did not function, and as such, the results have been deemed inconclusive. (Kourris & Kritharoulas)

4.2.8. Test T9: Detect farmer's start/stop hand gestures and comply accordingly

Test T9 was considered successful. It was performed with various positions of human gestures within the camera's field of view. However, recognition accuracy decreased with distance, and the camera image was rotated by 90º due to the way it was mounted on the gripper. (Kritharoulas)

4.3. Software changes

4.3.1. Test T1: Build a map of the environment

When the robot observes a dynamic obstacle during exploration, the map tends to have artifacts, such as single-pixel false-positive obstacles. These were mitigated by tuning the mapping parameters pertaining to *slam_toolbox_node*. The test T1 was later updated to eliminate such artifacts during mapping. (Saravanan)

4.3.2. Test T2: Classify trees, ripe and unripe apples

Data augmentation techniques were applied to enhance the robustness of the detection system. These included adjustments to lighting conditions using HSV value modifications and saturation adjustments of the apples to improve real-life detection of high or low-saturated red apples. This increased the detection mAP50 score after 50 epochs from 0.62 to 0.99. (Saravanan & Van der Weij)

4.3.3. Test T3: Estimate position of objects in the environment

This bias was traced to minor distortions in the gripper-mounted camera. It was effectively mitigated by recalibrating the camera and applying the obtained lens distortion parameters to rectify the image. After rectification, the distance was estimated using the intrinsic calibration matrix. The recalibrated camera showed significantly improved results, enhancing overall object localization accuracy. The inflation radius of the obstacles was reduced to make passing through tight corridors easier. (Saravanan & Kourris)

4.3.4. Test T4: Move robot base to a specific location on the map

To improve the planner, the parameters of the MPPI algorithm were tuned and tested in the real world. This resulted in a smoother global path and paths that were not very close to the inflated obstacles. Robot localization was also improved by tuning parameters related to the AMCL node, including sensor update frequency, particle size, and so forth. (Saravanan)

4.3.5. Test T5: Avoid dynamic and static obstacles

For Test T5, the configurations of the MPPI controller were modified, specifically the constraint, cost, and path alignment critics. These adjustments led to improved controller performance across the evaluation parameters defined in Test T5.(Saravanan)

4.3.6. Test T6: Move end effector to specific position and orientation

From the results of test T6 it was clear that a low-level controller for fine-tuning the gripper position needed to be developed; unfortunately, due to time constraints, that was not implemented. The velocity of the manipulator was, however, reduced in order to allow the gripper to reach the end position more accurately. (Kourris)

4.3.7. Test T7: Stop manipulator from colliding with a human

Since all the requirements were achieved successfully. No software change was required in the module. We hypothesize this due to its robust and accurate detection of the human arm in real time. (Saravanan)

4.3.8. Test T8: Grab apples and place them in the basket of the robot

Because of the hardware malfunction, the entire apple-picking strategy had to be reinvented. Using the compliance of the gripper, it was made to pick up apples by driving into them with the gripper and dropping them into the basket using friction. While the apple-picking success rate was very small, using friction to drop the apples into the basket was more reliable, but also very slow. Because this solution came at the last minute due to necessity, the team has considered it an acceptable comprise in the absence of a functioning gripper. (Kourris & Kritharoulas)

4.3.9. Test T9: Detect farmer's start/stop hand gestures and comply accordingly

From T9, we found we needed to adjust the gesture detection's confidence parameters for reliable detection at greater distances, and add a preprocess step to correct the camera's 90º rotation. (Kritharoulas)

5

Project Conclusions

In this chapter, we will present our final robotic solution and show our robot's potential. We will also discuss how the robot can be deployed and what the limits of our solution are. (Termote)

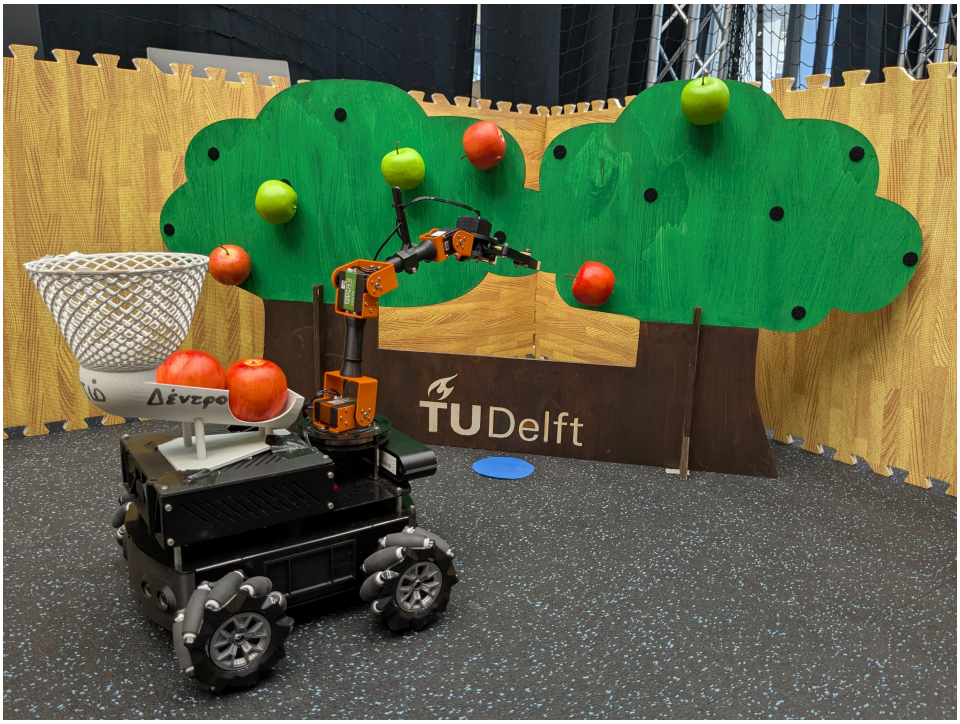


Figure 5.1: MIRTE Master robot in arena with custom basket (Van der Weij)

5.1. Concluding remarks

After an intense quarter of research and development, we are proud to finally present our solution. Currently, our solution is able to autonomously create a 2D map of its environment. It can find orchards in its view and obtain their location on the map. Our robot is able to navigate towards a given location, which can be the basket or the tree. Additionally, our solution can find green and red apples in its view and find the 3D position of red apples. Our gripper can autonomously move towards the red apples with help of its gripper camera and a custom storage unit (see Appendix C) on the back of the robot was designed and fitted which enabling efficient storage and retrieval of apples during runtime. For keeping humans safe while the robot operates, a functionality is present in which the robot arm will stop when

it sees a hand in front of the camera. Lastly, our robot can stop or continue driving based on human hand signals spotted on its camera. The potential of our robotic solution lies in integrating all these functions successfully. Our solution will then be highly autonomous and only limited manual control is needed. Also, a detailed GUI can be created which would give the user real-time information during the process of collecting apples. Small-scale apple farmers should be interested in our solution because of the potential of autonomous apple picking. Also we devised an original, robust apple storage/retrieval which can be seen in Figure 5.1. (Termote)

5.2. Deployment steps

To deploy our solution, the user should first place the robot in or near the orchard with a charged battery. The operator should be present to communicate with the robot. Then, start the autonomous exploration phase. Once the tree has been located, use this location to send the robot to the tree. The 3D apple locations will have been found autonomously. Next, start the autonomous arm-to-apple functionality. Once the apple has been gripped, the user should activate the gripper to move back to the apple carrier attached to the robot. Then, once a few apples have been grabbed, the user should send the robot to the basket to deliver the apples. This process can be repeated until there are no more apples or the battery runs out of charge. (Termote)

5.3. Operational design domain

First of all, the robot should be put in an environment with landmarks like trees or walls so it can find its locations on a map- a flat plane will not suffice. The terrain should be even and not slippery. The weather must be clear as to not block the LiDAR or cameras because without these sensors, the robot will not function correctly. The robot should have access to a network with a sufficiently large bandwidth so that data can be streamed from the robot to the user. Only red apples can be obtained, and only trees with green leaves and brown trunks can be recognized. (Termote)

5.4. Known issues

Unfortunately some issues could not be resolved, and optional requirements could not be realized within the development window of two months. There were global issues and issues as follows. (Termote)

Global

In general, there were a number of issues that could not be resolved or kept occurring. To start, the network in the testing and developing facility (CoR TU Delft) was often overloaded by other teams which slowed down progress and lowered the maximum network load by our robot. Also, some servos in the arm kept breaking, which resulted in us not being able to use the gripper during the demonstration day. However, a fault-tolerant solution was designed to make the gripper compliant with elastic bands and planned a path such that the un-actuated (broken) gripper grabs the localized apple. The dropping maneuver was performed such that the grabbed apple is slid along the basket, creating tangential force to eventually drop it. There was also not enough time to fully integrate all standalone functionalities into a single autonomous machine, and we had to rely on the user activating automated parts. Additionally, no finite state machine was designed for the robot. Having lower priority to the functioning of the robot, there was no time for an expanded Graphical User Interface (GUI). Currently unfortunately no GUI status messages are sent to the user. (Termote & Saravanan)

Perception

There were also some issues specific to the perception team. Firstly, there was no time to write logic for prioritizing certain apples in the object detection model. Also, the tree recognition node would also recognize half trees, which causes issues with the localization which were not solved. (Termote)

Planning

Some issues in planning remained active due to time constraints, like creating a low-level controller for fine-tuning the position of the gripper and base. Additionally, no node for emergency stopping was developed. (Termote)

Postface

Congratulations to finishing your project! As the teaching staff of the Multidisciplinary Project, we want to take the moment to thank you for joining the course and extend our appreciation for your dedication, creativity, and hard work throughout the entire duration of this 8 week project. We hope that your perseverance and collaborative spirit have truly paid off, resulting in the successful completion of the project and demonstration of your robot. As you reflect on this milestone, remember the valuable lessons you've learned, the team-spirit you've forged, and the memories you've created together. May this experience fuel your curiosity, drive, and ambition as you continue to pursue your passions throughout your Master studies at TU Delft and your future endeavors. We cannot wait to see where your journey will take you next!

Your MDP 2025 Teaching Team
November 19, 2025

A

YOLOV8n model evaluations

Here you can find visualizations of the evaluation of our YOLOv8n model after fine-tuning on the custom created dataset. In Figure A.1, model predictions on the input batch are shown. In Figure A.2 and A.3 the F1 curve and P curve of the model can be seen, respectively.

Figure A.1: Prediction of model for first batch to be processed. Confidence scores are shown next to class labels.

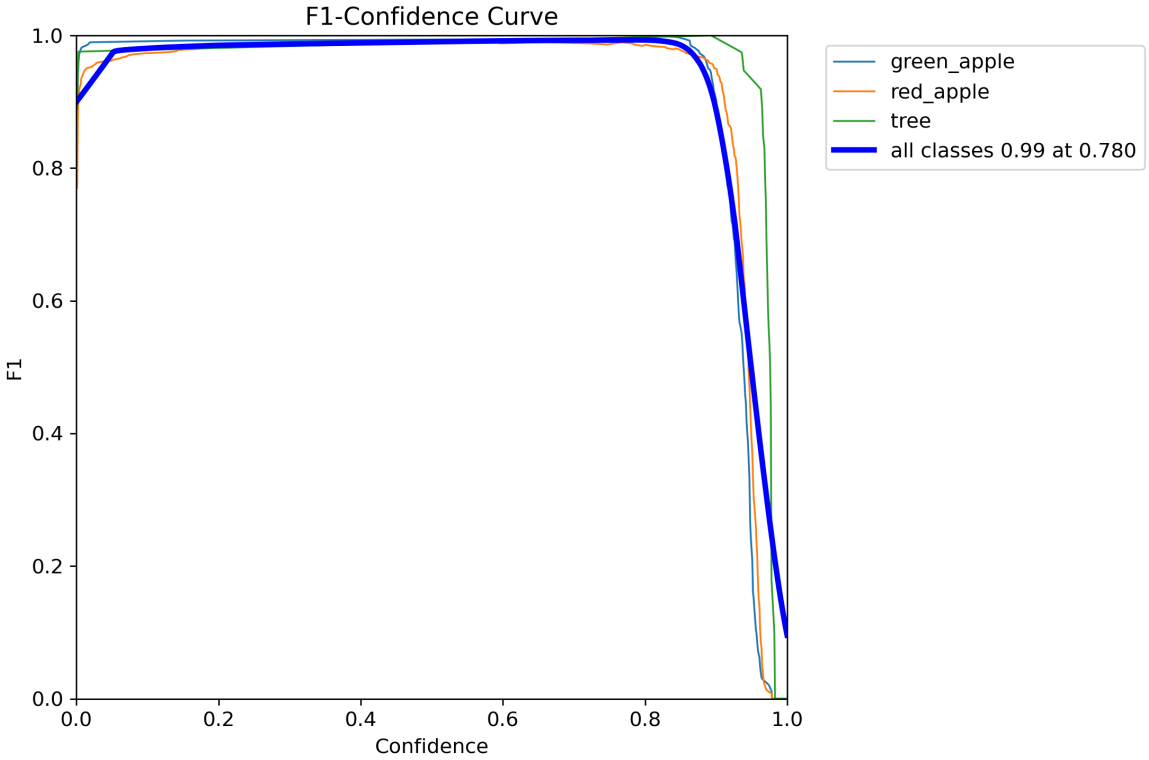


Figure A.2: F1 Curve of our model

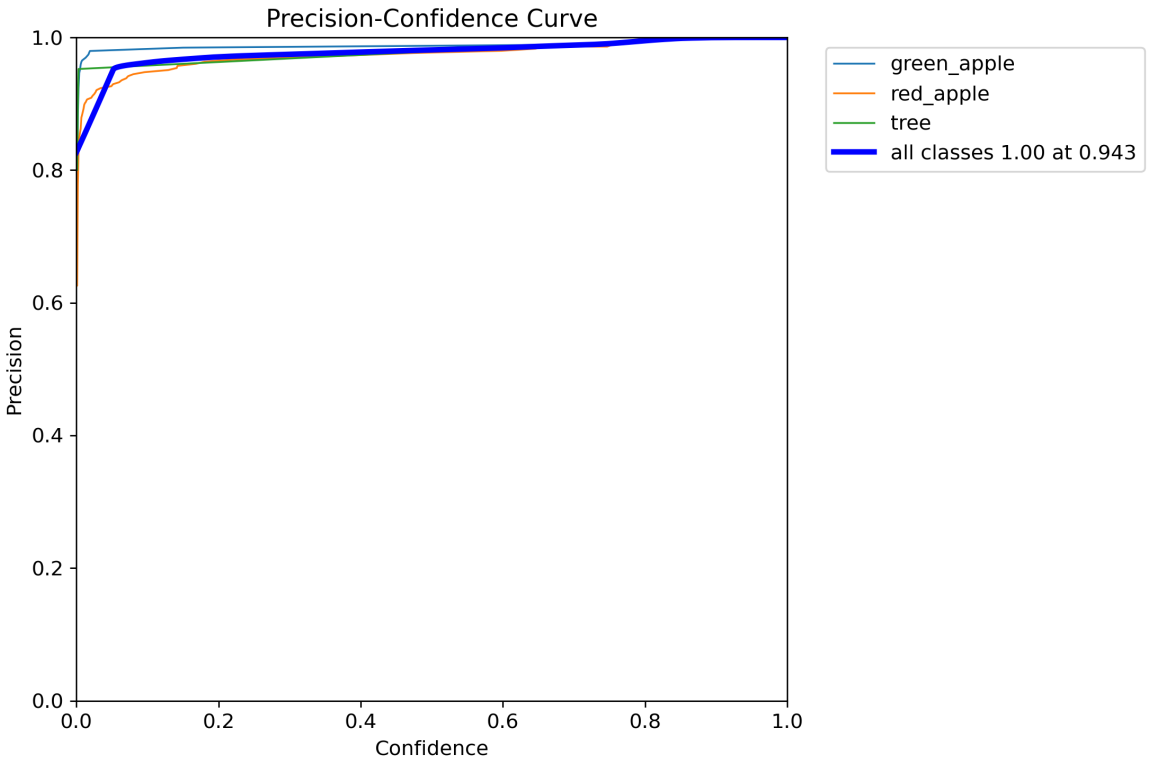


Figure A.3: P curve of our model

B

Navigation Module

Figure B.1 depicts the global costmap depicting free space of CoR department (white), obstacles (dark purple), and inflation layers (light blue) for robot navigation, along with a planned path (red) and sensor readings (scattered white/pink dots). (Saravanan)

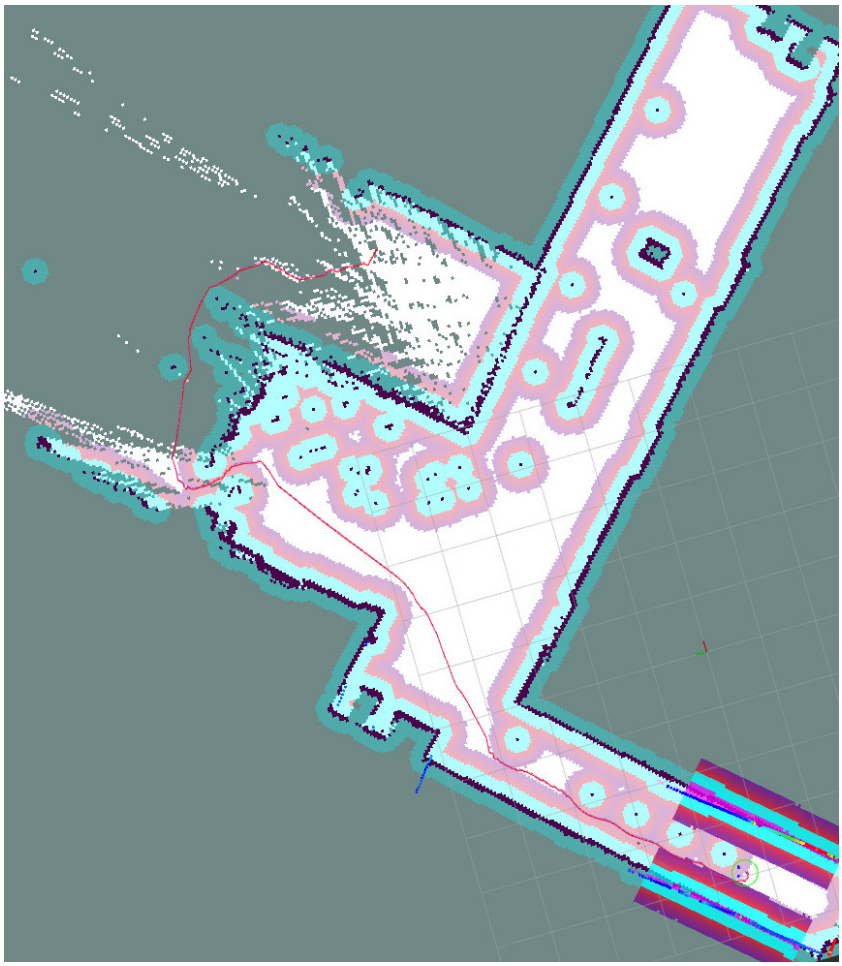


Figure B.1: Figure depicting visualization of robot navigation in RViz2 (Kourris)

C

Custom made basket

Here you can find the CAD files of the custom printed basket and roll support created by the team. These CAD files were 3D printed and attached to the MIRTE Master base of our robot to efficiently collect and pick apples. See Figure C.1 and C.2 for the designs. (Van der Weij)

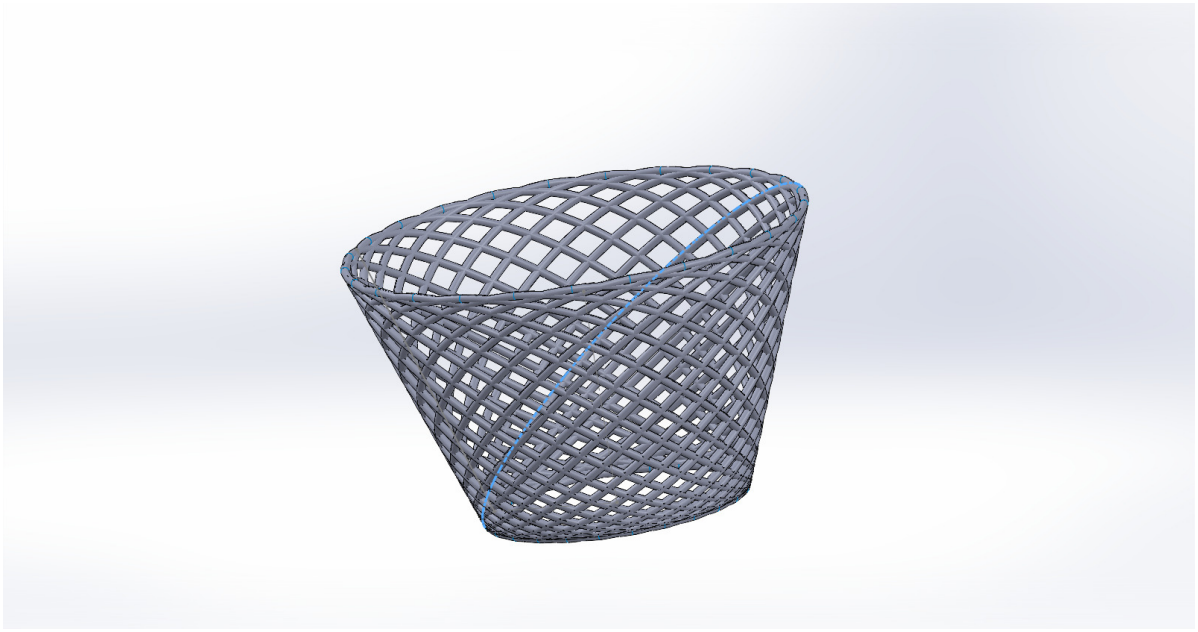


Figure C.1: Custom designed basket (Van der Weij)

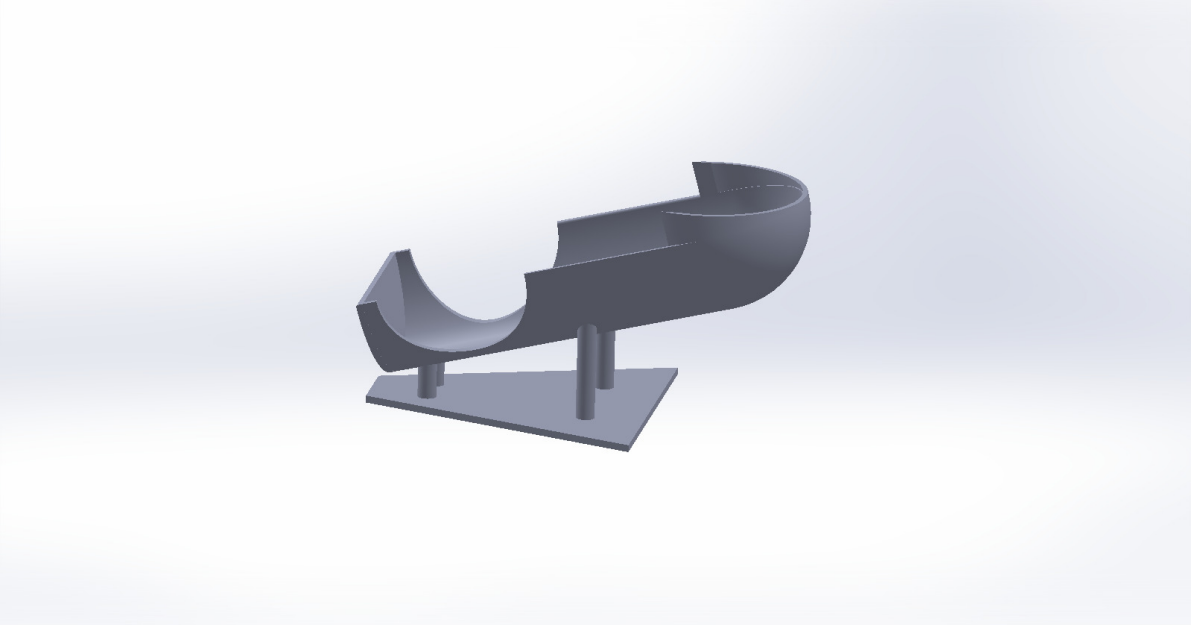


Figure C.2: Custom designed roll support for our robot (Van der Weij)

Bibliography

[1] European Commission. *Sustainable Development Goals*. https://international-partnerships.ec.europa.eu/policies/sustainable-development-goals_en. International Partnerships. n.d. URL: https://international-partnerships.ec.europa.eu/policies/sustainable-development-goals_en.

[2] Scott Fulmer et al. “Ergonomic exposures in apple harvesting: Preliminary observations”. In: *American Journal of Industrial Medicine* Suppl 2 (2002), pp. 3–9. DOI: 10.1002/ajim.10087.

[3] growAG. *Artificial Intelligence Opportunity Ripe for the Picking*. https://www.growag.com/highlights/article/artificial-intelligence-opportunity-ripe-for-the-picking. n.d. URL: https://www.growag.com/highlights/article/artificial-intelligence-opportunity-ripe-for-the-picking.

[4] G. D. Jukema and R. W. van der Meer. “Arbeidskosten in de akkerbouw en glastuinbouw”. In: *Agri-monitor* 2009.feb (Feb. 2009), pp. 11–12.