RO47007 - Multidisciplinary Project

Group 19

Marco de Böck (4494237) Jesse Dolfin (5560047) Guido Dumont (5655366)
Jurjen Scharringa (4708652) Jakob Bichler (5856795)

June 19, 2023

[21]

TU Delft Delft University of Technology

TNO innovation for life

Abstract

This document serves as a design report for the Multidisciplinary Project (RO47007) conducted at TU Delft, in collaboration with TNO. The objective of this project is to address the growing strain on the healthcare system caused by the rising aging population, which places an increasing workload on healthcare professionals. To mitigate this issue, our project focuses on utilizing the robot dog 'Spot' from Boston Dynamics to detect, pick up, and deliver items requested by healthcare personnel, thereby reducing their workload.

Extensive analysis has been conducted to come up with the requirements and architecture of the software, as described in Chapter 3. The software developed for this project is designed using ROS 1, which aligns with Spot's operating system. Eight nodes have been developed to facilitate different functionalities, including "SLAM," "Detection," "RRT Path Planning," "Exploration," "Motion Controller," "SPOT Control Panel," "Conversation node," and the "State machine" node. A comprehensive description of the functions performed by these nodes can be found in Chapter 4.

To overcome the limitation of having only one physical robot available, a simulation environment was provided for testing the nodes. However, due to discrepancies between the simulation environment and the actual robot, the nodes had to be aligned and fine-tuned for effective functioning during testing sessions with the real robot. Detailed results of this alignment process are presented in Chapter 5. Our implementation can be found on our GitLab [1].

This project successfully demonstrates the implementation and integration of the Spot robot, along with its associated systems, to address the identified use case. The group was honoured by receiving the first prize from the client TNO for this project, see figure 2. The links to our demo video [2] and full simulation run [3] can be found in the footnote of this page.

Figure 2: Prize ceremony SpotOn-Care

[1] GitLab: `https://gitlab.tudelft.nl/cor/ro47007/2023/team-19/champ_spot`
[2] Demonstration video: `https://www.youtube.com/watch?v=4hbgdUF2sGU`
[3] Full simulation run: `https://www.youtube.com/watch?v=niYJE1J9ac8`

Contents

1 Meet the Team

SpotOn-Care

Guido Dumont
Machine Perception

Jesse Dolfin
Machine Perception

Jakob Bichler
Human Robot Interaction

Marco de Böck
Planning/Navigation

Jurjen Scharringa
Planning/Navigation

Team logo

Our team consists of five first years Robotics Master students and is named *SpotOn-Care*. Each group member has a specialized role within the team: Planning and Decision Making, Human-Robot-Interaction, or Machine Perception. In the section, each group member will describe themselves, their goals, and their role within the team.

1.1 Team members

Guido Dumont is one of the Machine Perception specialists within the team. He is disciplined, motivated and tends to take on a leading role within a team. Furthermore, Guido will help others or take an extra step to improve the final results or performance of the project/team. His personal goal is two-sided. From the technical side, he would like to test his knowledge and experience what it takes to create a (working) robotic application in real life. From the social side, he would like to improve his social and communication skills within a group where most of the members were initially strangers.

Jurjen Scharringa is one of the two Planning/Navigation specialists. With his high interest in the previous courses Planning & Decision Making and Deep Learning, he would like to add this acquired knowledge to the project. During this project, his goal is to improve his programming skills, learn how to develop a real-world product, and become a better co-worker in a development team. Jurjen also enjoys working in a team and is a very social person. This is confirmed by the Belbin test where his highest outcomes were connector and coach.

Jesse Dolfin will fulfill the role of one of the Machine Perception specialists. He is interested in automation in general and has plenty of programming experience which can be applied in this role during the project. As a personal goal, he would like to better understand the dynamics within a team and learn how to effectively operate in such a team. He is a very analytical person and does not like to rush into design choices without first fully understanding all possibilities, this is also confirmed by the Belbin test where he scored highest on judge-appraiser.

Jakob Bichler is responsible for Human-Robot-Interaction in the project. His strengths are organization, communication within and outside of the group, and a general understanding of technical systems. This is reflected by the Belbin test scores, which suggest coordinator and specialist as team roles. In this project, he not only wants to deliver on his part but also gain a good understanding of his group members' tasks to understand the necessary steps to create a working application in robotics. He hopes this will clarify which field of robotics he is most interested in. From a social side, he looks forward to group work and hopes to contribute to a productive but also pleasant work experience for his teammates.

Marco de Böck will be one of the two members responsible for the position of Planning/Navigation specialist. He is interested in automation, planning, and perception, particularly in the automotive industry. His personal goal is to enhance his programming skills and become more decisive in making choices to improve the efficiency of the project. He is a team player and desires a positive and honest work environment where team members can speak honestly and point out each other's mistakes without a problem. He sometimes tends to be a perfectionist and focuses on minor details, which can slow down the process. He aims to overcome this tendency. This is also shown in the results of his Belbin test which indicate that he is a Connector and Finalizer. Overall, he looks forward to working on the project and achieving excellent results with a positive team dynamic.

1.2 Organization

The group expects to get practical experience in robotics and apply the knowledge they acquired in previous courses to a real-world project. They also expect to improve their interpersonal skills. The overall goal of the team is to achieve a grade of at least an 8.
Besides the hours that are already scheduled by the course. The group plans to meet every Tuesday from 1:30-5:30 p.m., and every Thursday from 1:30-3:30 p.m. They will talk about the process as it stands, decide on the next course of action, and assign tasks during this time. If there is still time left these tasks will be worked on, otherwise they will be finished outside of these hours. This ensures that the work is finished and ready to be discussed at the next meeting. The group will schedule additional sessions if it becomes evident during the project that these hours are insufficient.
Next to the assigned team roles, the team has decided to pick up an additional role, this is the role of team leader. This person will be responsible that each task is properly done and will lead the group meetings. Besides the team leader, there are also group members responsible for supervising and managing the planning, global system engineering part of the project, and control manager. Other than appointing one permanent role to each group member, the group has chosen to rotate roles each week. This allows each person to get the opportunity to improve a different skill set, see table 1.

	Roles			
	Team leader	*Planning*	*System enginieering*	*Quality*
Week 1	Marco de Böck	Jesse Dolfin	Dumont Guido	Jurjen Scharringa
Week 2	Jakob Bichler	Marco de Böck	Jesse Dolfin	Dumont Guido
Week 3	Jurjen Scharringa	Jakob Bichler	Marco de Böck	Jesse Dolfin
Week 4	Dumont Guido	Jurjen Scharringa	Jakob Bichler	Marco de Böck
Week 5	Jesse Dolfin	Dumont Guido	Jurjen Scharringa	Jakob Bichler
Week 6	Marco de Böck	Jesse Dolfin	Dumont Guido	Jurjen Scharringa
Week 7	Jakob Bichler	Marco de Böck	Jesse Dolfin	Dumont Guido
Week 8	Jurjen Scharringa	Jakob Bichler	Marco de Böck	Jesse Dolfin

Table 1: Rotated roles

1.2.1 Conflict handling

For conflict resolution the team is going to follow six steps that can be worked through to effectively tackle conflicts and resolve them while trying to be as objective as possible:

1. Identify and Define the Conflict.
2. Understand Perspectives.
3. Generate Possible Solutions.
4. Evaluate and Select Solutions.
5. Implement and Monitor.
6. Reflect and Learn.

To give effective feedback the team has decided to use the I-I-You feedback model [10]. This model creates an environment to give and receive feedback in an objective manner which is essential for effective team operation.

2 Project Goal

TNO is an independent research organization composed of units, councils, and a services organization [30]. TNO came to the Technical University of Delft with the question of building software for one of its Spot robots. The goal of this software is to find specific items inside some confined space (like a hospital space) and match these items to the right person. The project will be based on the guidelines that were provided in the course manual[32] and will be done according to the project description provided. The robot will be controlled with ROS 1 and is equipped with a robotic arm for manipulation.[33].

2.1 Problem definition

The problem this project addresses, is utilizing a Spot robot to assist healthcare professionals in their daily tasks, for example, in a hospital. The goal is to make their jobs easier by enabling the robot to help them bring required objects to save time, such as food, bedding, or towels. By automating and supporting these tasks, the Spot robot aims to alleviate the workload of healthcare professionals, allowing them to provide better care and attention to patients. This presents a challenge due to the unstructured and dynamic nature of the hospital environment, as well as the absence of available maps. Additionally, it is important to avoid damaging surrounding objects or causing harm to humans.

2.2 Project planning

The project runs for 8 weeks which are split up into 4 sprints. The agile framework is used during this project, making the planning dynamic and iterative, with an emphasis on adaptability and collaboration. [20] The team continuously reviews and adjusts the planning based on feedback received after each sprint, allowing for responding to changing requirements. The planning of the project is visualized in a Gantt chart which can be found in the appendix C.

2.3 Vision

The group, in collaboration with TNO, aims to create a real-world application for the healthcare industry. With this application of Spot, we hope to contribute to mitigating the problem of understaffed healthcare facilities.[16] Spot will help reduce the workload for healthcare professionals and thereby enabling them to spend more time per patient, improving the quality of care patients receive. The vision of TNO is to strive to develop innovative solutions to improve human life and well-being, this aligns well with our own vision.

Application: To assist healthcare professionals, Spot will perform the following course of action: It starts by navigating and exploring/mapping the environment while searching for items. Upon request of a healthcare professional, Spot will move to/ pick up the requested item and approach the healthcare professional in a gentle manner. Spot initiates a conversation to validate if the picked item is correct. If so, Spot will give the item to the healthcare professional by moving its arm and opening its gripper.
The healthcare professional can provide the request to specify which item Spot shall deliver via a Smartwatch. This can take place in two ways: a spoken conversation or entering the needed assistance in a text entry field on the Smartwatch. Spot then proceeds to assist with the designated task.

Stakeholders: The most important stakeholders are the healthcare professionals themselves as they rely on Spot to make their job easier and enhance the care they provide to the patients. The second most important stakeholders are the patients who benefit from the robot's assistance since healthcare professionals can spend more time per patient, improving the quality of care they get. Another important stakeholder is TNO as they are the client for which the solution is being

developed. The healthcare facilities that will purchase and utilize this solution are also relevant stakeholders, as they will be integrating Spot into their care practices. Furthermore, stakeholders that are involved in the operational range of the robot - such as Boston Dynamics - contribute to the development and maintenance of Spot. Maintenance technicians who handle the robot's upkeep and repairs are also essential in ensuring minimal downtime. All these stakeholders play an important role in the successful implementation and utilization of Spot for assisting healthcare professionals in their daily tasks.

Finance: When looking at the Finances the Spot could be a cost-effective solution for healthcare facilities. Compared to hiring additional healthcare professionals to provide assistance, Spot can provide a wide range of services at a lower cost on a large scale. Additionally, Spot can help to reduce the workload of existing healthcare professionals, allowing them to focus on more critical tasks. The spot could be a valuable tool in providing high-quality care to patients while also keeping costs under control. [2]

Ethics: For Spot to do its job it needs to obtain private information like voice information or visual information, this can raise concerns for the privacy of the patients and the healthcare professionals. Because of this, the raw data is not stored and the created database of detections is anonymous. This database will be deleted each time Spot powers off. An important ethical concern that is raised is how the patients respond to having Spot walking around. Because the solution is non-conventional and being near Spot can sometimes elect feelings of fear extra care has to be taken when the robot is moving in the near vicinity of patients. Ways to do this are by having Spot walk extra slowly when near patients and by trying to keep patient interactions to a minimum.

Trends: Nowadays more and more robotic applications are centered around helping and supporting nurses in a wide variety of tasks within the healthcare facilities. Most of these applications try to solve the shortage of nurses within the healthcare sector. [8] This problem will only grow in the future due to the decreasing number of healthcare professionals. Our solution tries to fill the gap by reducing the workload of healthcare professionals, so they can focus on providing high-quality care.

Safety: Because Spot is going to be working with patients a goal is to make sure that walking around and with the robot is as safe as possible. The robot has built-in obstacle avoidance which helps it not collide with persons. It will also be possible for anyone wearing the bracelet to stop SPOT at any time. In addition, Spot will move slower and with less torque when in the proximity of persons.

2.3.1 TNO - SWOT analysis

The SWOT analysis of TNO provides a comprehensive assessment of the organization's strengths, weaknesses, opportunities, and threats within the context of the project.

Strengths

- Strong reputation in the research community especially in the Netherlands
- Diverse range of expertise and research areas
- Strong partnerships with industry, government, and universities
- High-quality research facilities and equipment

Weaknesses

- Heavy reliance on government funding
- Limited international recognition

Opportunities

- Growing demand for research in emerging technologies especially autonomous systems
- Increasing global investment in research and development in robotics
- Expansion into new international markets
- Collaboration with other research institutions and universities

Threats

- Increasing competition from other research institutions
- Uncertainty surrounding government funding for research
- Ethical implications of AI and robotics
- Rapidly evolving technology landscape
- Difficulty balancing short-term and long-term research goals

2.4 Operational scenarios

The Operational Scenarios provide an overview of the various situations encountered during the deployment and operation of the developed solution. By examining these scenarios an understanding of Spot's operational behavior and potential challenges can be gained.

Deployment Phase	
S1 Startup procedure	The operator places Spot inside of the room and then powers it up. When Spot is fully powered, the operator runs the launch file to start up the software.
Operation Phase	
S2 Mapping the environment	Spot walks in its environment while it constructs an occupancy map using the point clouds of both depth cameras and the estimated odometry. If an object is detected in the camera feed, the 2D location of the object is also marked on the occupancy map.
S3 Explore the environment	Spot samples a valid exploration direction along the border of the occupancy map. Spot then moves in this direction and thereby extends the map.
S4 Interact with the healthcare professional for mission assignment	The healthcare professional gives a request to Spot by specifying which item is required via his/hers Smartwatch. This can take place in two ways: a spoken conversation or entering the required item in a text entry field on the Smartwatch.
S5 Searching	When Spot knows what item should be found, but the object is not marked on the map, it explores its environment further/again to find the required item.
S6 Approach item	Once the required item is found, Spot needs to walk up to the item.
S7 Grab item	When Spot stands in front of the item, the arm needs to be operated to pick it up.
S8 Approach healthcare professional	Spot will approach the healthcare professional to deliver the required item.
S9 Mission confirmation	Spot interacts with the healthcare professional to ask for mission confirmation.
S10 Give the item to the healthcare professional	Once Spot stands in front of the healthcare professional and the mission is confirmed, the arm needs to be operated to hand over the item.
S11 Soft stop	If Spot ever loses the connection between critical ROS nodes or the soft stop is triggered using the Smartwatch, it terminates all procces and lay on the ground in idle mode.

Emergency Stop Phase		
S12	Emergency Stop	If Spot ever gets into a dangerous situation it needs to stop right away, this is achieved using an e-stop that is accessible by everyone.

Table 2: Operational scenarios for the Spot robot

2.5 Nodes

In this section, an analysis of existing code that relates to the project is outlined. Furthermore, the general ideas of nodes that have to be written are explained and the link between nodes and operational scenarios is made.

2.5.1 Pre-existing nodes

The project started with nodes related to starting the simulation environment and spawning the Spot robot in the environment. For each specialization research has been done about existing solutions that could be applied to the project:

Human Interaction:

- Plugins package for interaction in ROS [24]
- Speech to text [27]

Perception:

- Yolo detection [14]
- 3D bounding box creation [28]
- SLAM [23]

Planning/Navigation:

- RRT path planning [15]
- RRT explore package in ROS [25]
- Frontier Exploration [3]

The last node that needs to be created is the state machine. This node executes the required nodes in the correct order and is responsible for high-level decision-making. This node also handles any exception that could occur and makes sure that the robot is shut down safely when it gets into an unsafe state. This node is application-specific, so only frameworks can be used from existing code.

While all of this software could be used to create most of the functionality for the robot, the decision was made to write all of the nodes in Python manually. The reason for not using these pre-written nodes directly is to improve the learning experience and keep all the software under our own management. This creates the opportunity to optimize the software explicitly for Spot, improving the overall performance. Another reason is the stipulation for passing the course, where at least 5 nodes need to be developed, from scratch, that *significantly* contribute to the overall capabilities of the system.

2.5.2 Node outlines

Each specialization has various nodes that will be implemented in the robot:

Human Interaction:

- `bracelet_GUI`: This node implements a graphical user interface (Spot's control panel) that provides a very simple and intuitive way of specifying new missions, monitoring mission status, and triggering the soft/emergency stop. Since the control panel is operated on a smartwatch the design is simplistic and only consists of four different tiles, that change over time depending on the robot's state and the previous user input.

- `conversation`: This node implements the two different types of conversation with the operator, namely giving a new mission and confirming a successful mission. It uses speech-to-text to record operators' answers and text-to-speech to communicate with the operator using spoken text.

Perception:

- `plane_segmentation`: This node segments both incoming point clouds and publishes the combined ground- and non-ground points on two separate topics.

- `detection`: This node uses the two front-facing cameras of Spot to detect items and persons. It also uses the point cloud data to assign a 3d location to each detection. It publishes 2d bounding boxes and a database that contains the location of each detection, the class id, and the confidence of each detection.

- `occupancy_map`: Using the segmented point clouds and estimated odometry, this node constructs an occupancy map of the environment to indicate whether a cell is free, occupied or undiscovered.

Planning/Navigation:

- `explore`: This node uses the concept of gradients to sample valid exploration directions based on the borders of the occupancy map. This direction is then converted to a point on the border of the map and is sent to the `rrt_path` node to serve as a goal.

- `rrt_path`: Using the RRT* (Rapidly-Exploring Random Tree-Star) algorithm, this node computes a collision-free path for Spot to follow. The algorithm uses the occupancy map to ensure that the path is collision-free, taking the dimensions and orientation of the robot into account. This node takes a goal as input and sends each point in the computed path to the `motion_control` node.

- `motion_control`: This node receives points in the occupancy map and controls Spot by sending velocity commands to its drivers in order to move Spot. [4].

Executive Layer:

- `state_machine`: This node is used to handle different states and their respective transitions. By organizing Spot's behaviour into discrete states and controlling their transitions, the state machine enables robust decision-making and task execution.

[4]The Motion Control specialization was called off due to the Spot robot having an internal motion control, however this simple implementation of motion control was necessary to make Spot walking.

2.5.3 Nodes per operational scenario

The following overview has been created that shows the relationship between the nodes and the operational scenarios.

Operational Scenario: S2 Mapping the environment

- Nodes involved: `state_machine, plane_segmentation, detection, occupancy_map`
- Description: The `plane_segmentation` node segments the incoming point cloud data. These segmented point clouds are then combined with the odometry, in the `occupancy_map` node to create an occupancy map. Meanwhile, the detections made by the `detection` node are marked on the occupancy map. This entire process is controlled by the `state_machine` node.

Operational Scenario: S3 Explore the environment

- Nodes involved: `state_machine, motion_control, occupancy_map, explore, rrt_path`
- Description: The `explore` node samples a valid exploration direction/point, given the occupancy map. This point is sent to the `rrt_path` and the `motion_control` nodes to respectively create and execute a trajectory. During the trajectory execution, the operation scenario S2: Mapping the environment is running. This process is controlled by the `state_machine` node.

Operational Scenario: S4 Interact with healthcare professional for mission assignment

- Nodes involved: `state_machine, bracelet_GUI, conversation`
- Description: Via the Smartwatch, controlled by the `bracelet_GUI` and `conversation` nodes, the healthcare professional can request Spot by assigning him a mission given a required item by speech or text. The required connection is made and controlled via the `state_machine` node.

Operational Scenario: S5 Searching

- Nodes involved: `state_machine, plane_segmentation, detection, occupancy_map`
- Description: Similar to Operational Scenario: S2 Mapping the environment.

Operational Scenario: S6 Approach item

- Nodes involved: `state_machine, motion_control, rrt_path`
- Description: The `rrt_path` node computes a valid path for Spot to approach the location of the item. The `motion_control` node controls Spot's movement to reach the item. This process is controlled by the `state_machine` node.

Operational Scenario: S7 Grab item

- Nodes involved: `state_machine`
- Description: The `state_machine` node triggers the action for Spot to grab the item once it is in front of the item. This scenario is done manually.

Operational Scenario: S8 Approach healthcare professional

- Nodes involved: `state_machine, motion_control, detection, rrt_path`
- Description: The `rrt_path` node creates a collision-free trajectory that is executed by the `motion_control` to approach the healthcare professional. The final location of the healthcare professional is finetuned using the `detection` node. This process is controlled by the `state_machine` node.

Operational Scenario: S9 Mission confirmation

- Nodes involved: `state_machine, conversation`
- Description: When Spot approached the healthcare professional, a conversation is started using `conversion` node to get mission confirmation. This process is controlled by the `state_machine` node.

Operational Scenario: S10 Give item to healthcare professional

- Nodes involved: `state_machine`
- Description: The `state_machine` node triggers the action for Spot to give the item once the mission is confirmed by the healthcare professional. This scenario is done manually.

Operational Scenario: S11 Soft stop

- Nodes involved: `state_machine, bracelet_GUI`
- Description: The `state_machine` node detects if there is a loss of connection between critical ROS nodes or the soft stop is triggered by a healthcare professional using the Smartwatch controlled by the `bracelet_GUI` node.

Operational Scenario: S12 Emergency Stop

- Nodes involved: `state_machine`
- Description: If `state_machine` node detects a dangerous situation or the energy spot is pressed, Spot will abort all operations immediately.

3 Functional Architecture

This chapter delves into the functional architecture behind the solution for the problem definition. This architecture serves as a blueprint for the system's behaviour, outlining the various functionalities and how they relate to one another. It extends the operational scenarios, gives a clear overview of the operational needs and functional requirements, establishes a hierarchy tree organizing the modules, and illustrates the flow of activities with an activity diagram.

3.1 Extended operational scenarios

This subsection provides a comprehensive analysis of the operational scenarios discussed earlier, offering a deeper understanding of each operational situation. Each scenario is broken down into the following components: preconditions, triggering events, event descriptions, and post-conditions.

	S1 Startup procedure
Preconditions:	• All batteries are charged and have reached their maximum capacity. • The area around the robot is cleared.
Triggering event:	• Operator initiates the startup procedure.
Description:	• The operator performs the necessary steps to start up Spot and prepare it for operation.
Post-conditions:	• The system is successfully initialized and ready for operation.
	S2 Mapping the environment
Preconditions:	• System is initialized. • Both cameras are streaming. • Both depth cameras are streaming. • Environment is not fully mapped
Triggering event:	• State machine gives a signal
Description:	• Construct an occupancy map using point clouds of both depth cameras and estimated odometry data. If an object is detected in the camera feed, the 2D location of the object is also marked on the map.
Post-conditions:	• Occupancy grid is updated

	S3 Explore the environment
Preconditions:	• System is initialized • Occupancy map of scenario S2 has still undiscovered regions
Triggering event:	• State machine gives a signal
Description:	• Spot samples a valid exploration direction along the border of the occupancy map. Spot then moves in this direction and thereby extends the map
Post-conditions:	• Scenario S2 is activated, Spot is standing in a new pose and the mapping area is increased
	S4 Interact with healthcare professional for mission assignment
Preconditions:	• System is initialized • Operator has the bracelet GUI/Smartwatch to operate the robot • Required item to search for is unknown.
Triggering event:	• Healthcare professional strat requesting Spot.
Description:	• The healthcare professional interacts with Spot to assign a mission.
Post-conditions:	• Spot knows what item the healthcare professional needs.
	S5 Searching
Preconditions:	• System is initialized • Required object is known • Location of the required object is unknown
Triggering event:	• Spot has received a new mission from the healthcare professional
Description:	• Spot explores its environment further/again until the required item is found.
Post-conditions:	• Spot knows the location of the required item

	S6 Approach item
Preconditions:	• System is initialized. • Items location is known. • Spot is not standing in front of the item.
Triggering event:	• State machine gives a signal
Description:	• Spot approaches and stops in front of the item.
Post-conditions:	• Spot is in front of the item.
	S7 Grab item
Preconditions:	• System is initialized. • Items location is known. • Spot is in front of the item. • Spot's gripper is empty.
Triggering event:	• Scenario S6 is completed.
Description:	• Spot request help from the operator to move its arm and pick up the item.
Post-conditions:	• Spot has grabbed the item. • Spot's gripper is not empty.
	S8 Approach healthcare professional
Preconditions:	• System is initialized. • Required item is in the gripper of Spot • Location of the detected person is known (given by Smartwatch upon request)
Triggering event:	• Scenario S7 is completed.
Description:	• Spot approaches and stops in front of the healthcare professional
Post-conditions:	• Spot is in front of the healthcare professional

	S9 Mission conformationl
Preconditions:	• System is initialized • Required item is in the gripper of Spot • Spot is in front of the healthcare professional
Triggering event:	• Scenario S8 is completed.
Description:	• Spot starts an interaction with the healthcare professional to get mission confirmation.
Post-conditions:	• Spot's mission is confirmed
	S10 Give item to healthcare professional
Preconditions:	• System is initialized • Spot is in front of the healthcare professional • Spot has the requested item in his gripper • Mission is confirmed
Triggering event:	• Scenario S9 is completed.
Description:	• Spot request help from the operator to move its arm and give the item to the healthcare professional.
Post-conditions:	• The healthcare professional has the requested item. • Spot's gripper is empty.
	S11 Soft stop
Preconditions:	• All systems are running.
Triggering event:	• Robot losses connection, encounters a fatal error or the soft stop is triggered by a healthcare professional
Description:	• To illuminate the change of potential harm, Spot stops his operations and will lay on the ground in idle mode.
Post-conditions:	• Spot's systems are offline. • Spot is idle and lays on the ground.

	S12 Emergency Stop
Preconditions:	• All systems are running. • Spot creates or is in dangerous situations.
Triggering event:	• Human presses the emergency stop.
Description:	• Immediate kill of all operations and opening of the gripper to reduce possible harm.
Post-conditions:	• Spot's systems are offline. • Spot's gripper is open.

3.2 Needs and functional requirements

This subsection presents the needs and functional requirements of the system. The needs include various operational aspects, including startup procedures, room mapping, human interaction, object detection, and item delivery. On the other hand, the functional requirements outline the specific capabilities that the system must possess to fulfil these operational needs. These requirements include hardware initialization, system checks, speech recognition, path planning, object grabbing, and emergency stop functionality. This sets the stage for the subsequent development and evaluation of the system.

S1: Startup Procedure

Operational Needs:

- ON_S1_01: The system shall be able to start up self-test and do initialization at every moment when activated.

Functional Requirements:

- FR_S1_01: The system shall initialize the robot's hardware components, such as sensors, actuators, and communication modules.
- FR_S1_02: The system shall perform a system check to ensure all components are functioning properly.
- FR_S1_03: The system shall establish a connection with the control interface or mobile application.

S2: Mapping the environment

Operational Needs:

- ON_S2_01: The system shall be able to create an occupancy map of its environment.
- ON_S2_02: The system shall update the occupancy map of the room with new information when it is presented.

Functional Requirements:

- FR_S2_01: The system shall be able to map an environment a surface area of 100 m^2.
- FR_S2_02: The system shall be able to detect objects inside of the room and show these objects inside of its occupancy map.

S3: Explore the environment

Operational Needs:

- ON_S3_01: The system shall be able to find a valid exploration direction given the occupancy map.

Functional Requirements:

- FR_S3_01: The system shall be able to find a point that has at least 1 meter of undiscovered space in any direction of the point.

S4: Interact with healthcare professional for mission assignment

Operational Needs:

- ON_S4_01: The system shall be able to communicate with healthcare professionals via speech or text in order to get and confirm the mission.

Functional Requirements:

- FR_S4_01: The system is able to recognize English natural spoken language (speech to text)
- FR_S4_02: The system is able to output predefined answers to the recognized text in order to clarify/confirm the mission. It does so by asking for confirmation out-load.

S5: Searching

Operational Needs:

- ON_S5_01: The system shall be able to use the previously made occupancy map to plot path within the environment.
- ON_S5_02: The system shall be able to detect persons.
- ON_S5_03: The system shall be able to detect items that are common in a hospital setting.

Functional Requirements:

- FR_S5_01: The system is able to detect a human that is between 1.4 and 2 meters tall.
- FR_S5_02: The system is able to detect humans that are standing.
- FR_S5_03: The system is able to detect objects that have a size of maximum 175mm. [5] So that the gripper can pick up the items.
- FR_S5_04: The items that can be detected are given in the COCO dataset.

S6: Approach item

Operational Needs:

- ON_S6_01: The system shall be able to approach required items that are marked on the occupancy map.

Functional Requirements:

- FR_S6_01: The system is able to plan a trajectory to an item.
- FR_S6_02: The system is able to follow the found trajectory of FR_S6_01.
- FR_S6_03: The system shall approach the object with a maximum velocity of 2 ms^{-1}

S7: Grab item

Operational Needs:

- ON_S7_01: The system shall be able to alert a human operator when it is ready to pick up an item

Functional Requirements:

- FR_S7_01: The system shall be able to give full operational control to the human operator
- FR_S7_02: The system shall be able to use a speaker on the laptop to alert the human operator the system is ready for picking up an item.
- FR_S7_03: The system is able to detect objects that have a size of maximum 175mm. [5] So that the gripper can pick up the items.

S8: Approach healthcare professional

Operational Needs:

- ON_S8_01: The system shall be able to approach and stop in front of a detected person.

Functional Requirements:

- FR_S8_01: The system shall be able to stop 0.5m in front of the detected human.
- FR_S8_02: The system shall be able to decrease its speed upon being in proximity of a human (1m) and decrease the speed further when it comes closer.

S9: Misson conformation

Operational Needs:

- ON_S9_01: The system shall be able to start a conversation with the healthcare professional to ask for mission confirmation.

Functional Requirements:

- FR_S9_01: The system is able to recognize English natural spoken language.
- FR_S9_02: The system is able to output predefined answers to the recognized text in order to clarify/confirm the mission. It does so by asking for confirmation out-load.

S10: Give item to healthcare professional

Operational Needs:

- ON_S10_01: The system shall be able to alert a human operator when it is ready to hand over the item.

Functional Requirements:

- FR_S10_01: See FR_S7_01.
- FR_S10_02: See FR_S7_02.
- FR_S10_03: The system is able to give objects to humans within 2-5 cm.

S11: Soft stop

Operational Needs:

- ON_S11_01: The system shall have a method to detect faults within node communication and alert the operator when this is the case.
- ON_S11_02: The system shall have a method to detect faults and error messages within ros and display these to the terminal for a human operator to see.

Functional Requirements:

- FR_S11_01: The system shall be able to use the speaker on the laptop when it detects a fault.
- FR_S11_02: The system is able to monitor inter-node communication
- FR_S11_03: The system will try to go to an idle state when it detects a software error.

S12: Emergency Stop

Operational Needs:

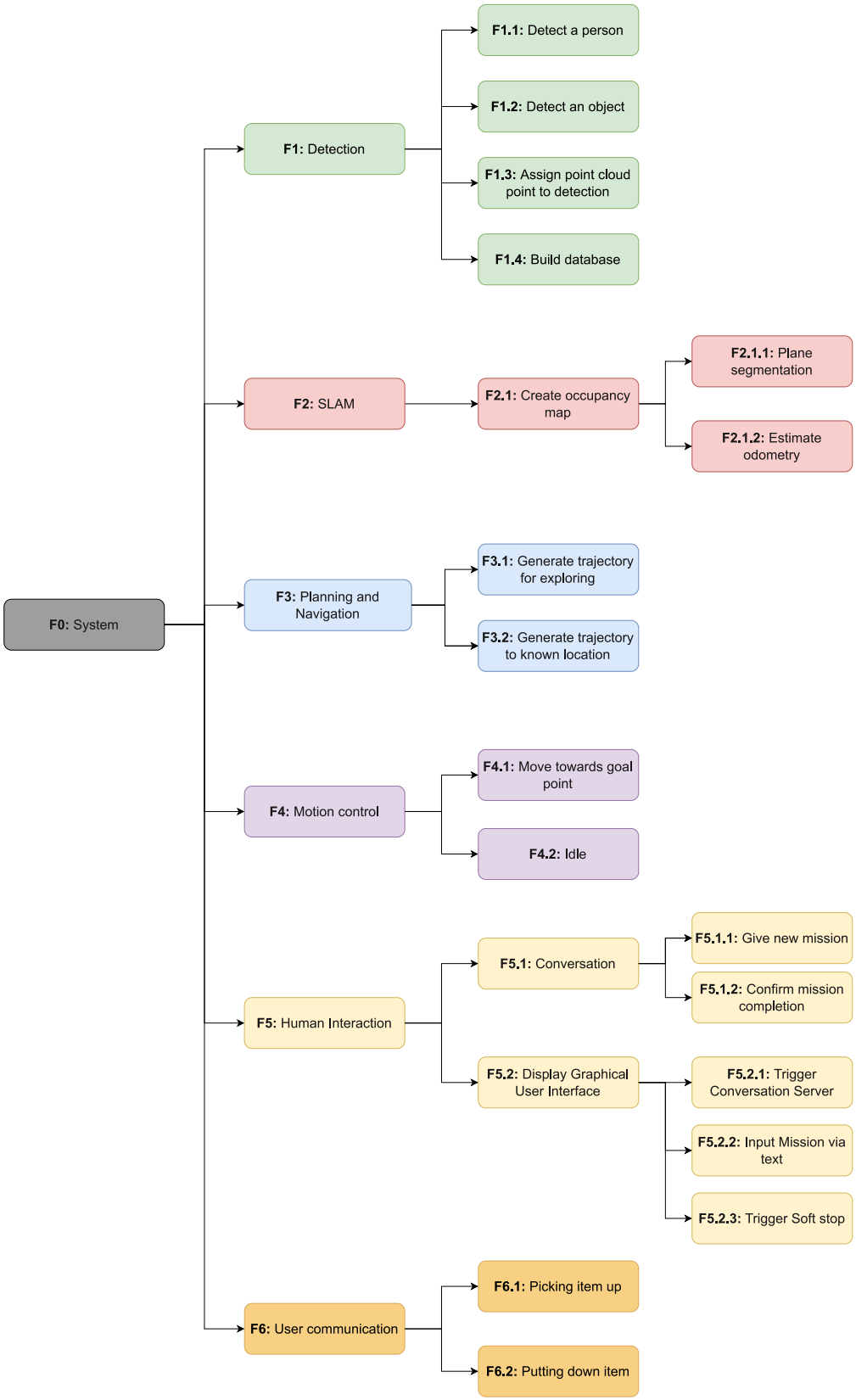
- ON_S12_01: The system shall quickly halt all robot operations in emergency situations.

Functional Requirements:

- FR_S12_01: The system is able to provide a clearly identifiable and easily accessible Emergency Stop button or switch.
- FR_S12_02: The system is able to initiate an immediate and complete shutdown of all robot movements and actions when the Emergency Stop is activated.
- FR_S12_03: The system is able to disable all motor outputs and power to prevent any unintended motion or operation.
- FR_S12_04: The system is able to communicate the activation of the Emergency Stop to the operator or monitoring system.
- FR_S12_05: The system is able to override any ongoing actions or behaviors, prioritizing safety over other tasks.
- FR_S12_06: The system is able to implement a fail-safe mechanism that requires manual intervention to reset the Emergency Stop state.
- FR_S12_07: The system is able to ensure that once the Emergency Stop is reset, the robot can resume normal operation safely.

3.3 Functional hierarchy tree

In this section, the functional hierarchy tree is shown and its subparts are explained in detail.

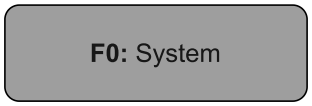


There are six high-level system functionalities that can be used to map the functional requirements:

0. System
1. Detection
2. SLAM
3. Planning and Navigation
4. Motion control
5. Human Interaction
6. User communication

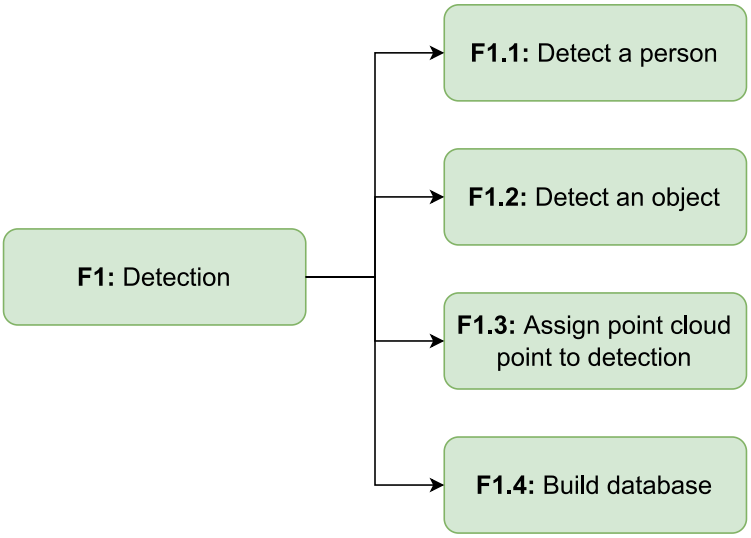
These high-level functionalities can be split up into lower-level functionalities. Even some of the lower-level functionalities can be split up. The whole hierarchy tree can be seen in figure 3.3 and each system will be elaborated on below.

F0: System



Function F0	System
Description	Behaviour model that decides when the system should transition to a next states
Input	Start-up of the system
Output	State transitions using actions and services
Parent function	None
Children function	All

F1: Detection



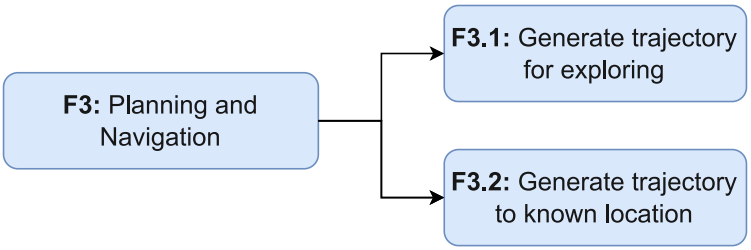
Function F1	Detection
Description	Spot detects when a human or a specific object is in view of the camera
Input	Camera input and pointcloud
Output	Database containing information about all found detections
Parent function	F0
Children function	F1.1, F1.2, F1.3, F1.4
Function F1.1	Detect a person
Description	An image is obtained and yolo detection is applied to this image
Input	Image
Output	2d bounding boxes
Parent function	F1
Children function	None
Function F1.2	Detect an object
Description	An image is obtained and yolo detection is applied to this image
Input	Image
Output	Bounding boxes
Parent function	F1
Children function	None
Function F1.3	Assign point cloud point to detection
Description	Spot uses point cloud information to assign a 3d location to a person
Input	Camera input and point cloud information
Output	3d location of a yolo detection
Parent function	F1
Children function	None

Function F1.4	Build database
Description	Combines all found information in a database
Input	Detection information and 3d point information
Output	Datbase
Parent function	F1
Children function	None

F2: Self Localisation And Mapping (SLAM)

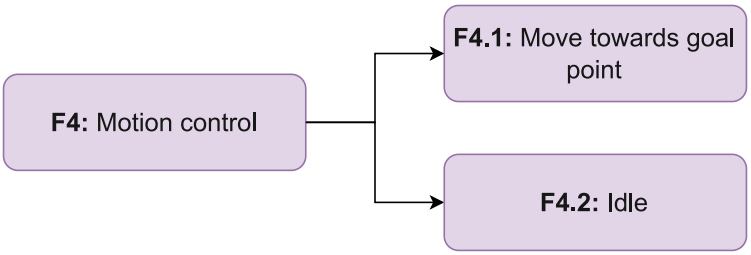
Function F2	Self Localisation And Mapping (SLAM)
Description	Spot can localize itself within the constructed occupancy map of its surroundings.
Input	Occupancy map
Output	Self localization within the occupancy map of its surroundings
Parent function	F0
Children function	F2.1
Function F2.1	Create occupancy map
Description	Spot can construct an occupancy map of its surroundings and localize itself within this map.
Input	Segmented pointclouds and odometry estimation
Output	Occupancy map
Parent function	None
Children function	F2.1.1, F2.1.2
Function F2.1.1	Plane segmentation
Description	Segmenting the ground plane from the incoming raw pointclouds.
Input	Raw pointclouds
Output	Two separate pointclouds containing ground and nonground points respectively,
Parent function	F2.1
Children function	None
Function F2.1.2	Estimate odometry
Description	Estimate the current position of the Spot within the odom frame.
Input	Odometry measurements
Output	Estimated robot pose in odom frame
Parent function	F2.1
Children function	None

F3: Planning and Navigation



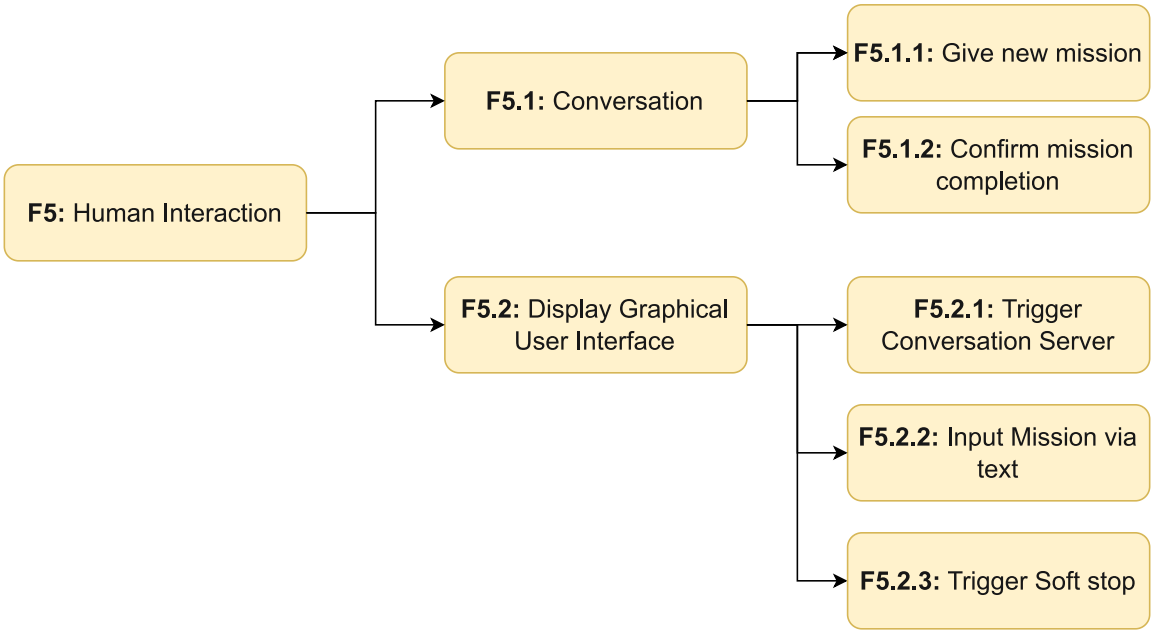
Function F3	Planning and Navigation
Description	Spot is able to generate paths for specific task.
Input	Occupancy grid
Output	Planned trajectory
Parent function	F0
Children function	F3.1, F3.2
Function F3.1	Generate trajectory for exploring
Description	Spot generates a trajectory for exploring the environment. Based on the occupancy grid a trajectory is generated towards an unkown frontier.
Input	Occupancy grid
Output	Path to follow
Parent function	F3
Children function	None
Function F3.2	Generate trajectory to known location
Description	Spot generates a trajectory towards a given objective, which is either the human or the specified object.
Input	Occupancy grid, object classification, object location
Output	Path
Parent function	F3
Children function	None

F4: Motion control



Function F4	Motion control
Description	Spot moves towards a given goal position and goes into idle when mapping is complete and has no mission. When a mission is requested it transitions to an active state to fulfill the assigned task.
Input	Goal position
Output	Spot moved to position
Parent function	F0
Children function	F4.1, F4.2
Function F4-1	Move towards goal point
Description	Given the goal position a velocity command is send to the Spot driver so it can reach the goal position.
Input	Goal position
Output	Velocity command
Parent function	F4
Children function	None
Function F4-2	Idle
Description	Spot goes into idle when mapping is complete and has no mission. When a mission is requested it transitions to an active state to fulfill the assigned task.
Input	Map completion and no mission
Output	Search for mission and activates when mission is requested
Parent function	F4
Children function	None

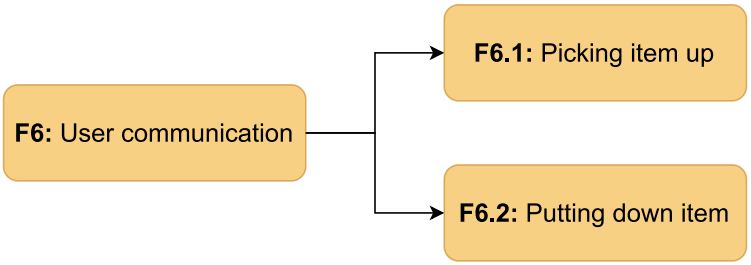
F5: Human Interaction



Function F5	Human interaction
Description	Spot is able to interact with humans (using speech and the graphical user interface) so that mission control can be performed by operators.
Input	Given through graphical user interface or spoken natural language
Output	Triggering desired Spot behaviour
Parent function	F0
Children function	F5.1, F5.2
Function F5.1	Conversation
Description	Spot communicates with the operator to get and confirm missions.
Input	Spoken natural language, trigger from state machine or GUI
Output	Synthesized text-to-speech, specification of required item or mission confirmation
Parent function	F5
Children function	F5.1.1. F5.1.2
Function F5.1.1	Give new mission
Description	In a conversational loop, the required item for the new mission can be specified.
Input	Spoken natural language, available items
Output	Synthesized text-to-speech, string of specified item for new mission
Parent function	F5.1
Children function	None

Function F5.1.2	Confirm mission completion
Description	In a conversational loop, the current mission can be confirmed as successful or a new item can be specified when the wrong one was delivered.
Input	Spoken natural language, item in gripper
Output	Synthesized text-to-speech, mission confirmation status or new item to get
Parent function	F5.1
Children function	None
Function F5.2	Display Graphical User Interface (GUI)
Description	A GUI can be displayed on a bracelet/smartwatch that enables the operator to control Spot's behavior.
Input	Touch gestures and text entries on GUI, spoken natural language, mission status published by Spot
Output	Displays current mission status and triggers Spot behaviors
Parent function	F5
Children function	F5.2.1, F5.2.2, F5.2.3
Function F5.2.1	Trigger conversation server
Description	The conversation server can be triggered from the GUI in order to give new missions and confirm completed ones as described in F5.1.
Input	Touch gestures on the GUI
Output	Synthesized text-to-speech, triggers conversation server
Parent function	F5.2
Children function	None
Function F5.2.2	Input mission via text
Description	A text entry field on the GUI can be used to specify new missions.
Input	Touch gestures and text entries on GUI
Output	Specification of new mission and item to get
Parent function	F5.2
Children function	None
Function F5.2.3	Trigger soft stop
Description	The soft stop service can be triggered from the GUI in the case that operators notice dangerous/unwanted behavior.
Input	Touch gestures on GUI
Output	Triggers ROS service /spot/estop/gentle
Parent function	F5.2
Children function	None

F6: User communication



Function F6	User communication
Description	Spot sends a message to the user so they can manually pick up and put down the requested item.
Input	Notification to pick up or put down an item
Output	Message
Parent function	F0
Children function	F6.1, F6.2
Function F6.1	Picking item up
Description	Spot sends a message to the user so that s/he can use the controller to pick up the item.
Input	Flag by state machine
Output	Message
Parent function	F6
Children function	None
Function F6.2	Putting item down
Description	Spot sends a message to the user so that s/he can use the controller to put down the item.
Input	Flag by state machine
Output	Message
Parent function	F6
Children function	None

3.4 Activity diagram

In this chapter, the activity diagram is explained and visualized in figure 3. The in-detail descriptions of the workings of each of the involved nodes can be found in Chapter 4. To implement the executive layer of this project, a state machine is used that handles different states and their respective transitions. It is based on the above-explained hierarchy tree. By organizing Spot's behavior into discrete states and controlling their transitions, the state machine enables robust decision-making and task execution.

After the startup of the system, the first executed state is ”Mapping”. The final goal of that state is to create a 2D occupancy grid (F2.1) of the environment and build up a database (F1.4) containing the detected objects and persons as well as their respective locations. To create the occupancy map, the pose is estimated using odometry data (F2.1.2), and ground plane segmentation is performed (F2.1.2). To build the database, both person detection (F1.1) and object detection (F1.2) are run on the current images. If there was a detection made, it is assigned a 3D location (F1.3), and the database is updated (F1.4). To fully map the environment, the state implements a frontier exploration algorithm that generates (F3.1) and executes trajectories (F4.1) to unmapped areas. To ascertain the completion of the mapping process, a predefined threshold is evaluated that measures the rate of change in the map per scan.

After ”Mapping” is complete and no mission request is given yet, Spot will remain ”Idle” and lay down until a mission is specified (F4.2).

If no mission is given yet, but the bracelet is active the operators can ”Give new mission” by using spoken natural language (F5.1.1) or the text entry on their bracelet (F5.2.2).

After the item is specified Spot is in the state ”Approach goal”. It generates (F3.1) and executes (F4.1.) a path to that item based on a query of the database.

Subsequently, the State ”Pick item” is triggered, which requests the operator to remotely pick up the object Spot is standing in front of (F6.1). If the item picking failed 3 times in a row, the mission will be aborted.

With the loaded gripper, Spot executes the state ”Approach goal” to the human that gave the mission and enters the state ”Confirm mission” to determine whether the item brought is correct (F5.1.2). If that is the case, it puts the item down (F6.2). If that is not the case, the item will be returned to its original location through the execution of the state ”Approach goal” and ”Place wrong item”. It will then transition to the state ”Idle” to wait for new missions.

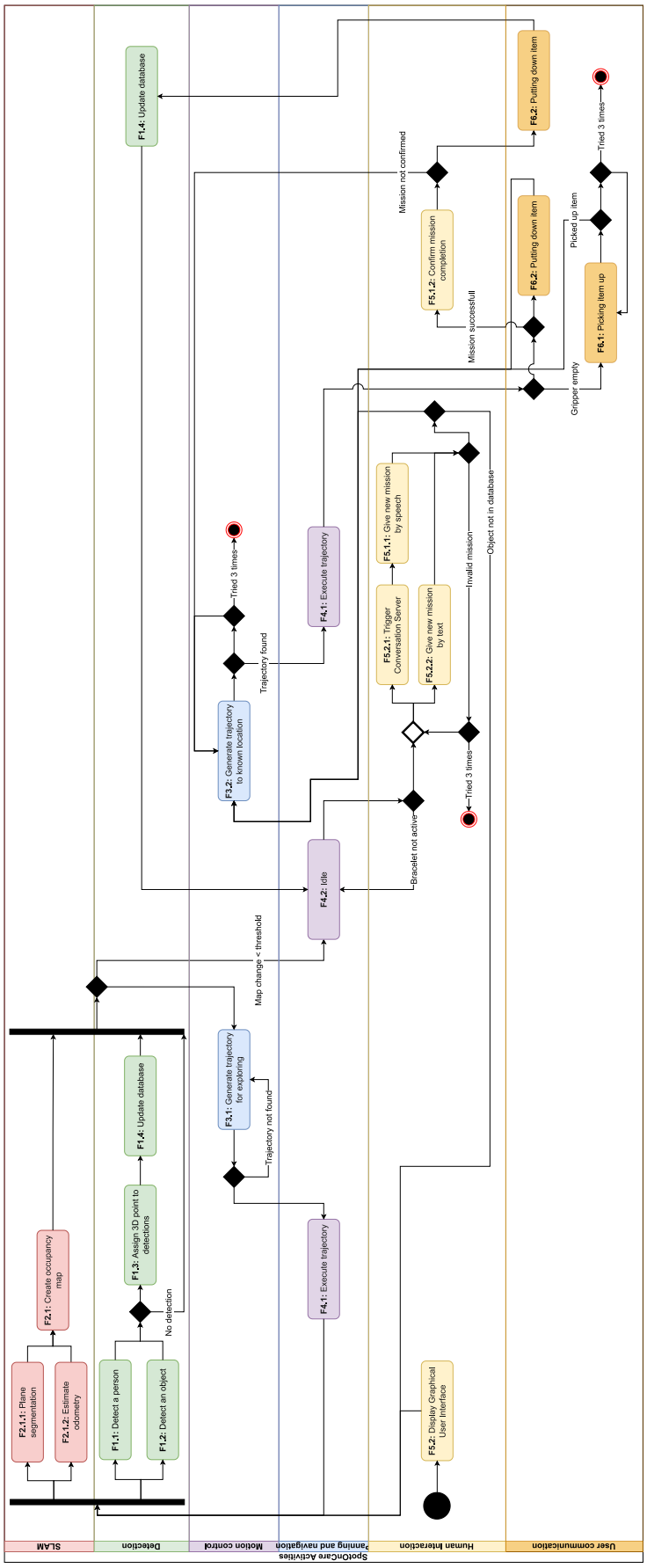


Figure 3: Activity diagram

4 Full Functional Description of the Robot Software

This chapter offers a detailed exploration of the software system developed for Spot. Whereas the previous chapter discussed the functional architecture and what the system is supposed to do. This chapter dives more into detail about the actual implementation for the Spot inside simulation. It begins by presenting a functional graph that represents the Spot software inside the simulation, accompanied by a comprehensive description of its functioning. The functional graph highlights the interconnections and relationships between various nodes and topics from the Spot, showcasing the contributions of individual team members. Furthermore, the chapter delves into the recorded behaviors from the Spot in the simulation, providing figures that demonstrate the main behaviors of the Spot, corresponding to the specializations within the team. In the end, the executive layer with the implemented state machine will be described to provide a summary of the system for Spot.

The full code repository can be found on Gitlab[5] and a video of the demonstration in the simulation can be found here[6].

4.1 Functional Graph

To present the functional graph of Spot the full rqt-graph has been split up into the two figures 4 and 5 . These two figures show all the nodes and topics made by the group and each node and topic is colored corresponding to their high-level functionality from 3.3. The full rqt-graph can be found in appendix E.

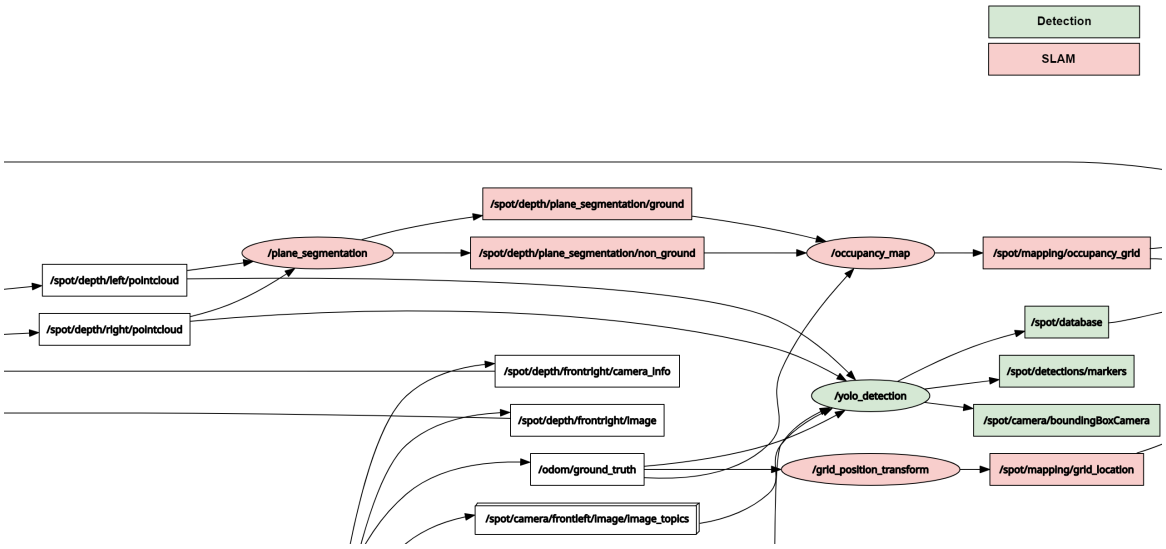


Figure 4: rqt-graph for SLAM and Detection

The node /plane_segmentation receives topics */spot/depth/left/pointcloud* and */spot/depth/right/pointcloud* to derive and publishes */spot/depth/plane_segmentation/ground* and */spot/depth/plane_segmentation/non_ground*. These topics are used by the node `/occupancy_map` which computes the *spot/mapping/occupancy_grid* topic. This topic is then used by nodes `/rrt` and `/explore_node` from figure 5. The main contributor to these nodes and topics corresponding to **F2: SLAM** are made by Guido Dumont.

[5]GitLab: `https://gitlab.tudelft.nl/cor/ro47007/2023/team-19/champ_spot`
[6]Full simulation run: `https://www.youtube.com/watch?v=niYJE1J9ac8`

Topics */spot/depth/left/pointcloud*, */spot/depth/right/pointcloud*, */odom/ground/truth*, */spot/camera/-frontleft/image/image_topics* and */spot/camera/frontright/image/image_topics* are subscribed to by the `/yolo_detection` node. This node publishes */spot/detections/markers* and */spot/- camera/boundingBoxCamera* topics that are used for visualizing the results of the node. The node also publishes */spot/database*, which is used by `/state_machine` in 5 to process the results. Jesse Dolfin is the main contributor of these nodes and topics corresponding to **F1: Detection**.

The node `/grid_position_transform` simply transforms topics from */odom/ground_truth* to a valid coordinate in the occupancy grid. This is then published as topic *spot/mapping/grid_location* which is used by the `/motion_control` node. The main contributor to this was Jurjen Scharringa.

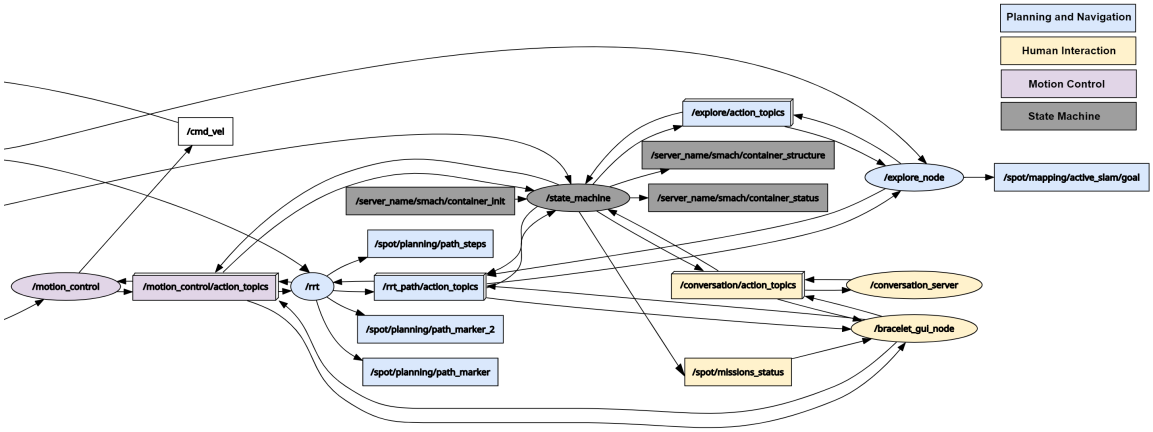


Figure 5: The rqt-graph for Planning and Navigation, Human Interaction, Motion Control and State Machine

For functionality **F4: Motion control** node `/motion_control` was made. This node listens to topic */spot/mapping/grid_location* and works as an action server to topic */motion_control/action_topics*. It publishes to */cmd_vel* so that Spot is able to move. The main contributor to this was Jurjen Scharringa.

For the pathfinding in **F3: Planning and Navigation** the `/rrt` node is responsible. It listens to the */spot/mapping/occpuancy_grid* topic and works as an action server with */motion_control/action_topics* and */rrt_path/action_topics*. It also published */spot/planning/path_steps*, */spot/planning/path_marker* and */spot/planning/path_marker_2* which can be used for debugging in RVIZ. The main contributor to this was Marco de Böck with the help of Jurjen Scharringa.

The `/explore_node` is responsible for the exploring in **F3: Planning and Navigation**. It is subscribed to the */spot/mapping/occpuancy_grid* topic and also works as an action server with */explore/action_topics* and */rrt_path/action_topics*. For debugging purposes, it also publishes the topic */spot/mapping/active_slam/goal*. The main contribution came from Guido Dumont with Jurjen Scharringa helping.

The first node responsible for **F5: Human Interaction** is `/conversation_server` which acts as an action server to topic */conversation/action_topics*. The second node is `/bracelet_gui_node`. It is subscribed to */spot/missions_status* and acts as a action server for both */rrt_path/action_topics* as */conversation/action_topics*. Jakob Bichler was the main contributor to the Human Interaction nodes and topics.

The final functionality **F0: System** connects every node together to a fully functioning system. This is done with the `/state_machine` node. The main contributor was Jakob Bichler with each specialist helping to connect their nodes and topics to the state machine.

4.2 Perception

The perception specialization has an important task in creating the final product. The goal is for Spot to work autonomously in a hospital environment, without perception this would not have been possible. Spot cannot work with a hard-coded environment as hospitals can be dynamic environments and the goal is for Spot to be employed at various different hospitals. Therefore it is required for the perception specialization to map the environment while running.
Another reason that supports the importance of perception is that it detects humans, the desired objects and obstacles. Obstacle detection is required for Spot to safely traverse the environment. As Spot works in a hospital, safety is one of the top priorities. Human and object detection is required for Spot to receive goals for the other specializations.
The perception specialization is responsible for both **F1: Detection** as **F2: SLAM**.

4.2.1 SLAM

In order to improve the flexibility of the system and make Spot usable in unknown/unmapped environments, such as hospitals, Spot has to be able to map its surroundings in real-time. Spot can then use his map to plan collision-free trajectories, save locations of detected items/persons and improve the searching algorithm by exploring the unkown/-mapped regions of the environment.

Motivation

In literature, this kind of mapping is often performed with the Self Localization And Mapping algorithm (SLAM) [4]. This algorithm uses 3D measurements from lidar, radar or RGBD sensors, in combination with odometry estimation, to create an occupancy map of the environment. Due to the popularity of this algorithm within robotics, there exists a ROS node within the ROS toolbox to perform the SLAM algorithm. However, to enhance the learning experience, the group decided to create their own implementation of the SLAM algorithm from scratch.
The SLAM algorithm is responsible for function **F2.1: Create occupancy map**.

Technical Details

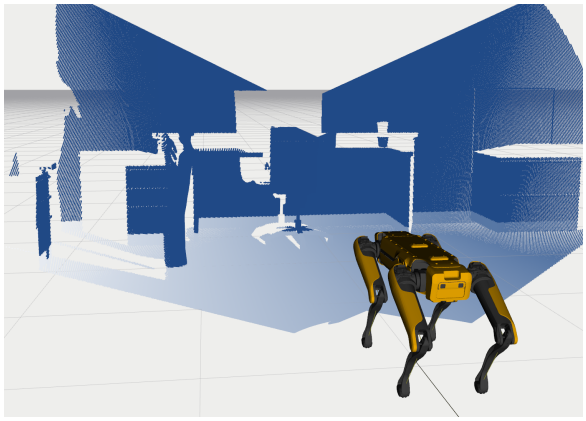
Before an occupancy map can be created using the SLAM algorithm, data preparation has to be done on the incoming RGBD measurements from the cameras. This data preparation step consists of two steps which are explained below.

1. Depth-images to pointclouds:
Spot's depth cameras measure depth in the form of depth images. A depth image contains information about the relative distance between a surface and the camera in one of the channels. However, these depth images need to be converted to pointclouds in order to serve as input for the SLAM algorithm. This is done using a build-in ROS node called `nodelet` to perform `depth_image_proc/point_cloud_xyz` [19].

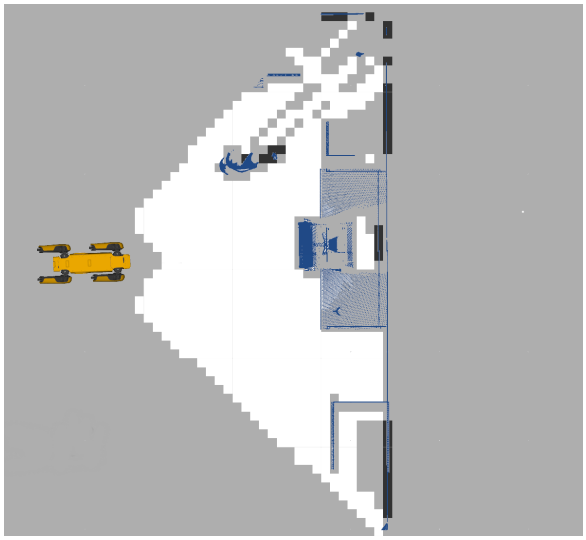
2. Ground plane segmentation:
To determine from the pointcloud which points correspond to the ground plane and which points to obstacles, ground plane segmentation is performed. Under the assumption that the working environment of Spot is on flat ground (inside) and to reduce computational time, plane segmentation is performed using a fixed Z-height threshold. This plane segmentation node publishes the ground and nonground points separately on two different topics, see figure 6a.

Recorded Behaviour

Using the segmented pointclouds and the estimated odometry data, an occupancy map with a resolution of 0.1 meters is constructed, see figure 6b. Every cell in this map has a value between 0 and 100, denoting the probability of occupied space or a value of -1 indicating that the cell is undiscovered. Initially, every cell on the map is undiscovered and has a value of -1. However as measurements come in, the occupancy probability of the visible cells changes towards either a low or high probability of occupancy. This probability converges when multiple sequential measurements of the same cell confirm the predicted state. The occupancy map updates in real-time, meaning that the map expands when Spot is moving around.



(a) Ground plane segmentation



(b) Occupancy map with segmented pointcloud

Figure 6: Spot performing SLAM

4.2.2 Detection

The goal of Spot is to find and deliver a particular item to a person. To do this Spot needs to know *what* the item is and *where* both the item and the persons the items need to be delivered to are. To accomplish this a detection algorithm has been designed that assigns a 3d point to a detection. The detection self is done with the Yolov7 algorithm [22] trained on the COCO dataset.

Motivation

The motivation behind using yolov7 is two-fold:

1. Yolo is a classification algorithm that is optimized for speed, it can work in real-time which is a necessity for this application. The reason for this is that the environment is prone to change rather fast which means that by the time an image is received in which a person is detected, the person may already not be at that location anymore. The comparison of Yolov7 to other networks can be found in Figure 7.

2. Open cv has integrated the use of the Yolo algorithm by means of a configuration file and weight file in their Python library. This allows for quicker development of the algorithm when compared to building its own detection pipeline.

The reason the COCO dataset has been chosen is that yolov7 has already been trained on this dataset which saves the trouble of re-training the algorithm. Furthermore, the COCO dataset already contains many items that can be found in a healthcare setting.

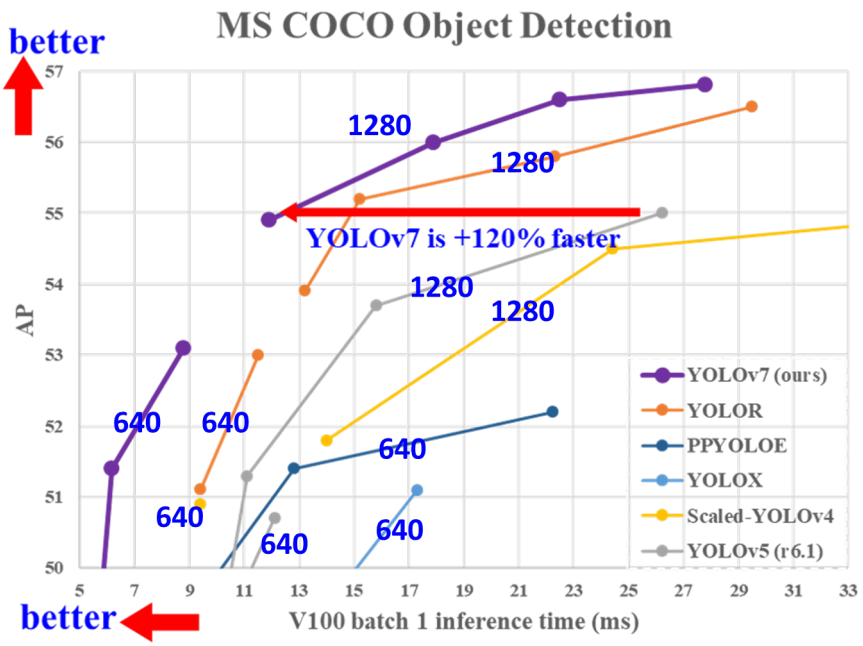


Figure 7: Yolov7 comparison to other detection models [13]

This node is responsible for functions **F1.1: Detect a person**, **F1.2: Detect an object**, **F1.3: Assign pointcloud point to detection** and **F1.4: Build database**.

Technical details

The working of the node can be explained in a couple of steps:

1. The node subscribes to the front-left and front-right cameras of the Spot robot. The node also subscribes to the point cloud data.
2. Yolo is applied to the image which then produces 2d bounding boxes. The center location of these bounding boxes is known.
3. The point cloud data is transformed to the camera frame.
4. The intrinsic camera properties are then used to determine the point cloud point that is closest to the u,v pixel coordinates of the center point of the detection.
5. The found point is transformed back into a world-fixed frame.
6. If the point is unique (there may not have been a previous detection within a set radius from this point) it is added to a database together with the type of detection (e.g.: ”person”, ”cup” etc) and the confidence of the detection.
7. The database is then published using a custom message

There are two solutions presented, the `yolo_detection.py` and the `arm_detection.py`. In the real world, the arm of the robot is used to obtain images, this is because it captures images in color without the need for a complicated SDK setup. Because the arm is higher up and centered no camera transformations need to be completed before the image can be obtained. This is the solution presented in the `arm_detection.py`. The other file is used for simulation and to test the code but is also able to be deployed on the real system if that is desired.

Recorded behaviour

The node outputs 2D location markers which can be used in accordance with the SLAM node to add detections to the map as can be seen in figure 9a and 9b. The node also draws 2D bounding boxes on the image which results can be seen visualized in figure 8.

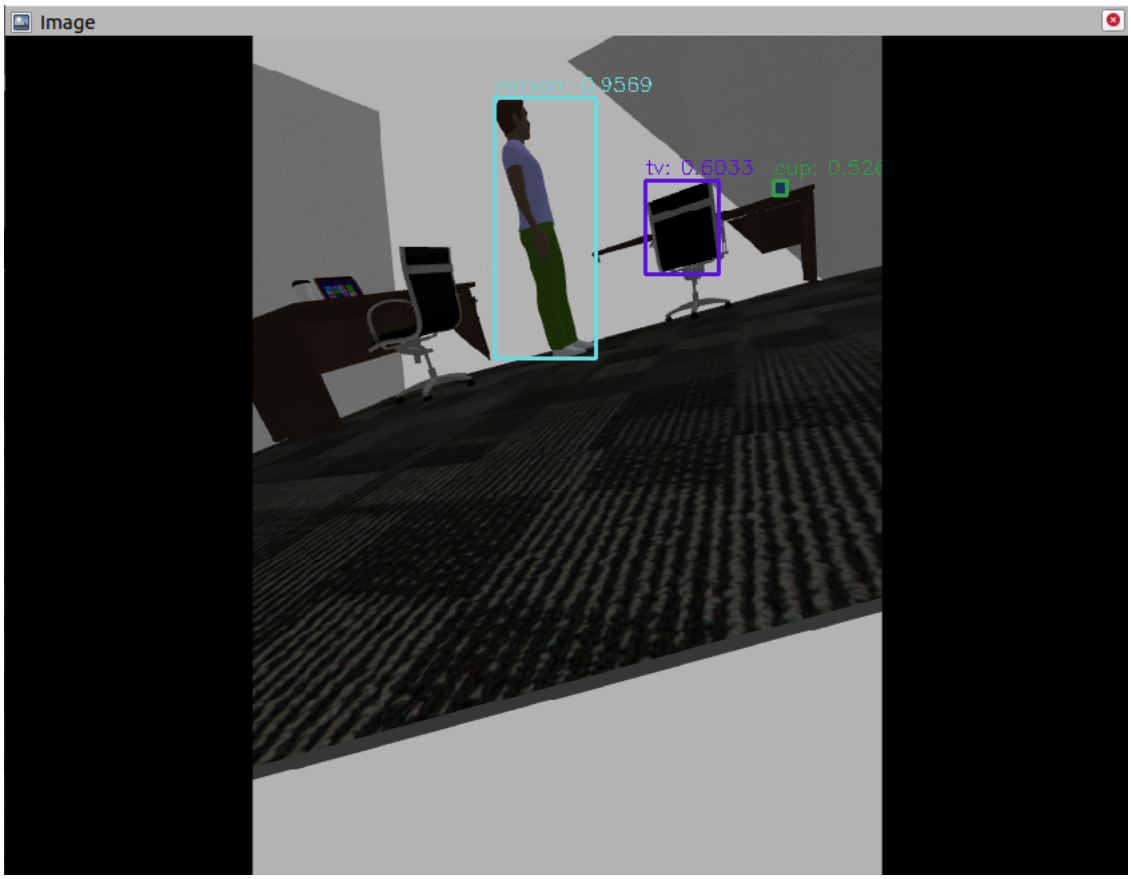
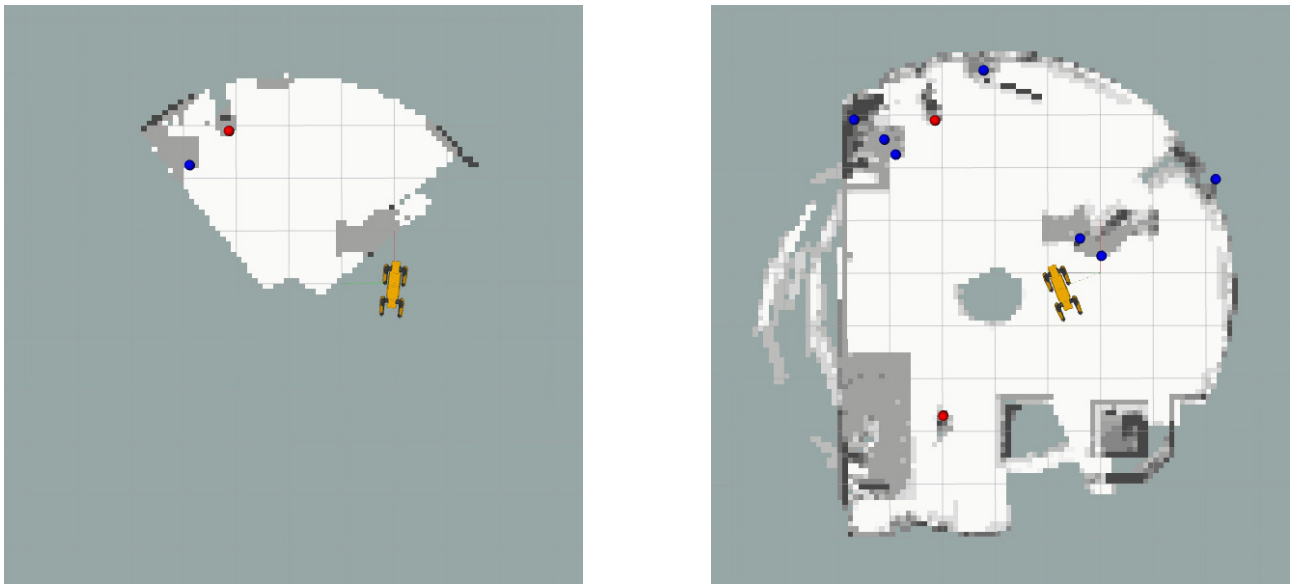


Figure 8: Output of the yolo detection

The location markers are visualized on the map. A red marker represents a person and a blue marker represents anything else. A thing to note on this map is that at each red location, the occupancy map shows that there is something standing there, which shows that the correct point is assigned to the detection.



(a) Initial mapping

(b) Mapping after full rotation

Figure 9: Output of the locations of detections

4.3 Planning and Navigation

The Planning and Navigation specialization plays a vital role in Spot's functionality. It allows Spot to navigate autonomously through the environment. In healthcare settings, the obstacles that Spot may encounter can include patients, healthcare professionals, and medical equipment. To ensure safety in the environment for both patients as professionals it is crucial to prevent any collisions or accidents. Thus having effective path finding capabilities become essential for creating a secure and harzard-free environment withing healthcare facilities.
For Spot to be actually helpful for healthcare professionals it is important that Spot can act autonomously. It should explore the unknown area so that it can map the whole environment. Exploring should be done autonomously without having to be send a location manually.
The Planning and Navigation specialization is responsible for the system function **F3: Planning and Navigation**. The specialization is split into path finding and exploration.

4.3.1 RRT* Path Planning

To ensure safety in the environment and also prevent any collisions or accidents having an effective path becomes essential. There is a great deal of different path planning algorithms. [26] To achieve this the team has chosen to use the RRT* (Rapidly-exploring Random Tree star) Algorithm. To construct the RRT* code inspiration is taken from the paper introducing the RRT* Algorithm.[11]. The RRT-Path Planning node is responsible for function **F3.2: Generate trajectory to known location** this node generates a path with the RRT* Algorithm to the known location.

Motivation

The reason to work with a Rapidly-exploring Random Tree algorithm is its flexibility in dynamic environments. As it is in the algorithm's nature to sample randomly, it is very robust to changes in the environment [12]. By probabilistically exploring the occupancy grid it inherently accommodates the changes in the grid such as first unknown spaces into known free or occupied changes.
Improving the RRT algorithm to RRT* was done to improve the path quality and optimality by adding an optimization step that improves the path quality by rewiring the tree structure. Another reason to work with this algorithm is that is has a high real-time performance. The environment is not complex and the emphasis is generating a quick and good solution.

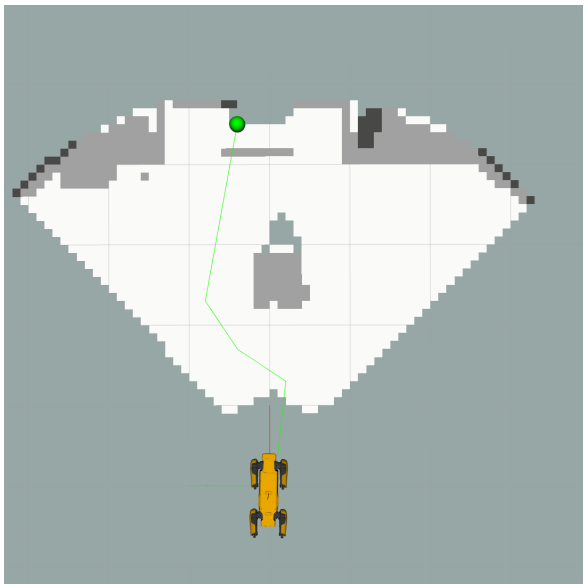
Technical Details

The working of the node can be explained in a couple of steps:

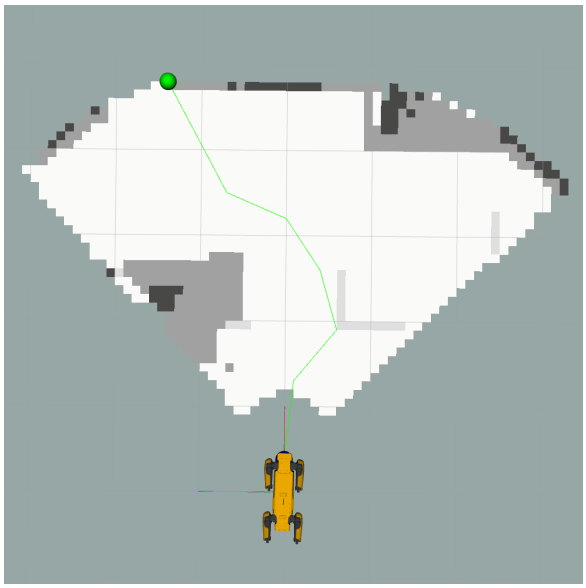
1. Before an RRT* path can be created, the following information needs to be known:
   a) Map: The map is given by the occupancy map created in the SLAM node.
   b) Goal Position: The goal is provided by the detection node, explore node, or human interaction node.
2. Once the map and goal are known, the `rrt_path` node will be triggered when a goal position is given to the Client server. The `rrt_path` node contains the file that computes the path for SPOT to follow. This path is computed using the RRT* algorithm within the OccupancyGrid.
3. The `rrt_path` node acts as an ActionServer that is triggered when it receives a MoveBaseGoal. This MoveBaseGoal will come from the `yolo_detection`, `bracelet_gui_node`, `conversation_server` node, or the explore node.
4. After receiving the goal, the path is computed using the RRT* algorithm in the OccupancyGrid.
5. This code implements the RRT* path planning algorithm within the environment.
6. The `RRT` class manages the RRT* algorithm, using a start node, goal node, and occupancy grid map. The algorithm generates a collision-free path by extending the tree through randomly generated nodes and connecting them to the nearest existing node.
7. The resulting path is executed using an action client, and the outcome is returned to the action server.
8. For each point in this calculated path, the `motion_control` node will be triggered and will move to the first point of the given path. After reaching the first point, the server notifies a success, and the next point in the path is sent to the server. This process continues until the goal is reached.

Recorded Behaviour

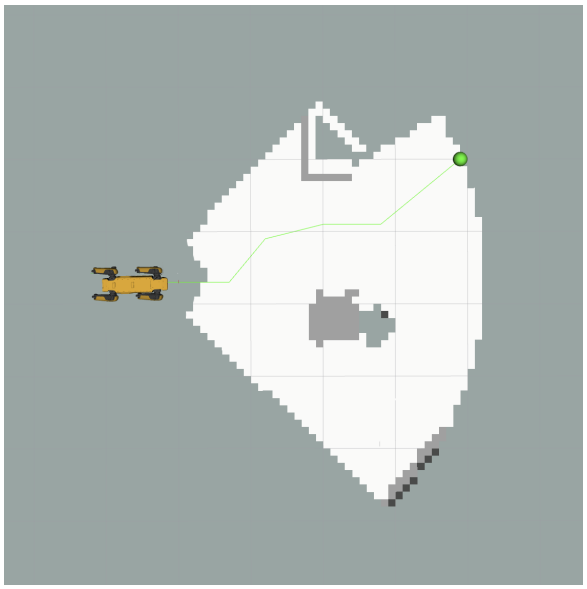
Figure 10 shows an example of the result of how the RRT* node looks like. Where the green line is the path created by the RRT* Algorithm and the green point is the goal position where Spot will move to while following the path. Figures 10a, 10b and 10c show successful runs of the RRT* algorithm. However, not all runs became successful as can be seen in figure 10d. Somehow a path is made through an obstacle in the occupancy grid.



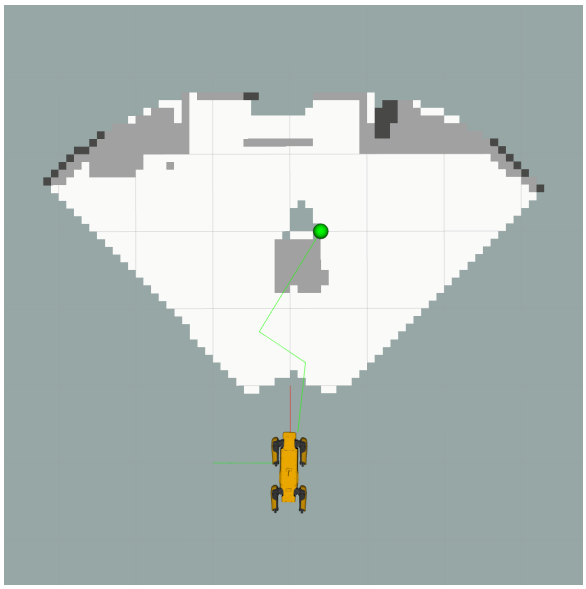
(a) Succesfull run RRT*



(b) Succesfull run RRT*



(c) Succesfull run RRT*



(d) Unsuccesfull run RRT*

Figure 10: Results of the RRT* Path

4.3.2 Exploration

Spot being able to autonomously explore an unknown environment is a fundamental problem in the field of robotics [31]. Autonomous exploration has the possibility to leave humans out of the loop and for Spot to be a working system on its own. Together with the RRT-Path Planning node the Exploration Node is responsible for the function **F3.1: Generate path for exploring** as this node creates the position to explore and the RRT* computes the path towards this position.

Motivation

To achieve this the team has chosen to implement the Frontier-Based Exploration Algorithm based on an existing GitHub project [3]. This repository makes use of gradient to determine valid exploration directions. We chose this methodology due to the simple underlying concept and optimization possibilities.

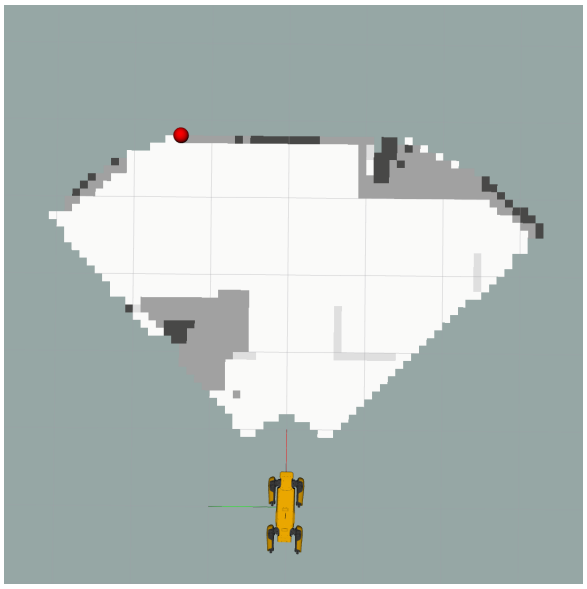
Technical Details

This algorithm uses Frontiers to explore the unknown space in the environment. Frontiers are used to describe the boundary between known and unknown space. While moving to this frontier Spot obtains new information about the environment. By repeating this process Spot can map the unknown environment. How this node finds a position on the frontier works as follows:

1. The node receives a 1D occupancy grid with the value -1 representing an unknown cell and values 0 to 100 representing the probability of the cell being occupied where 0 represents an unoccupied cell and 100 represents an occupied cell.
2. The 1D occupancy grid is transformed into a 2D numpy array.
3. After multiple transformations, the image gradient of the grid array is computed using Roberts Cross convolutional kernel.
4. From the image gradient cells are found above a certain threshold that ensures an unknown cell is picked and above a certain threshold in the grid to ensure an unoccupied cell is picked.
5. One of these cells is picked at random and sent to the RRT-Path Planning node.

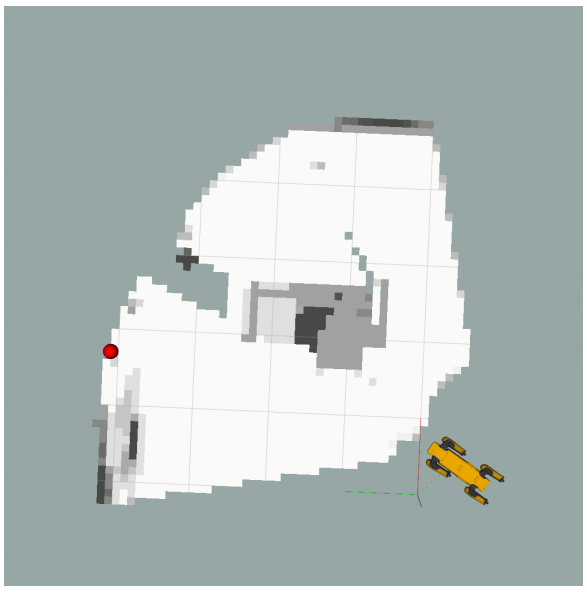
Recorded Behaviour

Figure 11 shows multiple results of the algorithm. The red dot represents the goal position Spot needs to walk to. Figure 11a, 11b and 11c show successful runs of the algorithm whereas 11d shows an unsuccessful run. This is due to that the algorithm picked an unknown cell in the grid because it was unable to see behind the obstacle.

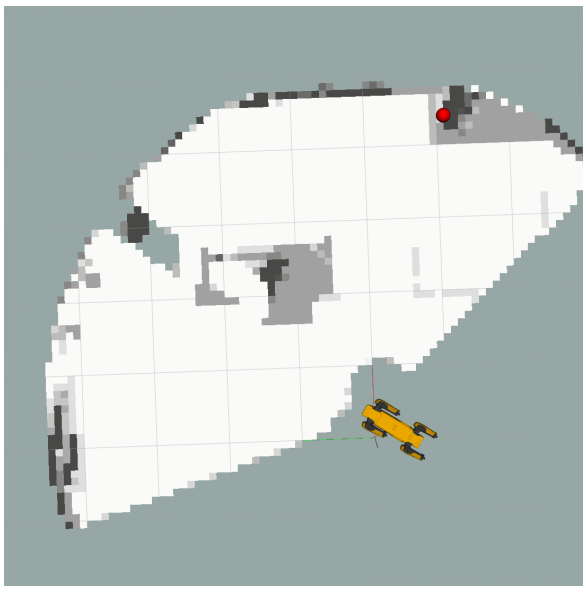


(a) Succesfull run of Frontier Exploration

(b) Succesfull run of Frontier Exploration



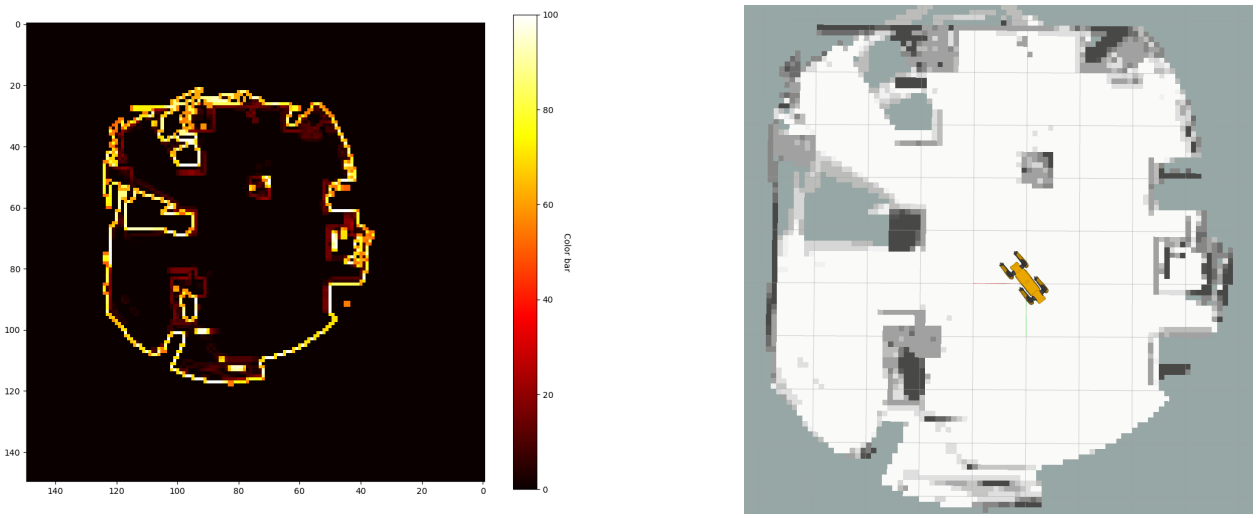
(c) Succesfull run of Frontier Exploration



(d) Unsuccesfull run of Frontier Exploration

Figure 11: Results of the Frontier Exploration

To visualize the image gradient a cost map can be plotted. Figure 12 shows this result where 12a shows the cost map to the occupancy grid shown in 12b. In this visualization it is clearly visible where the gradients between the grid's values are. Points are sampled where the cost is high in the cost map and where the cell is unoccupied in the grid.



(a) Costmap of environment

(b) Occupancy grid of environment

Figure 12: Costmap Frontier Exploration in occupancy grid

4.4 Motion Control

This subsection describes the implementation of the Spot movement towards a given goal using the Motion Controller node. The primary objective of this functionality is to enable Spot to navigate towards specific goals within the environment. The Motion Controller node effectively fulfills the high-level functionality of **F4: Motion Control**.

Motivation

At the beginning of the project, it became clear that there was no need for the Motion Control specialization as Spot could already move on itself given a goal. This function however was not implemented in the simulation of Spot and therefore required the team to implement some sort of motion control solution. The motivation behind introducing the Motion Controller node lies in the need to enable Spot's navigation within the environment. By implementing this functionality Spot can effectively move towards a specified goal.

4.4.1 Motion Controller

The implemented Motion Controller is a simple way for Spot to move in the environment. The node listens to the position of Spot in the OccupancyGrid and receives a goal for Spot in the same grid. By sending velocities commands to the driver Spot is able to reach the goal and thus implements function **F4.1: Move towards goal point**

Technical Details

The controller first calculates the yaw between the goal and Spot's current orientation. It does this using the following formula:

$$yaw_{goal} = atan2(y_{goal} - y_{current}, x_{goal} - x_{current}) - yaw_{current} \tag{1}$$

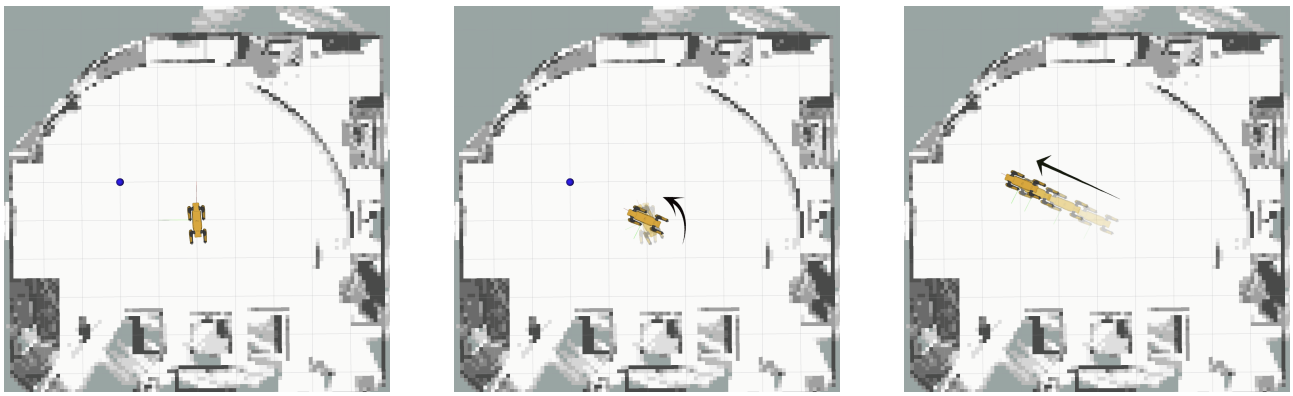
While y_{goal} is greater than a given tolerance, which can be specified, the Motion Controller sends an angular velocity to the driver, which can also be specified. In the current version of Spot the angular tolerance is set to 0.1 radians and the angular velocity to 0.15 rad/s.
Similarly, Spot moves towards the goal once the orientation is set. By calculating the distance to the goal using:

$$distance = \sqrt{(x_{goal} - x_{current})^2 + (y_{goal} - y_{current})^2} \tag{2}$$

While the distance is greater than the tolerance the Motion Controller sends a linear velocity to the driver. Currently, the distance tolerance is set to 3 grid cells and the velocity to 0.15 m/s. If necessary, Spot can use the first while-loop to obtain a certain orientation to the goal once it is within distance. If during either the first or the second while-loop the emergency stop is activated, the loop will be broken for safety purposes.

Recorded Behaviour

Figure 13a, 13b and 13c visualize how Spot follows the Motion Control. First it receives the goal position, then rotates is base towards it and finally moves towards the position until it is reached.



(a) Goal received (b) Rotating to goal (c) Moving to goal

Figure 13: Motion Control visualized

4.5 Human Interaction

One crucial aspect of integrating robots into healthcare settings is ensuring effective and intuitive human-robot interaction. This chapter focuses on how the developed solutions enhance the experience between nurses and SPOT, to facilitate seamless communication and mission assignment. To achieve this, two communication channels are implemented: a graphical user interface (GUI) that can be embedded in a bracelet and a pure speech interface. These communication channels are responsible for high-level functionality **F5: Human Interaction**

4.5.1 SPOT Control Panel

The GUI provides nurses with a user-friendly way to interact with SPOT, while the speech interface enables hands-free communication for convenient operation for example when the hands are not free to use the GUI. This approach aims to empower nurses, allowing them to assign missions to Spot and stay informed about its activities without being bound to a traditional computer interface. The GUI fulfills function **F5.2: Display Graphical User Interface**.

Motivation

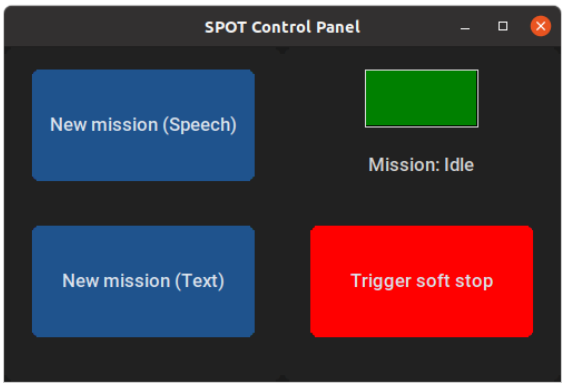
The use of a smartwatch to implement the GUI was chosen mainly due to user-friendliness, portability and convenience, real-time monitoring, and the potential to implement the smartwatch in a broader ecosystem of smart devices in the healthcare facility. It can therefore enable faster digitalization of the care sector.

Technical Details

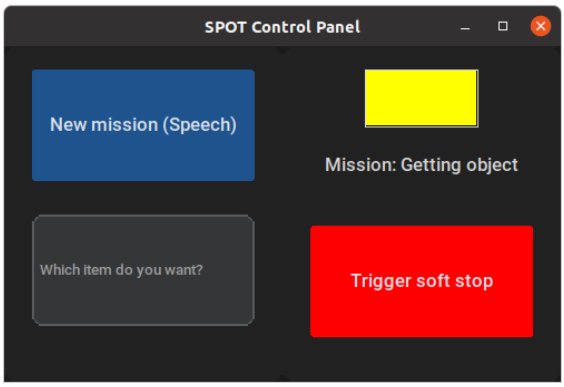
The implemented GUI (SPOT control panel) provides a very simple and intuitive way of specifying new missions, monitoring SPOT, and triggering the emergency stop. Since the control panel is operated on a smartwatch the design is simplistic and only consists of 4 different tiles, that change over time depending on the robot's state and the previous user input. The main library that the GUI is written in is `tkinter` and `customtkinter` and it triggers ROS actions.

Recorded behaviours

The GUI at startup can be seen in figure 14a. The mission status is idle and displays a green rectangle.
Figure 14b shows the GUI while SPOT is on a mission called "Getting object" and therefore the rectangle is yellow, which means occupied. In the bottom left, the text entry field is active. This happens after the operating nurse clicks on the button "New mission (Text)". Here s/he can specify a new object to find and therefore trigger the ROS action "Approach" with the given goal being the specified object. The location of the object will be queried from the database that the Detection nodes create. In this database, object names and their corresponding locations are stored.

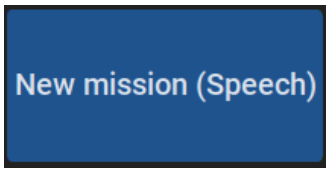


(a) after startup

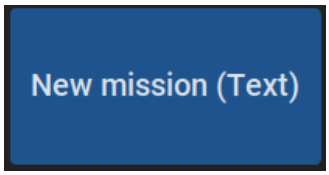


(b) on mission and text entry active

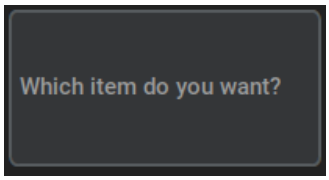
Figure 14: SPOT Control Panel in two different modes



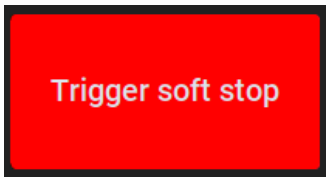
Input	Output
Click on button	Triggers ”ConversationAction” with goal ”give mission”



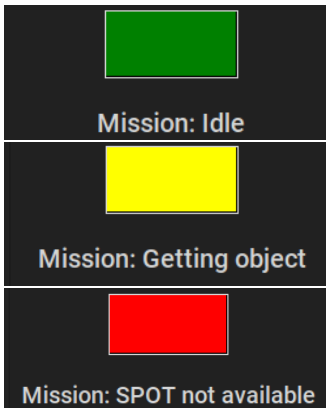
Input	Output
Click on button	Changes the bottom left tile to display text entry box



Input	Output
Typed in string	Triggers ”ApproachAction” with typed text as goal



Input	Output
Click on button	Triggers service ”spot/estop/gentle” to immediately stop all actions



Input	Output
Messages on topic /spot/mission_status	Display the current mission and the corresponding mission status • **Green**: ”Idle”, ”Available” • **Yellow**: ”Mapping”, ”ApproachGoal”, ”Conversation”, ”PickingItem” • **Red**: ”Unavailable”, ”Failed”

4.5.2 Conversation node

Additionally, a speech node was incorporated that enables voice commands to trigger SPOT's missions. The speech node can be activated both through the GUI on the bracelet and the underlying state machine that manages the autonomous activities. This speech node is responsible for the function **F5.1: Conversation**.

Motivation

The reason why the node relies on Google's speech recognition engine, is a fairly low word error rate (determined by a study in 2020) with 20.63% [7]. This outperforms other speech engines on the market, like IBM and Wit [7]. To assess the quality of speech recognition, the word error rate (WER) is a commonly used criterion [1]. It is defined in Equation 3:

$$WER = \frac{(S + D + I)}{N} \tag{3}$$

N = Total number of words
S = Substitutions (captures another word instead of the right one)
D = Deletions (words that are not captured)
I = Insertions (includes words that were not spoken)

The current and observed performance in the implementation is significantly better than what this paper found 2 years ago. Additionally, since the whole conversational structure is kept fairly simple (mostly yes/no) a significant performance increase could be achieved by the remapping of words. This method takes the 20 most common ways of saying yes and the 50 most common ways of saying no and remaps them to yes and no respectively in the answer string.

Technical Details

The conversation node consists of two subparts, namely speech recognition (speech-to-text) and speaking (text-to-speech). There are two possible conversation types:

1. **Giving a new mission**
   - Triggered from GUI or state machine after mapping
   - Starts with SPOT asking, if there is assistance needed and continues with specifying which item to get.
   - After completing the conversation, the ROS action "Approach" with the given goal being the specified object from the conversation.
   - The location of the object will be queried from the database that the Detection nodes create. In this database, object names and their corresponding locations are stored.
2. **Confirming successful mission**
   - Triggered from state machine after the item was picked up and delivered
   - Starts with SPOT asking, whether the correct item is currently in the gripper, and after confirmation, the item is delivered.
   - When an incorrect item is in the gripper, a loop to specify which item to get instead is triggered
   - After completing the conversation, the current mission will be finished

Both conversation types implement several loops that make the conversation robust, against misunderstanding or not recording answers by asking the user for confirmation or repeating the spoken answer. The text-to-speech functionality is implemented using the espeak module. The spoken text sounds slightly mechanic but is error-free and therefore enables an entirely conversation-based communication with SPOT.

Recorded behavior

In the below figures, the simplified flow of the two different conversations that happen in the conversation node are shown. The fallbacks due to errors are not shown in the flowchart, but can be found in Table 24. Each time one of the errors occurs, the corresponding system response is given and the previous question is repeated. This flow happens exactly like that in the terminal and in spoken language, but for visualizing purposes, these flowcharts were used. Figure 15 shows the flow of the conversation type ”give mission” and 16 the flow of conversation type ”confirm mission”.

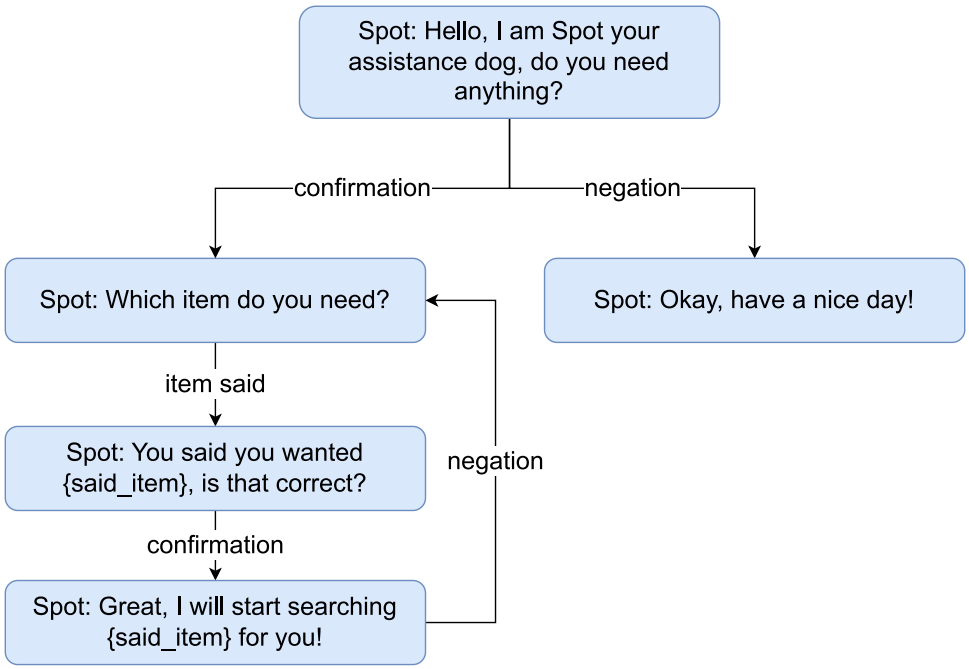


Figure 15: Flowchart of conversation ”give mission”

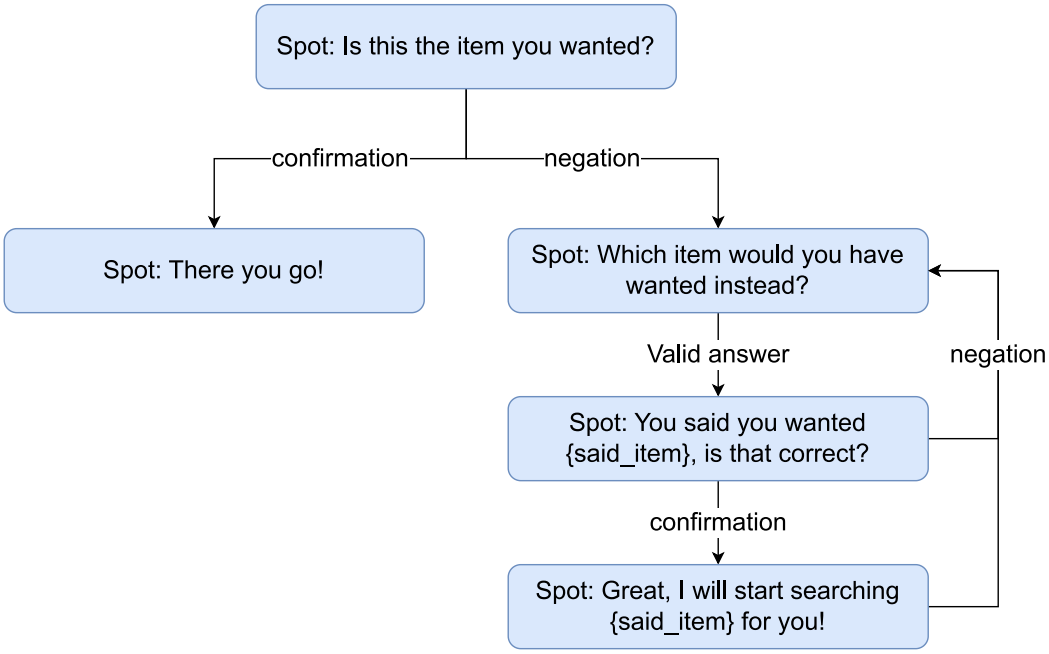


Figure 16: Flowchart of conversation ”confirm mission”

The following table provides an overview of typical errors in transcribed operator input and the corresponding answers. After giving the system answer, Spot will repeat the previous question to make the conversation robust to the below-mentioned errors.

Transcribed operator input	**System answer**
Empty string	”I’m sorry, I didn’t hear your response.”
Answer that does not contain one of the 40 remapped words to ”yes” and ”no” in a yes/no question	”Please respond with a ’yes’ or ’no’.”
Object specified that is not in the list of detectable objects	”Sorry, the item you requested is not a valid mission.”

Table 24: Typical errors during conversation

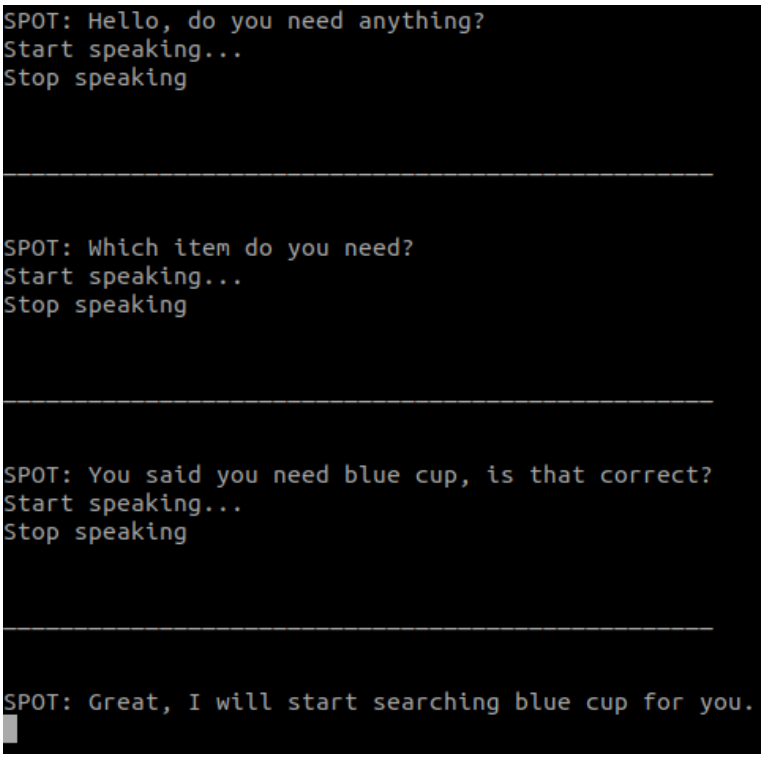


Figure 17: Example of successful conversation in Linux Terminal

4.6 Executive layer

The executive layer of the developed solution connects all the nodes, possible services and action servers that are described above in Chapter 4. Thereby it enables the coordinated execution of tasks and proper system behavior by handling different states and their respective transitions.

4.6.1 State machine

It is important that Spot has a behavior model that decides when the system should transition to the next state. To achieve this the team has chosen to implement a State machine node. This node is used to handle different states and their respective transitions. By organizing Spot's behavior into discrete states and controlling their transitions, the state machine enables robust decision-making and task execution. The State Machine is responsible for a function **F0: System**.

Motivation

Using the `SMACH` library in ROS offers significant advantages for developing state machines [29]. SMACH enables a modular and hierarchical design, allowing complex behaviors to be broken down into reusable states. Its declarative syntax enhances code readability and the `SMACH Introspection Server` simplifies debugging. With built-in mechanisms for managing state transitions and handling outcomes, `SMACH` enabled the development of a robust state machine.

Technical Details

The implementation of the state machine node makes use of the `SMACH` library which is a task-level architecture for creating complex robot behavior [29]. It is based on the activity diagram in Figure 3 and mirrors that behavior as closely as possible. The implemented states and their transitions and conditions can be described as follows :

1. `INITIALIZATION`: All the required nodes and servers are launched and the GUI is booted on the smartwatch/bracelet. If errors occur, the `ESTOP` is triggered and the state machine terminates with "failed". If `INITIALIZATION` was successful, the transition to `MAPPING` happens.

2. `MAPPING`: The final goal of that state is to create a 2D occupancy grid of the environment and build up a database containing the detected objects and persons and their respective locations. To ascertain the completion of the `MAPPING` process, a predefined threshold is evaluated that measures the rate of change in the map per scan. If that threshold is reached, the transition to the state `IDLE` is made when there is no mission given yet. As long as the threshold is not reached, `MAPPING` will be repeated.

3. `IDLE`: Spot remains idle and sits down on the ground until a request is given either by using spoken natural language in `CONVERSATION` or the text entry on their bracelet. In both cases, Spot will transition to `APPROACH_ITEM`.

4. `CONVERSATION`: activates the conversation server to give a new mission and if an object is specified, the state `APPROACH_ITEM` is triggered. If the conversation fails three times in a row, the `ESTOP` is triggered.

5. `APPROACH_ITEM` generates and executes a path to the specified item based on a query of the database that was created during mapping. If Spot reached the goal, the next state is `PICK_ITEM`, if the item could not be approached, the mission is aborted.

6. `PICK_ITEM`: requests the operator to pick up the object Spot is standing in front using the controller. If the item picking fails, it will be executed again. Once it fails 3 times in a row, the mission will be aborted and the `ESTOP` will be triggered. After successful picking, the state `APPROACH_PERSON` follows.

7. APPROACH_PERSON: generates and executes a path to the person that requested the item to be delivered. If Spot reached the person, the next state is CONFIRM_MISSION, if the item could not be approached, the mission is aborted and the ESTOP is activated.

8. CONFIRM_MISSION: determines in a conversation whether the item brought is correct. If that is the case, it executes PLACE_ITEM If that is not the case, the item will be returned to its original location through the execution of the state APPROACH_ITEM and placed back.

9. PLACE_ITEM: requests the operator to place down the object Spot is holding using the controller. If the item placing fails, it will be executed again. Once it fails 3 times in a row, the mission will be aborted and the ESTOP will be triggered. After successful placing, the state IDLE follows for Spot to take on new missions.

10. ESTOP: triggers the ROS-service /spot/estop/gentle in order to abort all current actions and make Spot sit down and shut down all motors. After executing that state, the state machine will terminate with the status "failed".

Recorded behavior

Figure 18 shows the generated graph when using smach_viewer to open the introspection server of the running state machine. In the graph, the system is currently in state CONVERSATION. A description of what each state does can be found in the paragraph above.

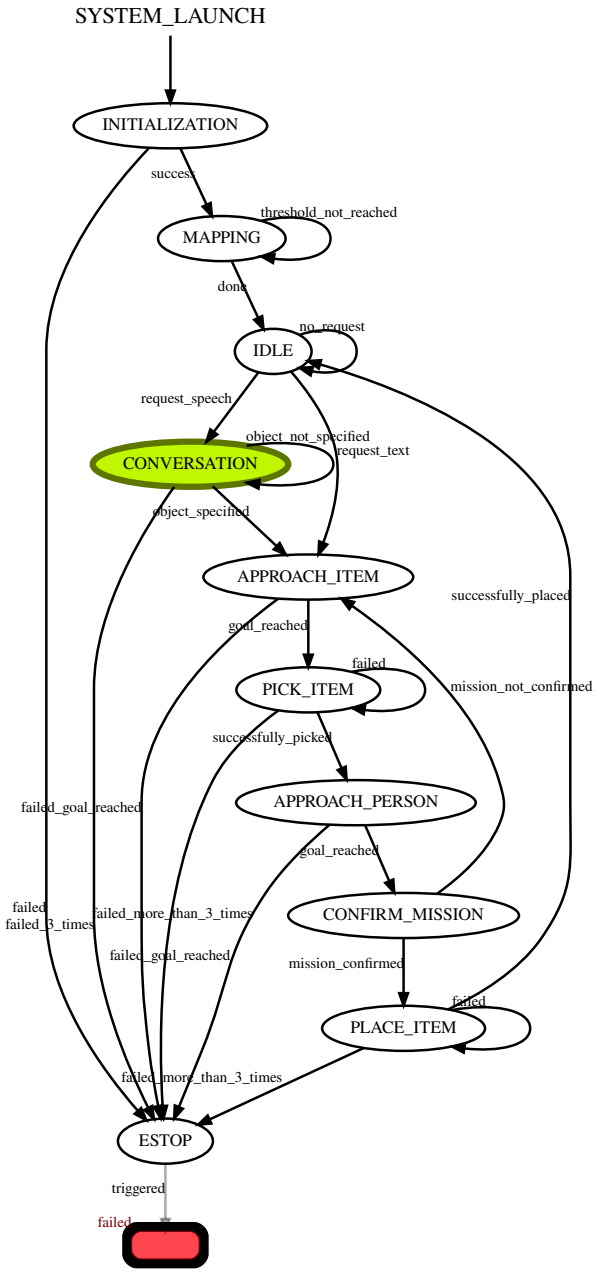


Figure 18: State machine visualized using smach_viewer

5 Summary of Real Robot Results

This chapter discusses the results obtained from the real robot and presents a debugging report highlighting the differences between the simulation and the actual robot. While the simulation provides valuable insights, it does not always translate directly to real-world implementation. Certain aspects that function well in the simulation may not work as expected on the physical robot, and conversely, there may be components, such as the arm, present on the robot that are not represented in the simulation. The following section provides an overview of the obtained results from the real robot and offers a comprehensive analysis of the divergences observed.

To see a video of how the real Spot could work this demonstration video was made [7].

5.1 Functional graph

In this section the differences between the functional graph of section 4.1 (obtained from the simulation) and the functional graph of the recorded robot behavior are outlined. To present the functional graph of Spot the full rqt-graph has been split up into the two figures 19 and 20. These two figures show the rqt-graph of the real Spot. All the nodes and topics made by the group and each node and topic is colored corresponding to their high-level functionality from 3.3. These are also used to show the similarities between the simulation and the Spot. The full rqt-graph of the Spot can be found in appendix E.

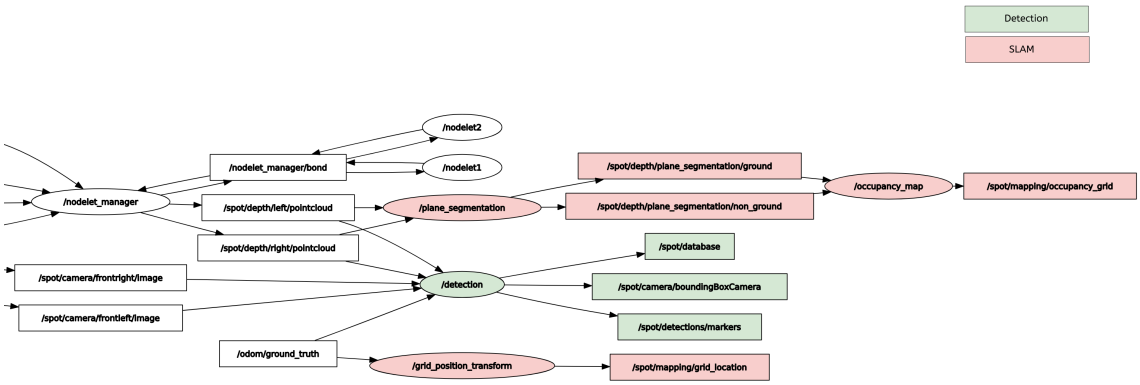


Figure 19: rqt-graph for SLAM and Detection for Spot

When comparing the rqt-graph for Detection and SLAM in both the simulation and the Spot robot, it is evident that there are no differences between them. The nodes and topics depicted in Figure 4 and Figure 19 are identical. This indicates that no modifications were necessary to enhance the real robot's performance regarding the nodes and topics associated with **F1: Detection** and **F2: SLAM**.

[7] Demonstration video: `https://www.youtube.com/watch?v=4hbgdUF2sGU`

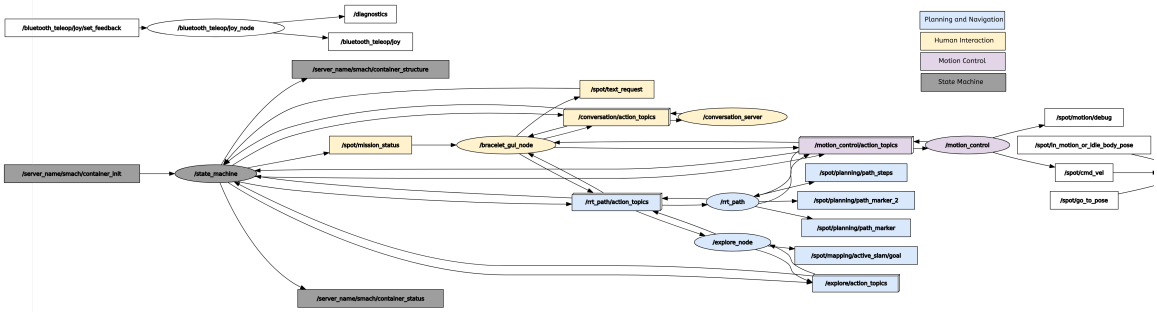


Figure 20: The rqt-graph for Planning and Navigation, Human Interaction, Motion Control and State Machine for Spot

Comparing the rqt-graphs for Planning and Navigation, Motion Control, Human Interaction, and the System, in both the simulation and the Spot robot, it is apparent that there are no discrepancies between them. Figure 5 and Figure 20 exhibit identical nodes and topics. Consequently, no adjustments were required to enhance the real robot's operation with regards to the nodes and topics corresponding to **F3: Planning and Navigation**, **F4: Motion Control**, **F5: Human Interaction**, and **F0: System**.

In the simulation, Spot is shown with the `/gazebo` node, while in the real world, Spot is represented by the `/spot/ros` node. Additionally, it is noticeable that the simulation does not include a robot arm, which is present in the real robot. These variations in nodes and topics are highlighted in yellow and can be observed in Appendix E in Figure 31 for the Spot in simulation and Figure 32 for the Spot in the real world.

5.2 Recorded behaviour

In this section, the recorded data from the testing sessions is presented to demonstrate the functionality of the code on the real robot, the Spot.

Perception
The cameras of the robot are RGB cameras which means that they can output color images but accessing the camera's through the associated nodes will output grayscale images. Running the detection algorithm on it then yields the result as shown in Figure: 21a. Spot also has a node that outputs the image from the arm: `/spot/camera/hand_color/image` Subscribing to this node instead and running the detection algorithm yields the output as can be seen in Figure: 21b

(a) Detection algorithm applied to the front camera of Spot

(b) Detection algorithm applied to the arm camera of Spot

Figure 21: Detection algorithm applied to the camera images of Spot

Point cloud information needs to be obtained for it to be used by other nodes. The result of the real recording of the point cloud information can be found in Figure: 22. A person standing in front of Spot is visible as well as the railing behind the person.

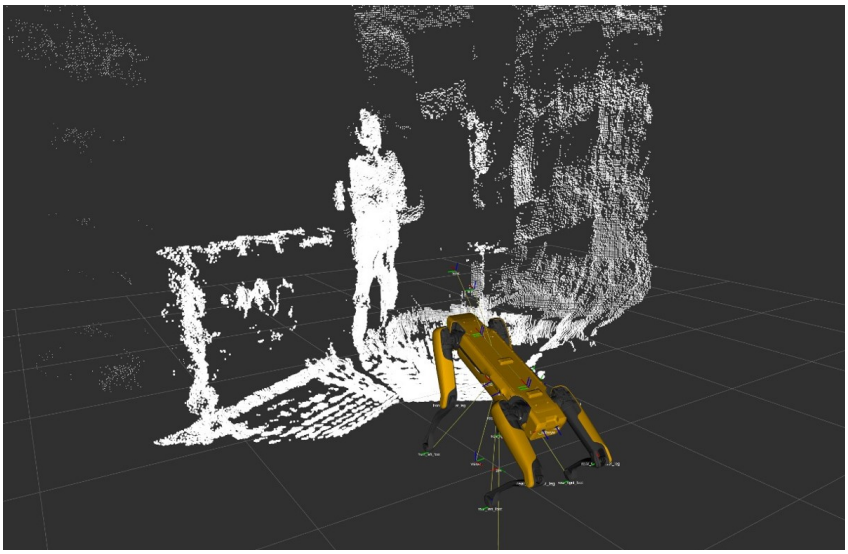
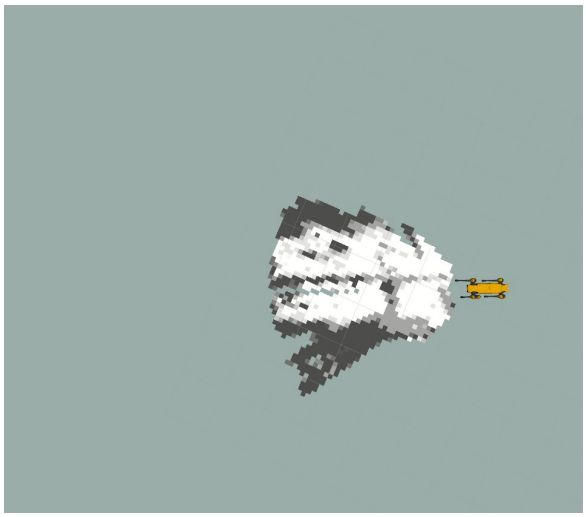
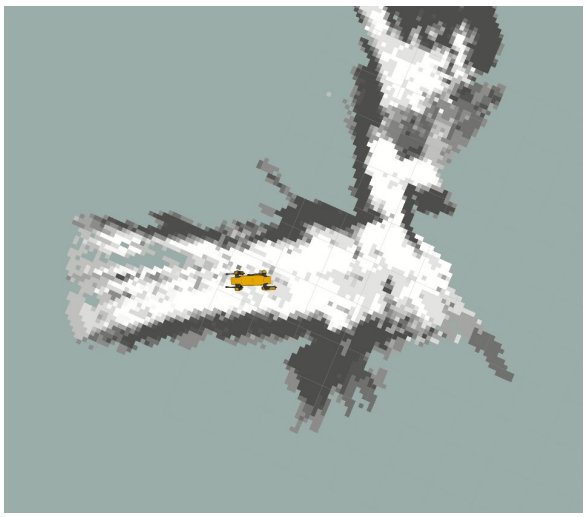


Figure 22: The result of combining the front and right depth cameras as pointcloud points

These points are used for example by the slam algorithm to build a map. Figure 23a shows the initial state of the map in the real world and Figure 23b shows the map after Spot has walked around the room. Due to our own implementation of the SLAM algorithm, the resulting occupancy maps had minimal noise.



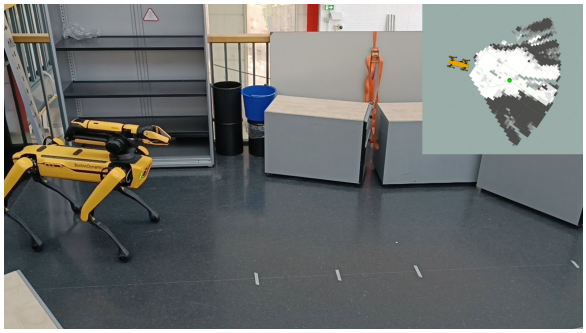
(a) Map build from the initial position of Spot



(b) Map after Spot has walked around the room

Figure 23: SLAM working on the real robot

Planning and Navigation
To show the capabilities of the real system all the Planning and Navigation nodes are combined with footage of Spot. Figure 24 shows the planning having executed and Spot following the path starting from its initial position 24a to its final position 24b.



(a) Initial position of Spot with the simulation in the top right corner that shows the path

(b) Final position of Spot after it has executed the path found

Figure 24: Spot autonomously finding a path and executing the path

Human Robot Interaction
Because the nodes related to human-robot interaction are all self-sufficient systems they did not have different recorded behavior with respect to chapter 4.

State machine
After applying the fixes that are described in the Debugging report in chapter 5.3 the major parts of the state machine were tested successfully. An entire frontier exploration run was performed, while simultaneously the object detection algorithm was running. A valid path was planned to the required object after specifying that object in the GUI. The authority for the teleoperation of the arm was shifted successfully and also the Estop was triggered from the GUI to prove the fail-safe mechanisms. Overall, the behavior was very similar with respect to chapter 4.

5.3 Debugging report

Most problems arose in the perception branch, due to the issues with the differences between the simulation and the real world so the report will focus on perception debugging. The state machine debugging is also documented.

Detection
The largest difference between real life and simulation came from the detection algorithm. This section will go into the bugs encountered and the way they were solved.

RGB camera outputs only grayscale images:
The simulation uses RGB encoding for its images. This essentially means that a single image is stacked three times for each color Red, Green, and Blue, but the real robot only provided a grayscale image. The issue arose from the Yolo algorithm being trained on RGB images and thus did not work with grayscale images.
The way this was debugged was by first changing the config file of the Yolo algorithm to accept only one channel but this did not work as it did not provide an output of the detection.
The second thing that was looked at was turning the grayscale image from the ros topic into an RGB image. This should be possible through the use of the spot image service but it was decided to first look for easier solutions as the setup required is non-trivial. [6]
Finally, the solution was found in 'creating' an RGB image from the original image by stacking it three times. This way yolo was able to accept the image and draw bounding boxes.

Detections were not good enough:
When the detections worked it became clear that yolov3 worked better in the simulation with respect to the real world. The problem was easily fixed by upgrading from yolov3 to yolov7.

Arm camera provides better imaging point compared to front-facing cameras:
There is no arm in the simulation so this could not have been tested initially but it was found that the arm camera provided a better vantage point with respect to the front-facing cameras. This is because the front-facing cameras are pointing down to the ground and to the side. This means that if a person is standing right in front of Spot a detection might not happen.
Because the arm is not in the simulation a fixed coordinate transform cannot be found using `tf.TransformListener()` this means that the dimensions have to be recorded from the camera to the baselink frame of the robot and a manual transform has to be carried out. Due to time constraints, it was not possible to implement this and thus the decision was made to stick with the front-facing cameras.

Mapping
When looking at the differences between the simulation and the real robot for mapping the environment. The following problems were encountered.

Pointclouds where rotated:
In the simulation the pointclouds were rotated in the other direction than the real robot. This could be easily fixed with rotating the pointclouds in the other direction. The pointclouds that were rotated in the other direction for the real robot can be seen in Figure 25

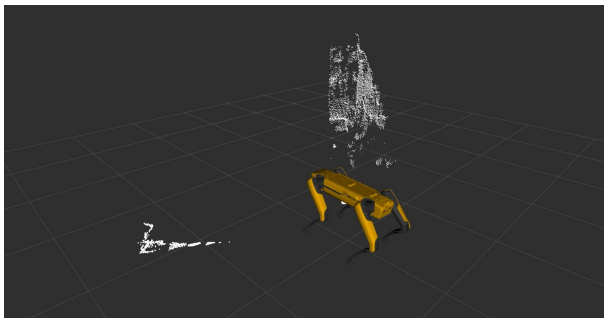
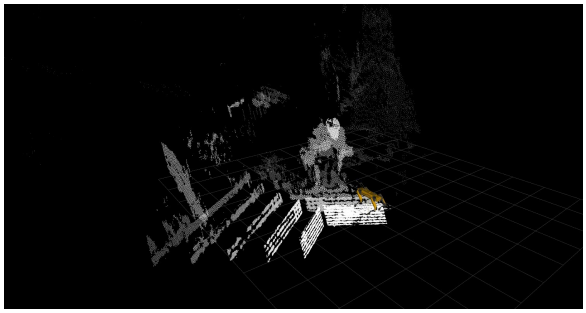
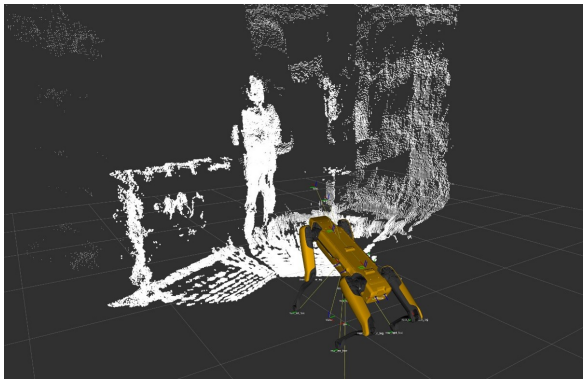


Figure 25: Rotated point clouds for the real robot

Pointclouds visualization:
When trying to visualize the pointclouds on the real robot. It was noticed that the nodes that were used in the simulation did not work. To solve this problem the `image_proc` node is used. [19] After using the `image_proc` node the pointclouds could be visualized on the real robot. This can be seen in Figure 27.



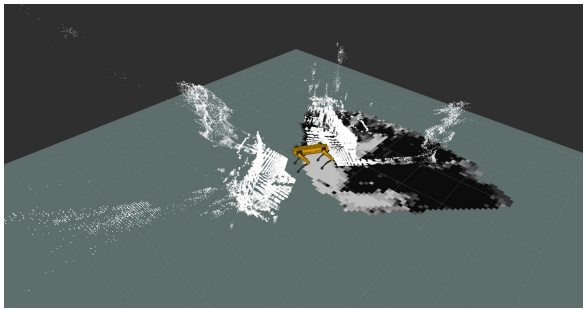
(a) Pointclouds when visualized with the simulation node



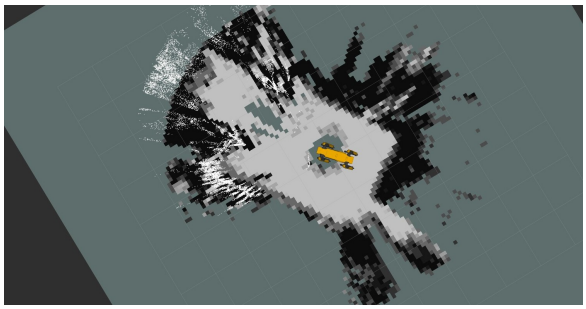
(b) Pointclouds when visualized with the image_proc node

Figure 26: Pointcloud visualization for the real robot

Pointclouds did not update when rotating:
When updating the map it was noticed that the pointclouds were stuck between the same 45 degrees frame even when rotating. The problem was that the frame could still translate, but was rotationally fixed. To solve the problem the frame was changed to baselink and the z height was adjusted to the height of the real robot. This fix allowed the pointclouds to update while rotating the real robot.



(a) Pointclouds are not updated when rotating the real robot



(b) Pointclouds are correctly updated when rotating the real robot

Figure 27: Pointclouds visualization when rotating the real robot

State Machine
Initially, there were discrepancies in the naming of frames and topics between the simulation and the real robot, which led to difficulties in the proper functioning of the state machine. To address this issue, we employed a hotfix approach by modifying the frames to align with the real robot on one machine, while the other machine contained the state machine code designed for simulation. Both machines were then connected to the same `roscore` running on Spot, our platform. This configuration allowed for synchronization and communication between the modified machine, which controlled the real robot, and the machine running the simulation-specific state machine code.

5.4 Future Work

In order to build upon the achievements of this project and explore further possibilities, future work is valuable. This subsection briefly outlines potential areas per specialization for future research.

Perception
One important aspect of machine perception within this project is the creation of the occupancy map. As mentioned before, our own implementation of the SLAM algorithm is used to create such a map, instead of the standard ROS package. This gave us the ability to fully develop this algorithm for Spot and thus achieve good results with minimal noise (see section 5.2). However, our implementation is not yet optimized for computational efficiency and does not consider the dimensions of the robot, meaning that the initial pose of the robot is seen as undiscovered space, see figure 27b.

The detection node works well but situations can arise where the output is not as robust. A common situation is when the detection algorithm detects an object that has a hole in the middle, for example a chair. What happens then is that the point that is sampled is sampled from the middle of the boundingbox, through the chair and is assigned some point behind the chair. If Spot then rotates the same detection will sample a different point behind Spot that is far away enough from the previous sampled point and add this to the database, this creates 'ghost' detections.

Another point of future improvement could be the tracking of objects and persons. Right now the system works well in a static environment but if objects get moved or persons walk away this will not be reflected in the database.

Planning and navigation
For path planning there are various ways that could potentially improve the current implementation. Other variations on RRT* or another algorithm that could have been examined are:

- RRT*-Smart [9]: which aims to accelerate the rate of convergence and to reach a near optimum solution much faster with intelligent sampling.
- MOD-RRT* [18]: which can improve the path by replanning the initial path where both the path length and smoothness are considered.
- A* [34]: which efficiently finds the shortest path by combining Dijkstra's algorithm and a heuristic estimation.

Another future work for path planning is to implement steering functions in combination with future work on Motion Control. With this, Spot will walk through the environment much smoother. Instead of rotating and then walking forward, Spot could walk and rotate simultaneously. One algorithm to examine that could work for this purpose is Theta*-RRT[17], which can find a non-holonomic path.

For exploration, as mentioned in section 4.3.2, the concept of gradients is used to determine valid exploration directions. This method worked well both in simulation and on the real Spot, however, there is still room for improvement. For example, the transformations used to create high gradients along the borders of free and undiscovered space are not fully optimized. This resulted sometimes

in high gradients along non-optimal exploration directions. To mitigate this, the transformation and sampling threshold used should be further optimized.

Human robot interaction
There are three potential ideas for expanding on the implemented interaction package.

Firstly, a desktop/tablet application could be developed that provides a comprehensive dashboard for monitoring and managing the solution. The dashboard could display extensive real-time data, for which the smartwatch GUI does not provide enough screen space. For example, robot location, movement status, battery level, pending tasks, and performance metrics. The dashboard could also include advanced map visualizations like an interactive map to provide a more detailed view of the hospital environment. This could include the ability to track the delivery robot's movement in real time, display delivery routes, and highlight specific locations or zones within the hospital. Such visualizations would improve situational awareness and enable better coordination and optimization of missions.

Secondly, The conversation node could leverage a custom ChatGPT implementation to replace hardcoded conversations, thereby enabling more dynamic and interactive communication. By integrating ChatGPT, which is a state-of-the-art language model, the speech node could generate natural and dynamic conversations, fostering a more immersive and engaging human-robot interaction experience. To implement this, the speech node would need to utilize the ChatGPT API. The node would receive the input speech or text from the user and pass it to the ChatGPT model for processing. The answer could be constrained by priming the model (i.e. forcing it to always answer in a certain way). The hardcoded remap of the most common 50 words of saying yes that is currently implemented could be exchanged by using ChatGPT to determine whether the given answer is more likely confirmative and thereby making the possible answers much broader.

Thirdly, a directional light system for communicating Spot's plan could prove beneficial. For example, the light system can project a focused beam or emit a distinct pattern in the direction the robot plans to move next. This allows users to anticipate the robot's trajectory and adjust their own movements accordingly, avoiding potential collisions or misunderstandings, adding a layer of safety to the system.

State machine
A valuable lesson learned during the project was that it might have proven beneficial to implement the architecture of the state machine first instead of first writing and testing the individual nodes. Thereby, it would have been easier to connect all the nodes in the end. This would have reduced the need to change/rewrite parts of many nodes in order to make them suited for autonomous missions.

6 Individual Reflection

This last chapter sets personal development goals for each team member and goes into the soft skills obtained during the project.

6.1 Guido

Learning Goals
Guido Dumont has two main learning goals, one on the technical side and one on the social side. Both goals are defined below using the SMART framework:

Technical development goal:
Specific: Gain proficiency in building, designing, and working within ROS.
Measurable: Make at least one working ROS node from scratch using Python and/or C++.
Achievable: Online ROS tutorials are readily available and free, and they can dedicate regular time to building and testing the new ROS package.
Relevant: Gaining more skills within ROS will help me in the future to design robotic applications more efficiently.
Time-bound: This learning goal should be reached within the time spent of the project.

Social development goal:
Specific: Explain my thoughts and ideas more clearly to the group.
Measurable: Participate in every team meeting and ask for feedback from group members to evaluate the progress.
Achievable: Focus on actively explaining my thoughts and ideas in every meeting.
Relevant: Reaching this goal will improve my collaboration skills and will hopefully improve the final results of projects.
Time-bound: This learning goal should be reached within the time spent of the project.

Peer Feedback Received and Peer Feedback Given
First feedback round:

- Give others a task to do so that you are not doing everything yourself.
- You could maybe take some more time to take your group member through this process and explain what you have done.
- Sometimes I have the feeling you go a bit too fast which makes it difficult for me to keep up.
- Sometimes a bit fast in explaining his (advanced) solutions, since not everyone is as far as he is.

Second feedback round:

- Sometimes communication can be a bit direct. This could be improved by listening more to someone before responding.
- Maybe ask for help a bit earlier.
- I can't really think of a good improvement point, you are doing well! :)
- Hard to think of something, to be honest, keep up the good work.

Looking back on my learning goals, I am pleased to say that I made significant progress in developing my ROS skills in Python. Throughout the project, I put many hours into building and testing various important nodes for the project, which allowed me to gain proficiency in designing and working within ROS. I successfully achieved the specific and measurable aspect of my technical development goal

by creating several working ROS nodes from scratch. However, I also didn't get the opportunity to explore ROS in C++. Therefore, in future projects, I aim to expand my knowledge by delving into ROS within C++.
On the other hand, the social development goal was a bit more challenging for me. Due to my enthusiasm and eagerness to contribute, I sometimes found it difficult to take the necessary time to actively explain my thoughts and work to others in the team. However, after the reminder during the first buddy check, I became more aware of this aspect and actively worked on improving these skills. During every meeting, I was constantly aware of my learning goal and tried behaving towards it by explaining everything I did and thought. This helped me improve my collaboration skills and also had a positive effect on the end result of the project in my opinion. The improvements I made with this learning goal are also visible in the feedback I received when you compare the two buddy checks.

Transferable Skills
Teamwork: Teamwork and collaboration skills were vital during this project. As mentioned before, I actively worked on better explaining my thoughts and ideas, which was certainly successful during the project. However, I think that my collaboration skills always improve within new teams and projects. Besides this, I also really enjoyed working with this team since the overall vibe and energy were very positive, which contributed to a nice end result.
Leadership: We as a group rotated the role of project leader every week, which gave me the opportunity to lead the group for two weeks. These were stressful weeks, but I managed them nicely in my opinion. The feedback I received on my leadership was also positive. However, I still think that I can improve my leadership skills when working with other (more challenging) teams and projects.
Entrepeneurial thinking: In the last week of the project I focused mainly on the final demo video and presentation. My goal and that of the team was to produce a high-quality demo/marketing video to convince the client about our project. We have achieved this by winning the first prize given by TNO. In my opinion, this was mostly due to our marketing video and presentation.
Strategic multidisciplinary problem-solving: This project gave me the opportunity to build a working robotic application from scratch, which gave me a lot of practical insides and definitely changed my problem-solving behaviour. Especially by thinking ahead and designing the code in a modular and flexible manner.

Agile, Specialist Roles, and Other Team Roles
In my opinion, the agile project workflow was not very well suited for this relatively short project. We rushed through everything due to the tight deadlines, which forced us to change many things around at the end of the project. Although this is the idea behind the agile workflow, it gave us a lot of extra documentation work, due to the constant changes. I would have worked better if there was more time in each sprint to really think ahead.
I really found it beneficial to give everyone a specialised role within the team. This gave everyone responsibility and motivation to deliver good work.
Unfortunately, the other, additional roles were not actively rotated during the project. Due to the tight schedule, it was more natural and beneficial to keep these roles mostly fixed.

Other Key Reflection Conclusions
Overall, I really enjoyed the project, despite the demanding workload. It gave me insides into what it takes to build a bigger robotics project within a team and on a real robot. This confirms my choice for Robotics and motivates me to work harder.

6.2 Jesse

Learning Goals
I have two main personal development goals, one is professional and one technical in nature.

Technical development goal:
Specific: Improve proficiency with ROS 1 by learning from online sources and building a functional ROS package that implements a new perception feature for the spot robot.
Measurable: By the end of the project, I will be able to demonstrate my improved proficiency with ROS 1 by successfully building and testing a new ROS package that adds a new perception feature to the spot robot.
Achievable: Online ROS tutorials are readily available and free, and they can dedicate regular time to building and testing the new ROS package.
Relevant: ROS 1 is the core framework used in the perception side of the robot, and improving my proficiency will contribute to the success of the project.
Time-bound: I will learn from online sources and build the new ROS package within the next seven weeks, and regularly test and refine the package throughout the project.

Professional development goal:
Specific: I want to improve my understanding of team dynamics and how to operate within a multidisciplinary team by regularly seeking feedback from team members.
Measurable: By the end of the project, I will be able to demonstrate my improved understanding of team dynamics by actively engaging in team discussions and applying this knowledge to contribute to the project's success.
Achievable: Seeking feedback from team members and reading articles are free and can be done at any time.
Relevant: Understanding team dynamics is crucial for effective collaboration in a multidisciplinary team and will contribute to the success of the project.
Time-bound: I will regularly seek feedback from team members and use the Buddychecks to obtain feedback

Peer Feedback Received and Peer Feedback Given
The feedback that was given:

- I should keep the group updated on the progress I made.
- I should speak up more during meetings.
- I should seek help more quickly when stuck with coding.
- When working from home I should more actively ask the group if I can help them with tasks.

I think all of this is good feedback and I agree with all points, I have tried after receiving the feedback to incorporate this. The feedback I gave was well received I think. Our group seems to handle giving and taking feedback really well.

Transferable Skills
teamwork: During the course this is the most practiced of the transferable skills. The team has a good dynamic, I think sometimes the team wants a bit too much for my liking but I tried to match this as best as possible.
leadership: During the project everyone took a leadership role on themselves for a period which is a nice learning experience. I do think the team members are very effective in the way they communicate and solve problems so a team leader was not really necessary.
entrepreneurial thinking: This was not really applicable to this project.
strategic multidisciplinary problem-solving: This proved to be quite difficult at the end I found. Because during the design of the state machine a lot of errors arose with the nodes, and effectively

communicating these errors to get them fixed can be quite a challenge.

I liked the fast-paced environment during code development the most and this is where my interest lie the most. I had no new strengths or weaknesses I identified during the course.

Agile, Specialist Roles, and Other Team Roles
I think the agile project management framework was a bit unnatural and forced during this project and I did not really like it. The time that was given for the asked end product was just not enough which forced a too short analysis phase and a too rushed development phase. As for the specialist roles, I think it was nice to be responsible for a particular piece of code and I was able to get it all done, with some help from the team.

6.3 Jakob

Learning Goals
I had three main goals, namely figuring out which area I am most interested in within robotics, what it takes to build an actual real-life application using hardware and thirdly, to practice my presenting techniques.

Technical development goal 1:
Specific: Figure out which area of robotics I am most interested in.
Measurable: Get a fundamental understanding of each of the areas and help to build each node.
Achievable: Since there are meetings each week, it will be possible to gain insights into all areas/nodes.
Relevant: Determining which area to focus on for later work but especially the internship is of high relevance for my future career.
Time-bound: This learning goal should be reached towards the end of the course.

Looking back, I think I really gained deep insights into all fields involved in this project, since next to writing my two human interaction nodes, I was also mainly responsible for the state machine. Taking on this job was a very good idea since it forced me to understand all the other nodes in detail. Nevertheless, I am still not entirely sure which area of robotics I am most interested in, since all of them have very interesting sides. If I had to choose now, I think I would go for either perception or path planning.

Technical development goal 2:
Specific: Understand what it takes to build a real-world application including hardware and the limitations simulations have.
Measurable: The final solution performance on Spot can be compared well against the simulation so therefore a comparison can be drawn.
Achievable: Since there will be multiple robot test sessions and an intense simulation development period beforehand, the project is suited well to understand the conversion from simulation to hardware
Relevant: Since the MSc Robotics mainly focuses on software and most of the courses only use simulation environments, it is of very high relevance to understand the implications that real-world solutions have, since in my future career I aim to work with real-world robots.
Time-bound: This learning goal should be reached towards the end of the course.

After the project, I think that I really progressed on that goal, since it was my first actual hardware project within robotics. There were a lot of differences, not only between how the sensors behaved in real life compared to the simulation (which I already expected), but unfortunately also in the naming of ROS internals for example frames and topics. This made it even harder to translate from simulation to real life and due to the very limited testing time, I think I did not fully reach that goal yet, but made a big step in the right direction.

Social development goal 1:
Specific: Improve my ability to give presentations to an expert audience, especially in the area of robotics.
Measurable: I will receive feedback on both the midterm presentation and the endterm presentation, the endterm presentation will even be graded.
Achievable: Since there are two presentations to be given within 6 weeks, this goal can be achieved well.
Relevant: Giving presentations is an important skill and especially a skill that many roboticists neglect. Therefore I think it is of high relevance to develop into a roboticist that can not only create but also present solutions.
Time-bound: This learning goal should be achieved at the two presentations (16.05 and 15.06).

I really learned a lot about presentation structure and techniques during both the presentation workshop and the preparation for the actual presentations. I am also happy, that the group entrusted Jurjen and me with giving both presentations. Although we did not get a formal grade on the first presentation, I think we really improved for the second presentation which was also reflected in the 4 out of 4 points that we got for presentation techniques in the final grade of the endterm presentation.

Peer Feedback Received and Peer Feedback Given
I was very happy with the feedback I received in both Buddychecks, which encouraged me to work even harder on the project since my group did appreciate the work I was doing. The feedback in Buddycheck 1 motivated me especially so I managed to act upon that feedback and thereby even slightly increase my given grades in Buddycheck 2. The two grades that I am most proud of are a 4.75 both in leadership and communicating. But this also connects to one of my potential weaknesses, sometimes talking too much, thereby maybe not giving others the chance to speak up. Not only was I happy with the grades I got in the Buddycheck, but also with the positive peer messages I was given. Furthermore, I really valued the constructive feedback on how I could further improve. The main improvements that were suggested were:

1. Asking others if they need help when I am already done with my task
2. Commenting my code more
3. While writing the state machine, ask for the other's input earlier

I think of all this feedback as valuable and valid. Therefore, I tried to incorporate it as much as possible. Point 1, asking others if they need help, I think I managed to act on that well for the first two weeks after getting that feedback. After that, I was mainly busy with writing the state machine and therefore sometimes did not have enough time to keep incorporating that feedback. Point 2, commenting code, I think I made very good progress with that and now I use docstrings, comments, and even type hinting in all my code which really helps me to become a better programmer, being able to work in groups in the same codebase. Point 3 was also very helpful, since when I sought the other's information on how their nodes work exactly, I could manage to connect them way easier than beforehand, where I tried to figure it out myself since I did not want to distract/annoy others by constantly asking. In hindsight, I should have done that from the beginning of taking on the task of the state machine, so the feedback was very valuable for me.

The feedback I gave to the team was met with enthusiasm and willingness to improve. Team members not only acknowledged the constructive criticism but actively incorporated it into their work. Witnessing their receptiveness and proactive attitude was satisfying, as it demonstrated a shared commitment. This positive response to feedback makes me excited about future group work.

Transferable Skills
Teamwork: This skill was really important for the project, and I feel like I improved in it since this project was way more workload than what I was used to from university projects. I have a lot of experience working in teams and I enjoy it, but I still think there's room for improvement with each new project and team I work with.
Leadership: I am very happy that the grades I received for leadership were 4.75, which gives me confidence that I have strength in that. After the first weeks in which we sometimes got stuck in the discussion of details, I think the team managed to streamline the project and we became very effective. I tried to always have an eye on who does what so we remain productive throughout the whole project. I also think that my groupmates were well structured so it was easy to do that.
Entrepeneurial thinking: This did not really play a part in the majority of the project, since we never considered the potential cost or economic impact of our solution. In the end for the final presentation however, we put quite some thought into how to sell our project to the client and decided to make more of an advertisement movie instead of going into too much technical detail. I think this paid off

since the client chose our group in the end.
Strategic multidisciplinary problem-solving: I really improved on this, and again realized, that communication is key to this. I also learned the importance of writing code in a very modular way and commenting it extensively from the beginning in order to enable others to quickly understand and use my developed solutions. It was also very interesting to brainstorm for a complete robotics solution from scratch since usually in university we only develop one predefined subsystem.

Agile, Specialist Roles, and Other Team Roles
In my opinion, the application of Agile project management in this project was somewhat challenging due to time constraints. I understand, that it makes sense to first think about the system architecture in depth, then start developing and iteratively improve the architecture alongside the code. But I think the course requested that the initial architecture concept be in too much detail which cost us a lot of time in rewriting that we could have spent better. While the specialist roles at the beginning of the project provided a helpful starting point, the additional roles were not utilized as much as expected, likely due to the focus on meeting deliverables within the limited timeframe. Overall, the project workflow was sometimes a bit unnatural due to the frequent deliverables. However, having specialized roles within the team did provide a sense of responsibility and aided in accomplishing tasks effectively.

Other Key Reflection Conclusions
After finishing this demanding and intense robotics project, and despite the challenges, I really can say that I found joy throughout the experience. As a student in robotics, the project pushed me to my limits and tested my skills. What facilitated that was the good team I had the chance to work with. We shared a good bond and supported each other. Looking back, I am very happy with how the project unfolded and the achievements we accomplished together.

6.4 Marco

Learning Goals
I had two main goals, one was to enhance my programming skills in ROS1 and Python to develop a complex program solving a real-world problem, and the other one was to improve my decision-making in project management by analyzing requirements and improving project efficiency.

Programming skills goal:
Specific: The goal is to focus on advanced programming concepts in ROS1 and Python, with the aim of developing a complex program that addresses a real-world problem in this project the Spot.
Measurable: The success of this goal will be measured by the ability to develop a complex program that solves a real-world problem using ROS1 and Python by the end of the project duration.
Achievable: This goal can be achieved by dedicating at least 3 hours every week to learning and practicing programming concepts, as well as watching online tutorials on YouTube.
Relevant: Improving programming skills in ROS1 and Python is relevant to the career aspirations of a robotics engineer/software developer, and will increase the chances of securing a job in the robotics/tech industry after graduation.
Time-bound: This goal will be achieved by the end of the project duration.

Decision-making skills goal:
Specific: The goal is to focus on improving decision-making skills in project management by analyzing project requirements, identifying potential roadblocks, and making informed decisions to improve project efficiency.
Measurable: The success of this goal will be measured by the ability to identify potential roadblocks in the project and make informed decisions to overcome them by the end of the project duration.
Achievable: This goal will be achieved by dedicating time to researching and analyzing project requirements, seeking feedback from peers, and practicing decision-making skills during the project.
Relevant: Improving decision-making skills is relevant to the career aspirations of a project manager and will increase the time efficiency of a project.
Time-bound: This goal will be achieved by the end of the project duration.

Reflecting on my learning goals, I can say that I have made significant improvements in my programming skills. When I initially started this project, my understanding of ROS was limited. However, through countless hours of dedicated work and coding in Python, I can say that I really improved myself. In the project, my primary role was focused on Planning/Navigation, where I was responsible for implementing the RRT* planning node. Initially, I successfully created this node but unfortunately due to a misunderstanding, I made it for the wrong environment. Thankfully, with Jurjen's help, we quickly resolved the issue and successfully implemented the RRT* node in the correct setting. After this project, I can say that I now know how to work with ROS combined with Python.

When reflecting on my second learning goal, I found it challenging to implement the decision-making skill goal into the project. It became clear after the initial sprints that we sometimes overlooked the criteria, resulting in additional workload. Here is where I could try to improve my decision-making skills to make the project more time efficient. At the end of the project, I focused more on meeting the criteria during the sprints, aiming to improve overall project efficiency. By carefully considering the criteria for sprint deadlines, I intended to save time in implementing feedback. In sprint 3, we followed the criteria, resulting in an impressive score of 14 out of 15 for criterion evaluation. This experience facilitated the development of my decision-making abilities by highlighting the significance of fulfilling project requirements. Throughout the project, I discovered that one of my strengths is having attention to detail, particularly when it comes to documentation and being exact in addressing the necessary details.

Peer Feedback Received and Peer Feedback Given
I appreciated the constructed feedback I received in both Buddychecks. With this constructed feedback, I could really try to improve myself. The feedback highlighted several areas where I could make improvements, but the most significant suggestions for improvement were:

1. A point of improvement is sharing more frequent updates of how far you are with your code so I have a better idea of what the progress is.

2. Maybe you could be a bit more active when we are discussing the possible solutions and let us know what your current problems with for example code are so we can jump in.

3. You could try to be a bit more present in the foreground

When looking at the feedback I agree with all the points provided. I made an effort to implement the suggested improvements as much as possible. Regarding point 1, I recognized that I had gone wrong in updating the team about my progress during the implementation of the RRT* node. I had updated them too late, which resulted in overlooking the fact that we were using a different environment. To prevent such mistakes in the future, I actively started asking my team members about their progress to avoid any miscommunications. For point 2, being a bit more active and letting problems know so we can jump in, is also valuable feedback. Although I couldn't resolve the problem on my own, I tend to be a bit stubborn, so my initial approach was to try and fix it myself. After receiving this feedback I immediately asked my team member Jurjen to help me with this problem. With his help, we were able to resolve the problem. This experience taught me the significance of updating the team, even if the update may not be ideal or satisfactory. For point 3, being more present in the foreground, I agreed with a lot. It is not in my nature to be too dominant, but after receiving the feedback, I made an effort to be more active and engaged. In the second Buddycheck, although I still can improve more at this it was noted that I did improve myself in the feedback: *"You could maybe speak up a bit more during meetings, but this is definitely improved wrt last buddy check! :)"*. Other positive feedback was: *"Does lots of work without the need of asking and has attention to detail, especially with documentation."* This was also my second personal development goal. So it was great that this was noticed.

The feedback was well-received by all team members, who demonstrated a proactive approach to implementing the given feedback. Each team member contributed with constant persistence and dedication, investing a notable amount of effort into the project. This helped the team to work for each other.

Based on this feedback, I have gained valuable insights into the areas I need to focus on for future improvement. In upcoming projects, my development goals will be actively being more in the foreground to enhance my leadership/initiative skills. Additionally, I plan to regularly check in with my team to ensure I am on the right track, thereby improving my overall job performance and preventing any potential misunderstandings, as mentioned earlier.

Transferable Skills
Teamwork: Teamwork is an important skill in group projects. As mentioned earlier, I made an effort to actively participate in discussions, resulting in noticeable improvement. However, I believe there is still room for further growth in this area. Improving this comes with doing more and more projects. I must highlight that the openness shown by all team members greatly facilitated our progress and overall effectiveness.
Leadership: As mentioned before it is not in my nature to be too dominant, but as a group we rotated the role of project leader every week. This meant that I had to take on the leadership role for a total of two weeks. Even though it's not where I feel most at ease, I believe I improved this skill during that time. This is the skill that I can improve the most. Moving forward, I see it as a personal development goal to improve in the future.

Entrepeneurial thinking: This was not really part of the project. This skill was only used when making a video of our vision for the final presentation.
Strategic multidisciplinary problem-solving: I made significant improvements in this skill throughout the project. We initially focused on creating individual nodes and later developed the state machine. If I were to undertake a similar project in the future, I would start with creating the state machine to align the team's direction before diving into node development. This approach will be useful for me in future projects of a similar nature.

During this project, I discovered an interest in working within a team and strategic multidisciplinary problem-solving. Working with smart and driven team members really helps me improve as an individual. I find that I can improve my skills much better when I collaborate with teammates. Engaging in discussions about the challenges we encountered and brainstorming solutions was a positive and enjoyable experience. There were no new strengths or weaknesses I identified during the course, but I do know better what my strengths and weaknesses are. Through this project, I learned a lot from my group members and improved my individual skills.

Agile, Specialist Roles, and Other Team Roles
Due to the project's short duration, implementing Agile project management did not prove to be beneficial. The limited timeframe made it challenging to find the optimal solution for the problem at hand. If the project had been longer or less complex, Agile project management could have been more suitable. Although having designated team roles helped me step out of my comfort zone, their effectiveness was limited due to the focus on meeting deadlines. It would be more advantageous to rotate these roles within larger and longer-duration projects, allowing me to truly develop my skills in specific areas. However, the specialized roles were promising as they clearly defined each team member's responsibilities.

Other Key Reflection Conclusions
When reflecting on the project, I found great satisfaction in the overall experience. Despite the workload being substantial, it provided valuable insights into problem-solving under time constraints. I genuinely find such projects fascinating and desire to participate in similar projects within a professional company in the future.

6.5 Jurjen

Learning Goals
With this project, I have set three learning goals for myself. Gaining more programming experience on a real-life project, becoming a better presenter for public presentations and client meetings and becoming better at working together on a project. This latter goal is specifically focused on the technical part of working together.

Technical development goal 1:
Specific: Become experienced in programming real-life applications.
Measurable: Create at least one fully functional node on the Spot robot and connect this to other nodes.
Achievable: Dedicate time to the actual programming. First by simply starting to program, do and watch some online tutorials and ask for help when stuck.
Relevant: Getting more experience in real-life applications helps for future, similar projects but also for future interns or jobs. Programming is something I enjoy so hopefully I can land a job where I'll be able to do this.
Time-bound: The goal can probably not be achieved within the hours for this course. Therefore I will also use time outside the course hours to work on this.

Social development goal:
Specific: Become a better public speaker for presentations but also for client meetings.
Measurable: Take the lead in the presenting role for the end presentation and client meetings.
Achievable: One or two weeks before the presentation start practicing. Maybe watch one or two videos about public speaking.
Relevant: Public speaking is an important skill for multiple job fields, from sales meetings to project presentations. This skill will come in very handy for future jobs.
Time-bound: This goal will be achieved at the end of the course. However it does not stop there, this is a skill that can constantly be improved.

Technical development goal 2:
Specific: Become more familiar with Git. Learn how to work this efficiently in a large project in a team.
Measurable: Make merges, branches and do re-basing. Perhaps write a small piece of documentation on it afterward for personal future use.
Achievable: Actively use Git, do online tutorials and ask for help from team members if stuck.
Relevant: Git is something that is used a lot in team projects. Becoming more familiar with it helps a lot for future work.
Time-bound: Hopefully I have become familiar enough with Git at the end of the course within the course hours. If not, I will continue working on it in future projects.

Looking back at the goals I set myself, I think I did pretty well. I got more experienced in real-life applications as I got thought valuable lessons about designing applications like Spot. Also, I became more used to working with Git, but not as much as I wanted to. It is still sometimes a difficult program to work with but can be so useful in projects like these. Unfortunately, I also did not get the time to make some sort of documentation for myself on working with Git. With this project I made a good start for these two goals, however, I think with more experience such as an internship I could really improve myself on this.
The presenting learning goal is something I am quite proud of as I took on the presenting role. During my studies I haven't prepared a presentation as much as I did for this final presentation. I tried to think of a way to present our product in a best-selling manner and personally I think it went very well, despite the nerves.

Peer Feedback Received and Peer Feedback Given
Personally, I was really happy about the received feedback. First off, my teammates gave me good grades on the job performances in both rounds. Given the time I put into this project, it is really nice to see how that pays off. In the second round of feedback I got graded even better. This is something I can relate to as I had some problems getting to start in the beginning of the project.
Not only was I happy about the grades, but also the peer messages really helped me improve my work attitude. In the first round I got *How to improve: "When you are a little behind ask for help if it is needed"* as feedback and worked on this. In the second round I received *"Strengths: "Actively seeks help when the code does not work"*, which was nice to receive given the feedback from the first round.
Other positive feedback I received was about being enthusiastic and bringing a nice vibe to the group. The downside of this enthusiasm was that I could be distractive sometimes and my English communication becomes slightly worse as I tend to speak before thinking.

Transferable Skills
Teamwork: This skill was really important during this project and I think my skills in this did improve. I have quite a lot of experience in teamwork and I really enjoy working in teams, but I still believe that for each new project in a new team, you improve yourself on this.
Leadership: This project thought how to steer a meeting in the right direction. Especially in the beginning we got sidetracked many times because of very enthusiastic and ambitious ideas that were quite unreachable. By reminding our group what our deliverables were for that week and dividing tasks we got back on track and delivered all our deliverables in a good manner.
Entrepeneurial thinking: Only at the end of the project do I think I improved on this skill. During preparations for the presentation I was thinking of various ways so that we could 'sell' our product as well as possible.
Strategic multidisciplinary problem-solving: I would say I have definitely improved on this skill since in retrospect, I would have handled some problems differently than we did during the project. One major lesson here remains that communication is key!

Because of this project, I found my interest in entrepreneurial thinking by trying to sell my product in the best way possible. Also, I think leadership might be one of my strengths and weaknesses at the same time. I received positive feedback on being a good leader but also that sometimes I can be a bit much in the foreground.

Agile, Specialist Roles, and Other Team Roles
In my opinion, the Agile project management was not applied successfully. Due to time constraints we were only able to sort of due one cycle of the method. However, with a bigger project with more time on hand this method could really pay off.
The specialist roles were nice so that we knew who was responsible for what at the beginning of the project. If we did not have these roles we would still divide responsibilities but it gave us a sort of headstart.
Unfortunately, in my opinion, the other, additional roles weren't used as much as I expected. Probably because of the time constraints and the focus on the actual deliverables it was not one of our top priorities to actually switch the additional roles.

Other Key Reflection Conclusions
This project was very nice to give a little taste of how it is to work on an actual big project within a company. Then, agile working might truly realize its potential. All in all this project really confirmed my interest in Robotics and has motivated me to look for an internship where I would work on similar projects.

7 Conclusion

This design report is written in conclusion of the Multidisciplinary Project (RO47007) from TU Delft in collaboration with TNO. The project aimed to alleviate the increasing strain on the healthcare system caused by the growing aging population, which places a heavy workload on healthcare professionals. The primary focus was to utilize the robot dog 'Spot' from Boston Dynamics to detect, pick up, and deliver items requested by healthcare personnel, thereby reducing their workload.

In chapter 3 the architecture of the nodes has been laid out starting with the extended operational scenarios in 3.1 which lead to the functional requirements in 3.2. This chapter also shows the activity diagram that describes the state machine. The requirements obtained in this chapter were used as a guide to design the process explained in chapter 4.

During the analysis of the problem multiple pre-existing nodes have been found that could be incorporated into the system but the decision was made to have all the nodes designed by the team because a requirement for passing the course was set that at least as many nodes as there are team members have to be written or *significantly* improved. The nodes developed for this project were designed using ROS 1, as the Spot robot operates on this framework. Eight nodes were developed to facilitate various functionalities, including SLAM, Detection, RRT Path Planning, Exploration, Motion Controller, SPOT Control Panel, Conversation node, and State machine node. A detailed description of their functions can be found in Chapter 4.

A simulation environment was provided for testing purposes. However, during the testing sessions with the actual robot, discrepancies between the simulation environment and the real robot had to be addressed. This required aligning and fine-tuning the nodes to function effectively. An account of the alignment process and its results can be found in Chapter 5.

This project has been completed in parallel with four other teams, at the end of the project the team is honored to have been selected as the team with the best solution to TNO's use case.

Overall, this project successfully demonstrated the implementation and integration of nodes that incorporate Spot to detect items and persons, and map these detections so that an item can be retrieved and brought to a specific person autonomously when requested. The outcomes of this project contribute to the ongoing efforts in finding technological solutions to support healthcare professionals and mitigate the challenges posed by an aging population.

References

[1] Ahmed Ali and Steve Renals. “Word Error Rate Estimation for Speech Recognition: e-WER”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 20–24. DOI: 10.18653/v1/P18-2004. URL: https://aclanthology.org/P18-2004.

[2] Alan Chan. “A technical report on a novel robotic lower limb rehabilitation device - Is ROBERT® a cost-effective solution for rehabilitation in Hong Kong?” In: *Hong Kong Physiotherapy Journal* (2022).

[3] ctsaitsao. *Turtlebot3 SLAM*. https://github.com/ctsaitsao/turtlebot3-slam/tree/main. June 19, 2023. 2023.

[4] Tom Duckett. *A genetic algorithm for simultaneous localization and mapping*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9340790. June 19, 2023. 2003.

[5] Boston Dynamics. *Arm Specification*. Accessed on [Insert Date]. Boston Dynamics. URL: https://dev.bostondynamics.com/docs/concepts/arm/arm_specification.

[6] Boston Dynamics. *Using the Image Service*. https://dev.bostondynamics.com/python/examples/get_image/readme. June 19, 2023. 2023.

[7] Foteini Filippidou and Lefteris Moussiades. “ Benchmarking of IBM, Google and Wit Automatic Speech Recognition Systems”. In: *Artificial Intelligence Applications and Innovations*. Ed. by Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis. Cham: Springer International Publishing, 2020, pp. 73–82.

[8] Sylvie Guerrero. “New Graduate Nurses’ Professional Commitment: Antecedents and Outcomes”. In: *Journal of Nursing Scholarship* (2017).

[9] Fahad Islam et al. “RRT-Smart: Rapid convergence implementation of RRT towards optimal solution”. In: (2012), pp. 1651–1656. DOI: 10.1109/ICMA.2012.6284384.

[10] Jolanda Zeeman. *I-I-You model*. https://brightspace.tudelft.nl/d2l/le/content/500953/viewContent/3153723/View. Accessed on May 15, 2023. 2023.

[11] Sertac Karaman and Emilio Frazzoli. *Sampling-based Algorithms for Optimal Motion Planning*. 2011. arXiv: 1105.1186 [cs.RO].

[12] Sertac Karaman et al. “Anytime Motion Planning using the RRT*”. In: (2011), pp. 1478–1483. DOI: 10.1109/ICRA.2011.5980479.

[13] Tom Keldenich. *YOLOv7 How to Use ? – Best Tutorial simple*. https://inside-machinelearning.com/en/use-yolov7/. June 19, 2023. 2023.

[14] leggedrobotics. *Darknet ROS Repository*. https://github.com/leggedrobotics/darknet_ros.

[15] motion-planning. *RRT Algorithms Repository*. https://github.com/motion-planning/rrt-algorithms.

[16] Nursing Home Abuse Center. *Nursing Home Abuse Center - Understaffing in Nursing Homes*. Accessed on May 29. Nursing Home Abuse Center. URL: https://www.nursinghomeabuse.org/nursing-home-neglect/understaffing/.

[17] Luigi Palmieri, Sven Koenig, and Kai O. Arras. “RRT-based nonholonomic motion planning using any-angle path biasing”. In: (2016), pp. 2775–2781. DOI: 10.1109/ICRA.2016.7487439.

[18] Jie Qi, Hui Yang, and Haixin Sun. “MOD-RRT*: A Sampling-Based Algorithm for Robot Path Planning in Dynamic Environment”. In: *IEEE Transactions on Industrial Electronics* 68.8 (2021), pp. 7244–7251. DOI: 10.1109/TIE.2020.2998740.

[19] Vincent Rabaud. *image$_proc - ROSWiki$*. http://wiki.ros.org/image_proc. June 19, 2023. 2015.

[20] T. Reichlmayr. “The agile approach in an undergraduate software engineering course project”. In: *33rd Annual Frontiers in Education, 2003. FIE 2003.* (2003).

[21] Robotics247. https://www.robotics247.com. Accessed on May 13, 2023. 2023.

[22] Rohit Kundu. *YOLO: Algorithm for Object Detection Explained*. Accessed June 1, 2023. https://www.v7labs.com/blog/yolo-object-detection. 2023.

[23] *ROS Navigation2 with SLAM Tutorial*. https://navigation.ros.org/tutorials/docs/navigation2_with_slam.html.

[24] *ROS rqt Plugins*. http://wiki.ros.org/rqt/Plugins.

[25] *ROS Wiki - RRT Exploration*. http://wiki.ros.org/rrt_exploration.

[26] J. Ricardo Sánchez Ibáñez, Carlos Perez-del-Pulgar, and Alfonso Garcia. “Path Planning for Autonomous Mobile Robots: A Review”. In: *Sensors* 21 (Nov. 2021), p. 7898. DOI: 10.3390/s21237898.

[27] ScazLab. *ros_speech2text*. GitHub repository. URL: https://github.com/ScazLab/ros_speech2text.

[28] skhadem. *3D Bounding Box Repository*. https://github.com/skhadem/3D-BoundingBox.

[29] *SMACH ROS*. http://wiki.ros.org/smach.

[30] TNO. *TNO website*. https://www.tno.nl. Accessed on May 13, 2023. 2023.

[31] Anirudh Topiwala, Pranav Inani, and Abhishek Kathpal. *Frontier Based Exploration for Autonomous Robot*. 2018. arXiv: 1806.03581 [cs.RO].

[32] TU Delft, Multidisciplinary project. *Course Manual*. https://brightspace.tudelft.nl/d2l/le/content/500953/viewContent/3140637/View. Accessed on May 13, 2023. 2023.

[33] TU Delft, Multidisciplinary project. *Project Description for Customer TNO*. https://brightspace.tudelft.nl/d2l/le/content/500953/viewContent/3146361/View. Accessed on May 13, 2023. 2023.

[34] Wikipedia contributors. *A* Search Algorithm*. https://en.wikipedia.org/wiki/A*_search_algorithm. Accessed: June 16, 2023.

A Change Log

Date	Task	Assigned to
01-05-2023	Create individual introductions for team members.	All
	Divide team organization roles.	All
	Start working on the first chapters of sprint 1.	Guido, Marco
	Starting the project planning with a Gantt Chart.	Jurjen
	Start was made for the project goal and Work Breakdown Structure.	Jakob, Jesse
02-05-2023	Evaluate and discuss possible solutions.	All
	Modify the introduction accordingly.	Marco, Jurjen
	Changed problem definition accordingly	Jesse
	Develop Chapter 2.3: Vision with SWOT analysis.	Jakob, Guido
04-05-2023	Add Chapter 6: Individual reflection.	All
	Finish Chapter 2.2: Planning with initial Gantt chart.	Jurjen
	Finish Chapter 2.3: Vision and SWOT analysis.	Jakob, Guido
	Brainstorm initial system architecture.	All
	Append Work Breakdown Structure to the appendix.	Jesse, Marco
	Finish the tasks for sprint 1	All
07-05-2023	Include Chapter 2.4: Operational scenarios table.	Jurjen, Jesse
	Include Chapter 2.5: Nodes	Jurjen
	Create a template for Chapter 3.1: Extended operational scenarios table.	Guido
	Started working on chapter 3.2: Needs and functional requirement	Marco, Jakob
09-05-2023	Almost finished a template for Chapter 3.1: Extended operational scenarios table.	Guido
	Almost finish working on chapter 3.2: Needs and functional requirement	Marco, Jesse
	Start on Chapter 3.3: Functional hierarchy tree.	Jurjen, Jakob
	Start working on Chapter 3.4: Activity diagram	Jesse, Jakob
11-05-2023	Almost finished Chapter 3.3: Functional hierarchy tree.	Jurjen, Jakob
	Almost finish working on Chapter 3.4: Activity diagram	Marco, Guido
	Finishing chapter 3.1 and 3.2	Jesse and All
15-05-2023	Incorporate feedback from the first sprint report.	Jesse, Jakob
	Complete final details of Chapter 2 and Chapter 3 for sprint 2.	Jurjen, Marco and All
	Prepare client presentation for 16-05.	Guido

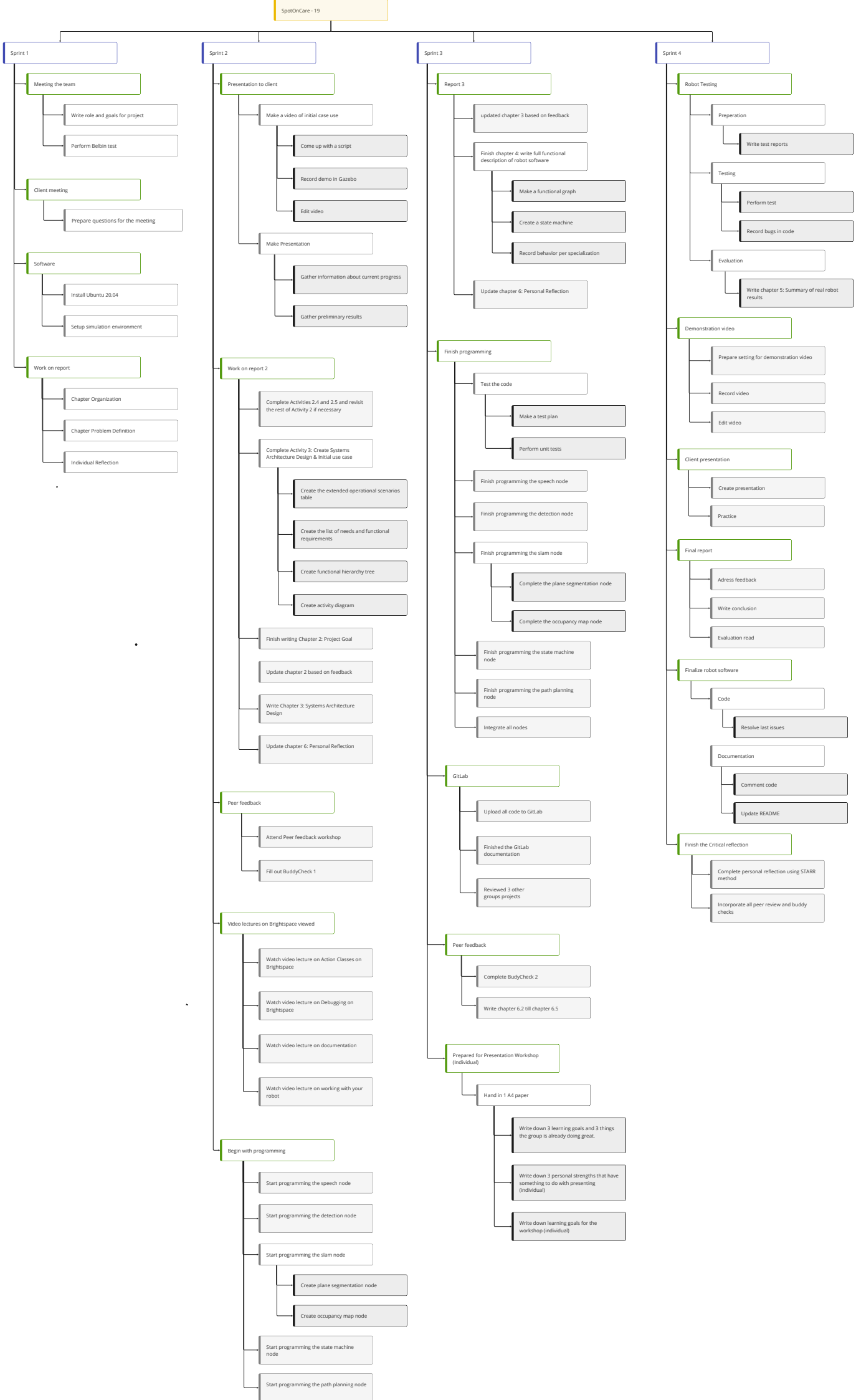
Date	Task	Assigned to
18-05-2023	Online meeting to show the progress of our codes.	All
	Discuss the problems that we walk into and discuss how to solve this.	All
23-05-2023	Incorporate feedback from the second sprint report.	Jesse, Jakob
	Create a planning for the testing day.	All
	Prepare client presentation for 16-05.	Guido
25-05-2023	Testing day	All
	Set up the connection with the Spot	Guido
	Tried to test the node for Point cloud detection	Jesse and Jakob
	Test node for mapping	Jurjen, Marco and All
30-05-2023	Rewrite the code for Pointclouds detection.	Guido
	Rewrite the planning for RRT.	Jurjen and Marco
	Test simulation for nodes combined.	Guido and Jesse
	Test the Human-robot interaction nodes together with RRT.	Jakob and Jurjen
01-06-2023	Test day	All
	Capturing videos/take pictures from Nodes	All
	Make screenshots of the simulation	Jurjen, Marco and Jakob
	Test the YOLO Detection node and almost works.	Jesse
	Test the mapping of the environment.	Guido
	Test the RRT path for the Spot, but Twist did not work.	Jurjen
	Work on the report while others are testing.	Marco and Jakob
	Report all the debugging.	All
02-06-2023	Work on the Gitlab documentation.	All
	Trying to finish the state machine for testing next week.	Guido and Jakob
	Update the Change Log.	Marco
	Implementing the gripper camera for YOLO detection.	Jesse

Date	Task	Assigned to
06-06-2023	Write Chapter 4	All
	Initiate subsection on SLAM in Chapter 4	Guido
	Begin work on subsection about Detection in Chapter 4	Jesse
	Take charge of the Explore subsection in Chapter 4	Jurjen
	Handle the subsection on RRT in Chapter 4	Marco
	Contribute to the Motion control subsection in Chapter 4	Jurjen
	Begin the topic of Human interaction in Chapter 4	Jakob
07-06-2023	Work on the report	All
	Complete the subsection on SLAM in Chapter 4	Guido
	Finalize the subsection about Detection in Chapter 4	Jesse
	Conclude the Explore subsection in Chapter 4	Jurjen
	Wrap up the subsection on RRT in Chapter 4	Marco
	Contribute to the completion of the Motion control subsection in Chapter 4	Jurjen
	Conclude the exploration of Human interaction in Chapter 4	Jakob
08-06-2023	Test day	All
	Implemented rqt graph of the Spot	Guido
	Record demonstration video for Spot	Jesse, Guido
	Test Human interation node	Jakob
	Test the RRT path planning node	Marco
	Test the Motion control node	Jurjen
09-06-2023	Finishing the report for deadline sprint 3	All
	Edit recorded demonstration video	Guido
	Implement the rqt graph in Chapter 4.1 and made clear who did what	Marco and Jurjen
	Already make a start with debug chapter 5	Jesse, Marco and Jurjen
	Finish the requirements for sprint 3	All
12-06-2023	Recorded interviews for video presentation	All
	Edit recorded interviews video	Guido
	Made presentation slides and devise structure for presentation	Jakob and Jurjen
	Further work on chapter 5	Jesse and Marco

Date	Task	Assigned to
13-06-2023	Final meeting before presentation	All
	Finalized video for presentation	Guido
	Practised giving presentation	Jakob and Jurjen
	Further work on chapter 5	Jesse and Marco
15-06-2023	Attended presentations and gave own presentation	All
	Won first prize for best TNO Robot Design	All
16-06-2023	Finalized report for final deadline	All
19-06-2023	Delivered final report	All

B Work Breakdown Structure

See next page. The image has been vectorized, allowing it to be zoomed in without any loss of quality. The WBS is build up in such a way that if an action has a grayed-out backround it is a final action meaning that it is the lowest level at the breakdown.


SpotOnCare - 19
Sprint 1
Meeting the team
Write role and goals for project
Perform Belbin test
Client meeting
Prepare questions for the meeting
Software
Install Ubuntu 20.04
Setup simulation environment
Work on report
Chapter Organization
Chapter Problem Definition
Individual Reflection
Sprint 2
Presentation to client
Make a video of initial case use
Come up with a script
Record demo in Gazebo
Edit video
Make Presentation
Gather information about current progress
Gather preliminary results
Work on report 2
Complete Activities 2.4 and 2.5 and revisit the rest of Activity 2 if necessary
Complete Activity 3: Create Systems Architecture Design & Initial use case
Create the extended operational scenarios table
Create the list of needs and functional requirements
Create functional hierarchy tree
Create activity diagram
Finish writing Chapter 2: Project Goal
Update chapter 2 based on feedback
Write Chapter 3: Systems Architecture Design
Update chapter 6: Personal Reflection
Peer feedback
Attend Peer feedback workshop
Fill out BuddyCheck 1
Video lectures on Brightspace viewed
Watch video lecture on Action Classes on Brightspace
Watch video lecture on Debugging on Brightspace
Watch video lecture on documentation
Watch video lecture on working with your robot
Begin with programming
Start programming the speech node
Start programming the detection node
Start programming the slam node
Create plane segmentation node
Create occupancy map node
Start programming the state machine node
Start programming the path planning node
Sprint 3
Report 3
updated chapter 3 based on feedback
Finish chapter 4: write full functional description of robot software
Make a functional graph
Create a state machine
Record behavior per specialization
Update chapter 6: Personal Reflection
Finish programming
Test the code
Make a test plan
Perform unit tests
Finish programming the speech node
Finish programming the detection node
Finish programming the slam node
Complete the plane segmentation node
Complete the occupancy map node
Finish programming the state machine node
Finish programming the path planning node
Integrate all nodes
GitLab
Upload all code to GitLab
Finished the GitLab documentation
Reviewed 3 other groups projects
Peer feedback
Complete BudyCheck 2
Write chapter 6.2 till chapter 6.5
Prepared for Presentation Workshop (Individual)
Hand in 1 A4 paper
Write down 3 learning goals and 3 things the group is already doing great.
Write down 3 personal strengths that have something to do with presenting (individual)
Write down learning goals for the workshop (individual)
Sprint 4
Robot Testing
Preperation
Write test reports
Testing
Perform test
Record bugs in code
Evaluation
Write chapter 5: Summary of real robot results
Demonstration video
Prepare setting for demonstration video
Record video
Edit video
Client presentation
Create presentation
Practice
Final report
Adress feedback
Write conclusion
Evaluation read
Finalize robot software
Code
Resolve last issues
Documentation
Comment code
Update README
Finish the Critical reflection
Complete personal reflection using STARR method
Incorporate all peer review and buddy checks

C Gantt Chart

Figure 28: Gantt Chart sprint 1, 2

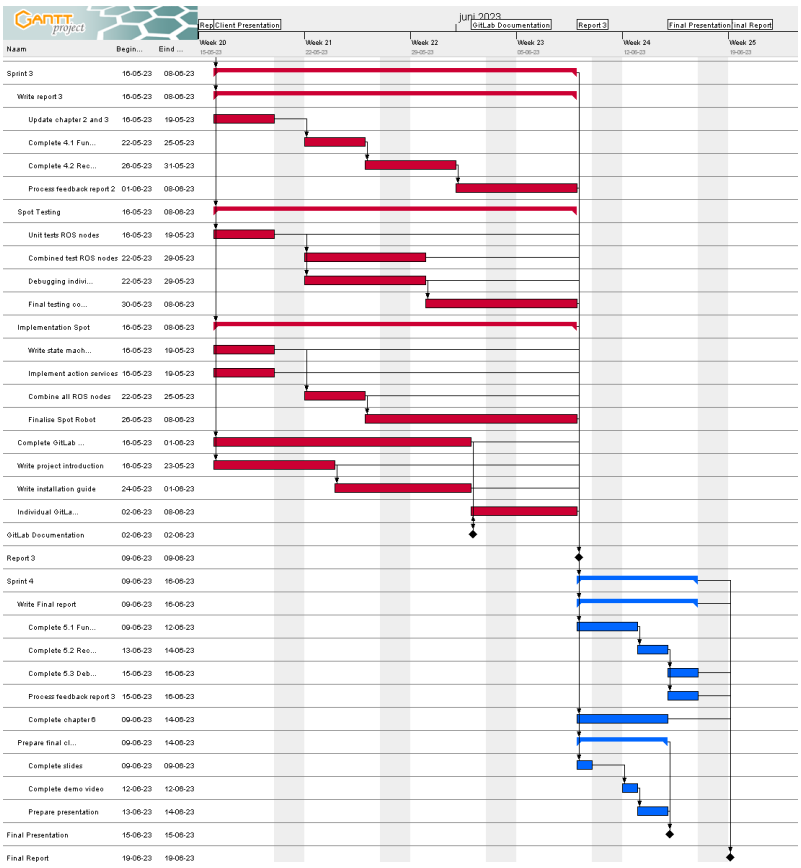


Figure 29: Gantt Chart sprint 3, 4

D State Machine diagram

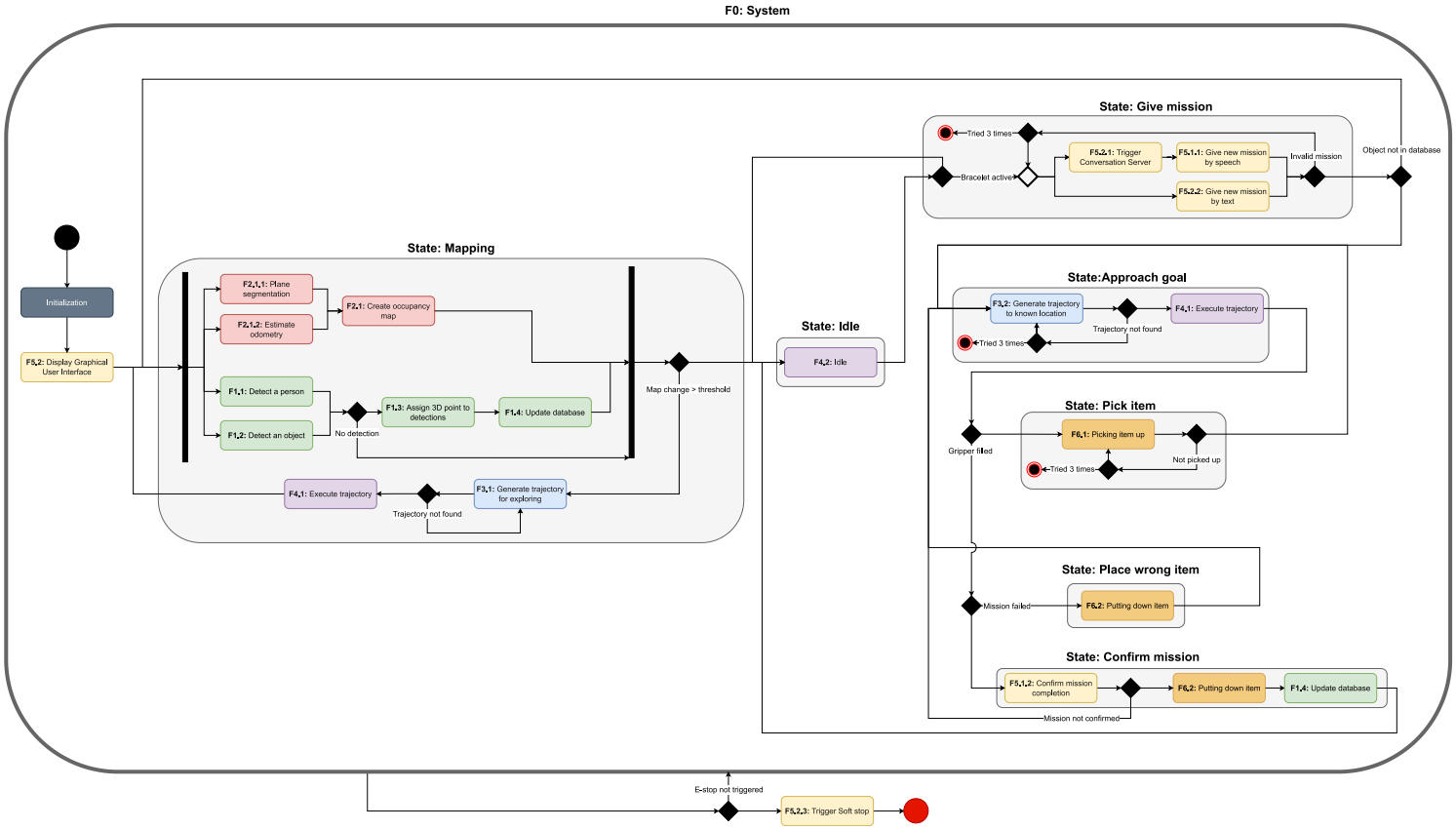


Figure 30: State Machine

E RQT Graph

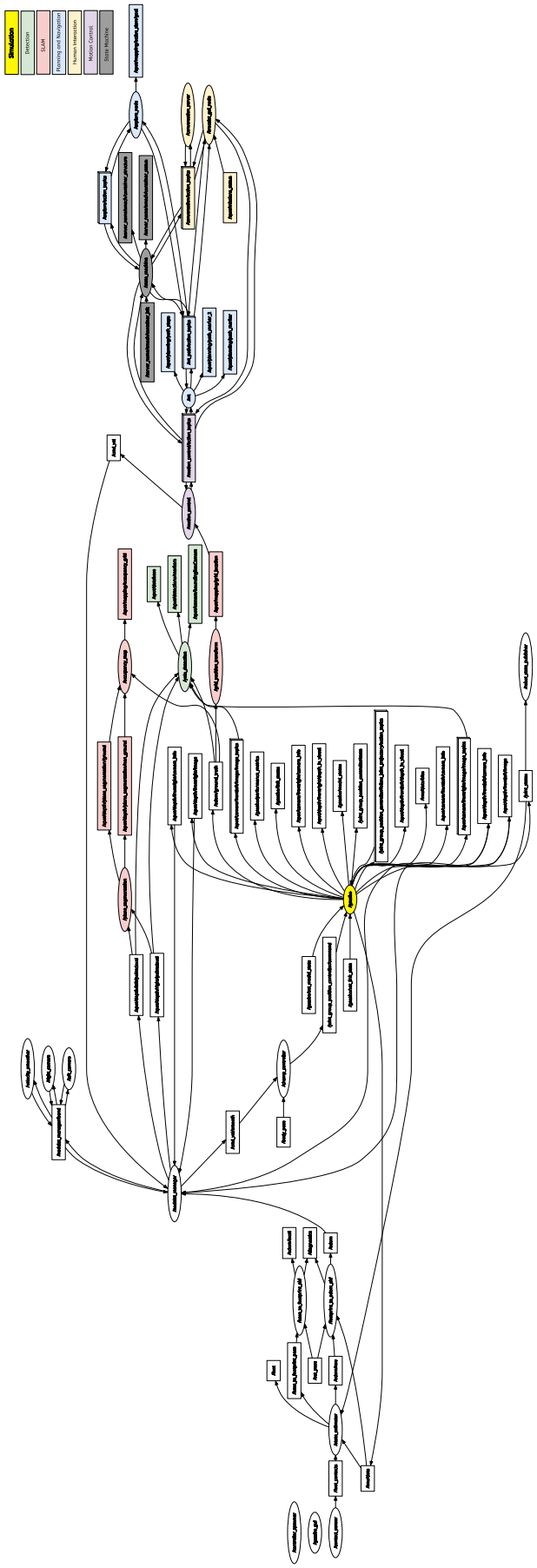


Figure 31: RQT graph recorded using the simulation

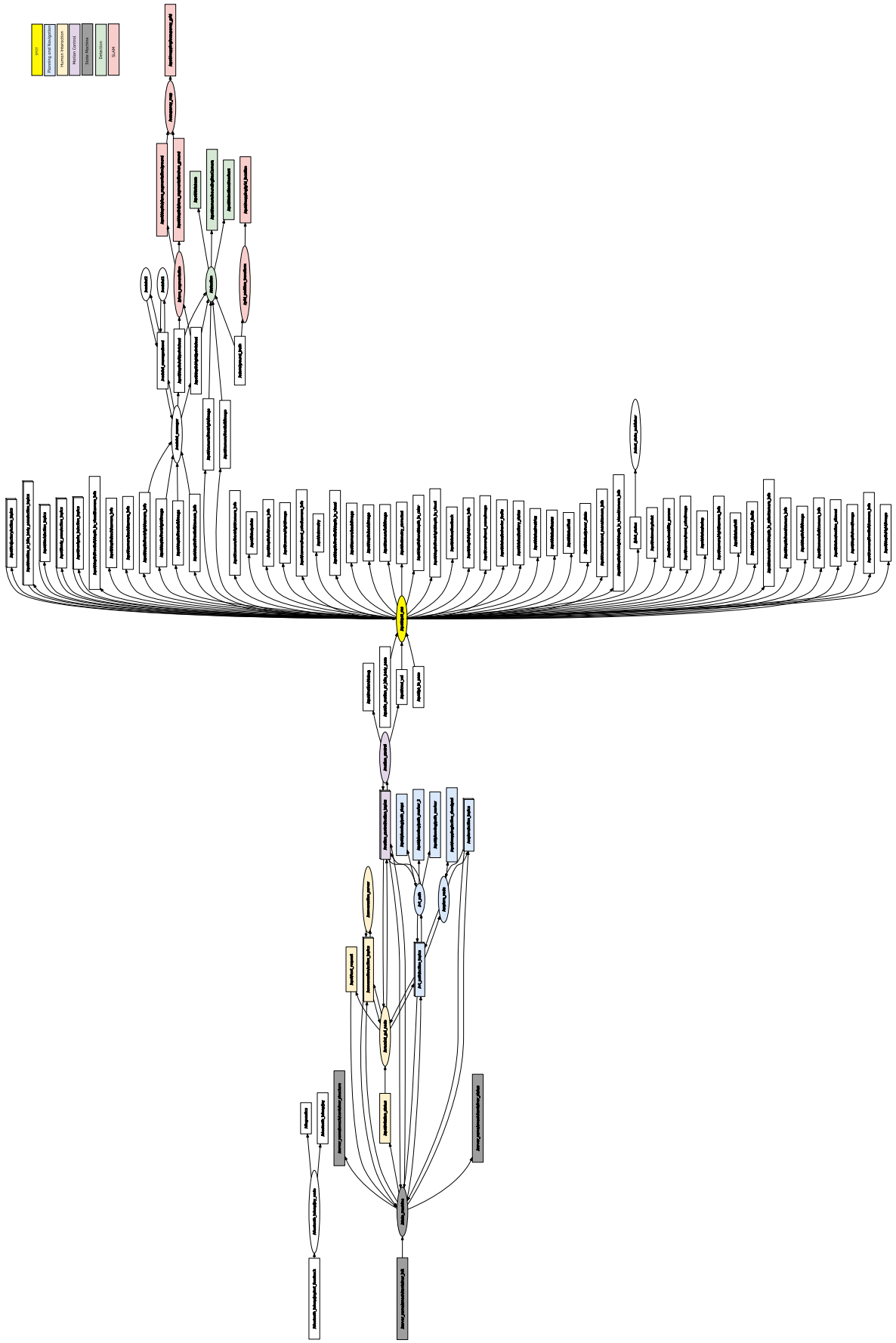


Figure 32: RQT graph recorded on the real robot