Norwegian University
of Life Sciences

Master's Thesis 2020 30 ECTS
Faculty of Science and Technology

The Development of an Autonomous Docking System for Charging of the Agricultural Robot, Thorvald

Utviklingen av et autonomt navigasjonssystem for lading av landbruksroboten, Thorvald

Casper Andersen Rødsrud
Mechanical Engineering, Process Technology and Product Development

The Development of an Autonomous Docking System for Charging of the Agricultural Robot, Thorvald

By Casper Andersen Rødsrud

– Master's Thesis –
Mechanical Engineering, Process Technology and Product Development
NMBU – Faculty of Science and Technology
2020

Preface

This master's thesis defines my final work as a graduate student at the Norwegian University of Life Sciences. Through five years of intense studies, extracurricular work, and loads of fun, I am now prepared to take on new challenges in life. The Mechanical Engineering Programme at NMBU has given me unique tools for my toolbox, that I know will become a great advantage as I move forward. I am thankful for all the expertise the University has shared with me and the that opportunities I have been given.

My motivation to work with autonomous systems started to flourish during my year abroad in the Bay Area. As a central place for research on autonomous technology, I was able to undertake interesting classes at UC Berkeley that gave me insight into ongoing research. Here, I was introduced to control theory and mechatronics, which became a big part of my exchange. What I brought home was an eagerness to learn more, which led me to take on the task of developing an autonomous docking system for Thorvald.

My learning curve throughout this project has been tremendous. I have learned about the many aspects of autonomous navigation, become more comfortable with the programming of embedded systems and developed an understanding of how robots work.

I would first like to thank Associate Professor, Lars Grimstad, for supervising me throughout the intense process of writing this master's thesis, and for giving me the opportunity to dive into the world of robotics without much prior experience. Thank you for the opportunity to work on your creation, I am truly grateful for the knowledge you have given me.

Furthermore, I would like to thank Associate Professor, Jan Kåre Bøe, for being my secondary supervisor. I am grateful for our many discussions about product development and design, and for all the advice you have given me.

Thanks to Assistant Professor, Antonio C. Leite, for spending afternoons and Friday nights discussing control theory and robotics with me, and not to mention, introducing me to his previous co-workers and students in Rio de Janeiro.

Last, but not least, I would like to thank my family and friends for being loving and supportive through my ups and downs at the University, my girlfriend who has kept me motivated and helped me during the master's process, and finally, my friend and study mate, Oscar, who has been my "partner in crime" since day one at the University.

Oslo, 27. May 2020

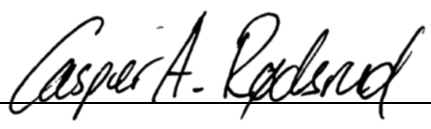


Table 1-1: Overview and description of abbreviations used throughout the thesis.

Abbreviation	Description
2D/3D	2 Dimensional / 3 Dimensional
ADAS	Advanced Driver Assistance Systems
AI	Artificial Intelligence
Amcl	Adaptive Monte Carlo Localization
API	Application Programming Interface
BC	Before Christ
CAD	Computer Aided Design
EHS	Environment, Health and Safety
FOV	Field of view
FPS	Frames per second
GLASOD	Global Assessment of Human-Induced Soil Degradation
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
ID	Identity
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IPD	Integrated Product Development
Lidar	Acronym for Laser Image, Detection and Ranging
QR Code	A machine-readable code
RF	Radio Frequency
Radar	Acronym for Radio Detection and Ranging
ROS	Robot Operating System
RPC	Remote Procedure Calls
SLAM	Simultaneous Localization and Mapping
UGV	Unmanned Ground Vehicle
UN	United Nations
UV	Ultraviolet
Yaw	Short for yaw angle

Abstract

While the population growth continues at a steady pace, so does the need for increased food production. With an estimated population of nearly ten billion people within 2050, food production in agriculture has to remain of great focus. The development of mobile robots that can increase the yield and reduce the footprint of big machines is an important contribution to the prevention of possible future food shortages.

The Thorvald concept considers an electric versatile robot platform that is lightweight and can perform various agricultural tasks. However, for the robot to operate on its own, it is desired to develop autonomous functionality that requires minimal human interaction. The main goal for this master thesis has been to develop a navigation system that allows the robot to autonomously recharge. From an arbitrary point in an operational environment, Thorvald needs to navigate to its charging station, enter it, and align with the charger. This report presents the planning, investigation, research, development, and validation of an autonomous navigation system for charging of Thorvald.

The first part of the report provides the reader with background information and an introduction to the Thorvald concept. Then, the purpose of the thesis is presented before the development process is explained step by step.

The initial phase of the development involved investigation and planning. A comprehensive plan was constructed to provide an overview and to ensure steady progression during the development. IPD, Integrated Product Development, was alongside several design methods with roots in this philosophy, used to develop the system. Among these methods were Pugh's method for specification and selection, an extraction from the SCAMPER method, and a modified version of the Waterfall method.

Existing solutions were investigated in search of inspiration for the docking system. Important aspects of autonomous navigation were revealed and used as guidelines to determine the system specifications. Also, to ensure rigorous evaluation of these specifications, a literature review was conducted. During this review, important knowledge about relevant theoretical principles and technology was obtained.

Based on the system goals and specifications, concepts were generated for the various components of the system. The concepts were then evaluated based on the system criteria that were determined through the specification and assessed in a selection process. Pugh's weighted matrices were used to converge into an optimal solution to the problem. However, in a selection of concept for the detection of the charging station and the dock, two of the generated concepts were inseparable. Hence, small-scale experiments were conducted for each of the two concepts to determine which was better. Results from these experiments were used to create a survey that was sent to people with experience in navigation of mobile robots and sensor technology.

Feedback from the survey led to a third small-scale experiment, which helped select a final concept.

Multiple design iterations and re-evaluations led to adjustments and improvements of the system specifications before a final solution was proposed. The proposed solution included a concept for detection, an algorithm for the generation of waypoints, a pose regulator for motion control, and a module for state estimation as a complete system for autonomous docking of Thorvald.

Virtual and real-world experiments of the developed system were conducted to validate the various system components. Results from these experiments and important moments of the development process were used as a basis for the discussion and conclusion in the final parts of the report.

Not directly presented in the report is the software developed for the function. The software was developed according to a framework named ROS and was simulated and improved continuously throughout the development process through the utilization of several digital tools.

Link to GitHub repository: https://github.com/CappiJoe/master_thesis_code.git

Sammendrag

Samtidig som befolkningen stadig øker, øker også behovet for matproduksjon. Med en estimert befolkning på nærmere 50 milliarder mennesker i 2050, må det fokuset som er rettet mot matproduksjon i landbruket opprettholdes. Utviklingen av mobile roboter som kan øke utbytte og redusere avtrykkene fra store maskiner er en viktig bidragsyter i forhindringen av potensiell matmangel.

Thorvald omhandler en allsidig, elektrisk robot plattform som både veier lite og som kan utføre en rekke oppgaver i jordbruket. Dog, for at roboten skal kunne operere på egenhånd så er det ønskelig å utvikle funksjonalitet som kan minimere behovet for menneskelig kontakt. Hovedmålet i denne masteroppgaven var å utvikle et navigasjonssystem som gjør det mulig for roboten å lade autonomt. Det vil si at Thorvald, fra et vilkårlig punkt, skal kunne navigere seg selv til sin ladestasjon, entre den og plassere seg ved siden av laderen. Denne rapporten beskriver utredningen og utviklingen av et autonomt navigasjonssystem for lading av Thorvald.

Den første delen av rapporten gir leseren bakgrunnsinformasjon og introduserer konseptet rundt Thorvald. Deretter blir hensikten med prosjektet presentert før utviklingen av navigasjonssystemet blir beskrevet steg for steg.

I første fase av utviklingen ble det gjort det gjort et dykk i eksisterende litteratur innen navigasjon for mobile roboter. Deretter ble en plan laget for prosjektet med hensikt å få oversikt over prosessen samt sørge for progresjon i arbeidet. Integrert produktutvikling, IPD, ble brukt sammen med flere ulike designmetoder for å utvikle systemet. Blant disse var Pughs metode for spesifisering og seleksjon, et utdrag fra SCAMPER, samt en modifisert versjon av Waterfall for programvareutvikling.

Eksisterende løsninger ble brukt som inspirasjon for den autonome «docking-funksjonen» til Thorvald. Et dypere dykk i litteraturen avduket viktige momenter for autonom navigasjon som ble brukt i spesifiseringen av funksjonen. For å sørge for tilstrekkelig kvalitet på vurderingen av spesifikasjonene ble det også gjennomført en litteraturstudie der relevant teori og teknologi for prosjektet ble redegjort for.

Basert på systemmål og spesifikasjoner ble det generert konsepter for de ulike delene av systemet. Konseptene ble deretter evaluert med hensyn til ulike kriterier som ble bestemt sammen med systemspesifikasjonene. Pughs seleksjonsmatrise ble her tatt i bruk for å selektere den mest passende løsningen. Dog endte en seleksjon for deteksjonskonsepter opp med to konsepter som tilsynelatende var veldig passende, og det ble derfor gjennomført små eksperimenter for å evaluere ytterligere. Resultatene fra disse eksperimentene ble så brukt til å lage en eksperttest som ble sendt ut til personer med erfaring innen sensorteknologi og med navigasjon av mobile roboter. Tilbakemeldinger fra testingen førte til et tredje eksperiment som videre la grunnlaget for den valgte løsningen.

Et større antall designiterasjoner førte til justeringer og forbedringer av konseptene før én løsning ble bestemt. Den foreslåtte løsningen bestod av et konsept for deteksjon, en algoritme for generering av rute, en kontroller og et forslag til lokaliseringsmodul som del av et komplett system for autonom docking.

Virtuelle og reelle eksperimenter ble gjennomført for den utviklede løsningen for å validere de ulike systemkomponentene. Resultatene ble, sammen med momenter fra utviklingsprosessen, brukt som grunnlag i en endelig diskusjon og konklusjon for prosjektet.

Det som ikke direkte er gjort rede for i rapporten er programvaren, som ble utviklet i samsvar med rammeverket ROS, og som kontinuerlig ble testet og forbedret gjennom utviklingen ved bruk av digitale verktøy.

Link til GitHub repository: https://github.com/CappiJoe/master_thesis_code.git

Table of Contents

1 Introduction

This is a master thesis that is connected to the development of Thorvald, an agricultural robot designed by a robotics research group at the Norwegian University of Life Sciences. As a contribution to sustainable agriculture through the utilization of lightweight robots, this thesis will address challenges connected to the autonomous charging of the robot platform.

1.1 Background

12.000 years ago, the majority of the human society consisted of foragers, also called hunter-gatherers. Foragers were nomadic groups that traveled vast distances while hunting and gathering for sustenance. However, at approximately 10.000 years BC, groups individually started to settle. Why they did is arguable, but a common belief is that they realized that if they collected enough seeds from grass and vegetation, they could plant these and later harvest enough food to last through the non-growing season. The transition led to permanent communities, in which citizens domesticated plants and animals, and utilized what today is referred to as agriculture.

Through photosynthesis, plants and vegetation have made it possible for people to divert energy emitted from the sun into the development of products for human sustenance. Throughout history, farmers have utilized agriculture to ensure a steady production of food. However, as the human population continues to grow towards eight billion people, and with the UN's estimation of almost ten billion people in 2050, the demand for food along with climate changes pose challenges for the future of agriculture [1].

The evolution of agriculture has enabled farmers to take advantage of technologies that allow them to control the growth and harvesting of agricultural products. Development of tools and technology, such as the tiller machines replacing the conventional hoe, and harvesting machines replacing manual labor, has made back-breaking work such as tillage and weed killing far less demanding, improving many aspects of EHS in agriculture. However, the biggest challenges in agriculture are not connected to the level of effort needed to plant or harvest. Problems due to soil degradation, resource depletion, and food waste are becoming more common in agricultural production.

Soil degradation is a term that describes a decrease in the quality of soil as a result of improper handling or management. Soil compaction, for instance, caused by heavy machinery, is an example of a process that can cause a reduction in fertility because it prevents air and water to infiltrate the soil [2]. Another example is rapid crop tillage which breaks down the soil and increases wind and water erosion and threatens agricultural sustainability.

Due to the many challenges the agricultural industry is facing, the demand for solutions that will reduce the risk of degradation and depletion, while maintaining productivity and efficiency, is increasing.

Figure 1-1 shows a map that is generated from soil degradation data gathered through an assessment (GLASOD) conducted by ISRIC between 1988 and 1991 [3]. Although the map shows data from almost three decades ago, it indicates how large human-induced soil degradation in agriculture can be.

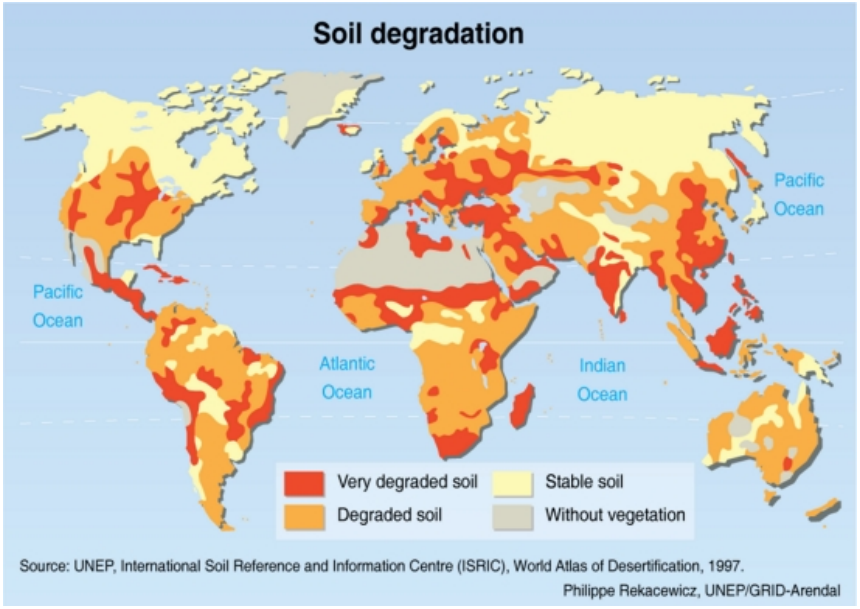


Figure 1-1: Map of Global Soil Degradation (Grida.no)

1.1.1 Precision agriculture and agricultural robots

To prepare the agricultural industry for future challenges, researchers from all over the world work to find innovative solutions to the problems. The development of technology moves at a fast pace, and robots intended for the agricultural industry are already able to perform tasks such as weeding, seeding, and harvesting. Modern farming technology such as the robot Jäti's weed-killing lasers [4] and AVO's accurate pesticide spraying [5] are agricultural techniques that, potentially, will replace conventional pesticide and herbicide methods. As we enter the 21st century, the relevance of precision agriculture increases alongside the need for farmers to treat each part of their crop individually.

By utilizing these precise methods, farmers can perform tasks on specific plants and areas based on their need for hydration, nutrients, and care. One major advantage is the large potential for reduced use of chemicals by utilizing precision methods for fertilizing, pest killing, and weed killing. Precision agriculture can, therefore, be beneficial by reducing destructive environmental effects, improving EHS in farming, and reducing costs for both producers and consumers. One way of performing precision agriculture is by using terrestrial or aerial robots, as they can perform challenging tasks with high accuracy, making them very suitable for precision agriculture [6].

Lightweight, mobile robots can eliminate many challenges, especially related to soil compaction and degradation. A reduction in the impact on the soil can potentially increase the overall yield and help maintain sustainability in crops. Lightweight characteristics also allow for the robots to operate in wet and muddy environments, in which heavy machinery tends to get stuck or damage the soil.

The majority of all mobile robots are powered by electricity, which means that they can be fueled by energy from renewable resources. A transition from using fossil fuels in tractors into using renewable energy to power robots will contribute to a reduction in CO2 emissions. Electric power is also more appropriate for systems that need to recharge autonomously.

Autonomous charging is essential for autonomous robots but requires implementation of additional functionality. A function for autonomous charging needs to enable the robot to navigate itself to for instance a charging station and dock with a charger. This navigation problem requires the robot to be able to perceive the goal, plan a path to it, and execute motion to get there.

Creating an optimal path from one point to another can be challenging in dynamic environments where few objects are stationary. However, the operating environment for an agricultural robot is often predictable, and path planning can, therefore, be less challenging to solve than, for instance, self-driving cars.

1.1.2 The Thorvald concept

The focus of this thesis will be directed towards an agricultural mobile robot platform named Thorvald [7]. The development of the Thorvald concept started as a project at the Norwegian University of Life Sciences (NMBU) in 2014, where the aim was to develop a lightweight and low-cost mobile robot for operations in agriculture.

The newest version of the robot, Thorvald II, has a modular design and can be programmed to perform many different tasks. Examples of tasks are data collection for crop prediction, phenotyping, and practical tasks such as seeding, weeding, and harvesting. In comparison to manual labor and conventional tractors, the robot can potentially perform tasks both more energy-efficient and cost-efficient. Two different configurations for Thorvald II is shown in Figure 1-2.

Figure 1-2: An illustration that shows two different configurations of Thorvald II (nmbu.no).

The mechanical structure of Thorvald II is rather simple, mostly consisting of off-the-shelf standard components. However, a big advantage for the platform is its low mass, which makes it both very agile and suitable for operations in agricultural fields, as its footprints remain small, also when the ground is wet.

A primary goal for the research group is to make Thorvald II fully autonomous. To allow for full-time operation, a charging station that will be placed within the robot's operating environment is currently under development. However, the robot needs a function that enables it to navigate to the charging station and align with a charger to recharge. The need for a system that performs the necessary tasks for docking defines the main purpose of this project.

This thesis will investigate techniques for navigation of mobile robots to design and implement a function that will allow Thorvald II to navigate itself into a charging station and align with its charger. The problem will include localization of the charging mechanism, planning a path to align with it, and motion execution. As the proposed designs for the charging station consider closed structures, investigation of methods for indoor navigation in the absence of GNSS will also be necessary.

This chapter has introduced the problem that will be considered in this thesis. The next chapter presents the scope of the thesis along with a plan for the development process.

Thorvald II is the platform considered in this thesis, and from this point and on, Thorvald II will be referred to as Thorvald.

2 Scope and project planning

The main limitation for this master's thesis is the timeframe, which is limited to four and a half months. It is, therefore, important to create a stable framework for the project through comprehensive planning. This chapter presents the scope of the thesis and challenges that will need to be considered. Furthermore, a main goal and associated intermediate objectives for the project are presented, as well as a list of additional constraints and limitations.

2.1 Scope of the thesis

The scope of this thesis revolves around investigating existing solutions to develop a function that allows Thorvald to autonomously dock with a charger. The function will enable Thorvald to navigate to a charging station from an arbitrary position, enter the station, and align with a charger. The thesis will not consider tasks for when the robot has finished charging. It desired to make the function as transferable as possible to make it usable on various configurations of the robot. A list of constraints and early assumptions is provided later in this chapter to provide a better understanding of the scope of the thesis.

2.2 Challenges

This section provides a list of challenges that are relevant to the project. The challenges are based on early investigation on mobile robot navigation systems.

- For localization, both accuracy and precision are crucial. The localization module needs to be robust to ensure minimal estimation errors for the pose of the robot.
- An indoor navigation system should not be dependent on satellite system signals.
- A navigation system for a robot that operates in a space that is occupied by obstacles should include functions for detection and avoidance.
- A navigation system has to be computational appropriate for the considered robot's controller and processor.
- Environmental conditions such as light and weather pose challenges to robot perception.
- Environmental conditions may also pose challenges to the motion of the robot. Examples are mud, ground irregularities, and rocks.
- Unforeseen incidents may occur. Safety functions must be included or able to override a navigation system.
- Sensors may require maintenance and calibration to maintain accuracy and precision.

2.3 Project objectives

Goal setting is a powerful tool that can be useful in many situations. In this project, goals are made to give the project purpose and to motivate throughout the process.

2.3.1 Main goal

To autonomously dock with a charger, the robot first needs to localize the charging station and safely navigate to it, before the robot needs to enter the station and align with the charging mechanism. The main goal for this master thesis is as follows:

"The main goal is to develop a navigation system for autonomous recharging of the agricultural robot, Thorvald. The system must be integrable with the Thorvald concept and should be transferable between different configurations. A comprehensive report will describe and discuss the development process."

An example of a docking situation is illustrated in Figure 2-1, in which Thorvald starts a docking procedure with an initial configuration at A and docks with a final configuration at D in the charging station, F.

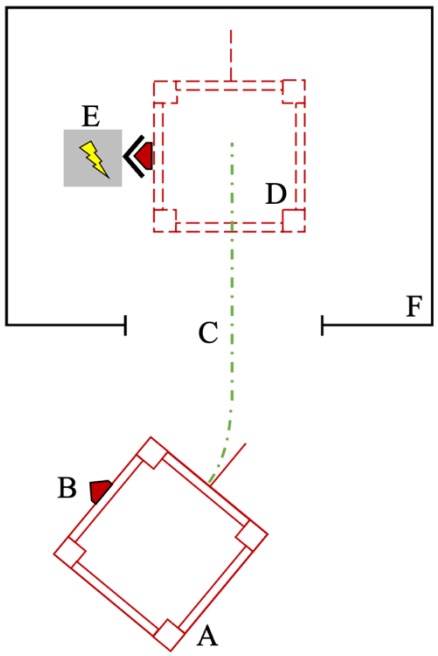


Figure 2-1: Illustration of a docking scenario. A: Thorvald's initial pose, B: Charging port socket, C: Generated path, D: The final pose with the charger plugged, E: Charger, F: Charging station.

2.3.2 Sub-goals and activities

In this section, the main goal is broken down into smaller sub-goals and activities to ensure structure, and to allow for monitoring of progression throughout the process.

1) Collect background information.
    - Obtain the necessary knowledge about agriculture and mobile robots to understand Thorvald's operating environment.
    - Become familiar with Thorvald's system and existing functionality.
2) Conduct in-depth research.
    - Investigate technologies used for indoor navigation and autonomous charging of mobile robots.
    - Investigate sensor technologies and techniques for robot perception.
    - Investigate control strategies that can be transferable between various configurations of Thorvald.
3) Propose a solution.
    - Define system specifications for the autonomous docking system.
    - Analyze, evaluate, and select appropriate sensor technology for localization of the charging station gate and the charger inside it.
    - Design an algorithm that generates waypoints and goal poses for Thorvald.
    - Select and include an appropriate control strategy for Thorvald's motion.

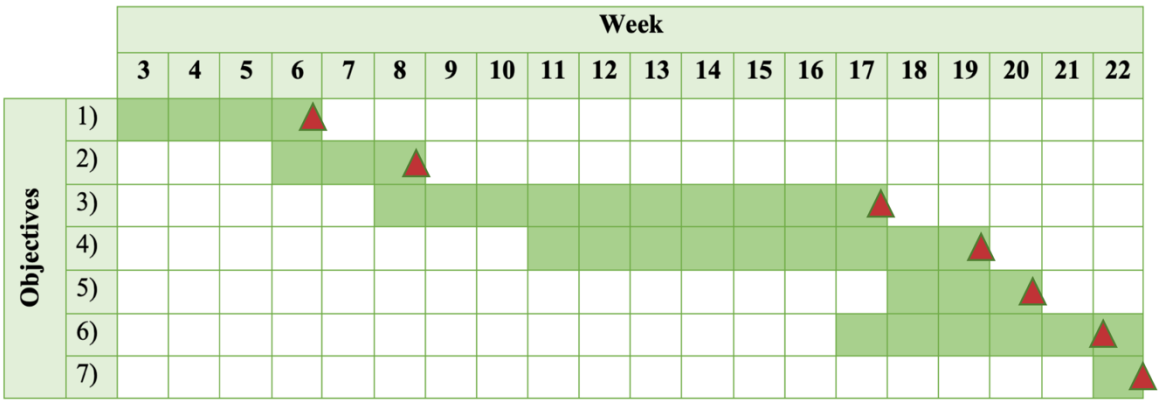
4) Develop system software.
    - Develop software that allows for localization of the charging station and the charger.
    - Develop software that generates a goal pose that aligns with the charger or as a final pose when entering the charging station.
    - Develop software that generates waypoints as a path to the goal.
    - Develop software that generates necessary motion commands for Thorvald to move to a desired location.
5) Validate system software.
    - Test the developed software in simulation.
    - Test the developed software on Thorvald.
6) Finish the thesis (deadline: 02.06.2020).
7) Make a presentation based on the contents of the report.

A brief milestone plan based on the sub-goals and activities is presented in Table 2-1 and Table 2-2.

Table 2-1: An overview of the milestones planned for the project.

Milestone	Description	Date
1)	The necessary background information has been collected.	07.02.2020
2)	The in-depth research has been conducted.	21.02.2020
3)	The final solution is proposed.	20.04.2020
4)	The system software is finalized.	08.05.2020
5)	The system software has been tested on the robot platform.	15.05.2020
6)	The thesis report is finalized and submitted.	27.05.2020
7)	The master thesis has been presented.	27.05.2020 -

Table 2-2: Gantt chart for planned progress with milestones marked with red triangles.



2.4 Early project constraints

Because the charging station in which Thorvald will dock is under development, there are several uncertainties that need to be considered. To count for these, and to narrow the scope of the thesis, the following limitations, constraints and assumptions have been set:

Development constraints

- The time limit for the project is 20 weeks.
- A standard configuration of Thorvald will be used for the development.
- Existing research will be used as the main inspiration for the function components.
- The charging mechanism will not be considered, nor will the connection with it.
- The undocking of the robot will not be considered.
- There will not be conducted any comprehensive patent search due to the strict timeframe.
- There will not be conducted any economic analysis for the docking system.

Technological constraints

- Thorvald's existing safety system will override any high-level control structure.
- The docking system is constrained to control longitudinal velocity and angular velocity about the z-axis of Thorvald. Lower level controllers will interpret these commands and translate them into motor commands.
- Software development will be constrained to the ROS framework to ensure integrability with Thorvald's existing systems.
- The physical design of the charging station is already proposed, and the docking system will be developed based on this.
- With limited access to the robot, the majority of testing will be conducted through simulation and with simple home-made experimental setups.
- The testing of sensors will be constrained by their availability and will be limited to small-scale experiments.
- The path planning problem will be simplified to generating waypoints.
- Potential path planning algorithms will not consider energy consumption by the robot.

Early assumptions

- An existing topological navigation system can be used to move the robot to an initial pose which faces the charging station's gate.
- Thorvald will be facing the station when the docking procedure begins.
- The design of the charging dock includes two pillars. The midpoint between the pillars marks the center of the dock,
- The center of the dock marks the desired position for the robot's geometric center.

- The robot uses a differential drive for its motion and behaves like a non-holonomic system during the docking procedure.
- The charging station environment is static and remains unchanged throughout the docking procedure.
- Satellite signals are absent inside the docking station.
- The surface of the ground remains flat throughout the docking procedure. Calculations can, therefore, be simplified to 2D.

Additional assumptions will be described later in the report if it is considered relevant.

The next chapter describes the methodology for the development and provides an overview of the terminology and vocabulary for the process.

3 Methodology

This chapter describes the methodology for the development process. Lists with terms and abbreviations formulas, symbols, and corresponding units are also provided in this chapter.

Figure 3-1 depicts the base link frame, which defines Thorvald's reference coordinate system. As the navigation system will be simplified to 2D, the roll and pitch angles, and movement along the z-axis will be set to zero. Thorvald's x-axis depicts the positive longitudinal direction.

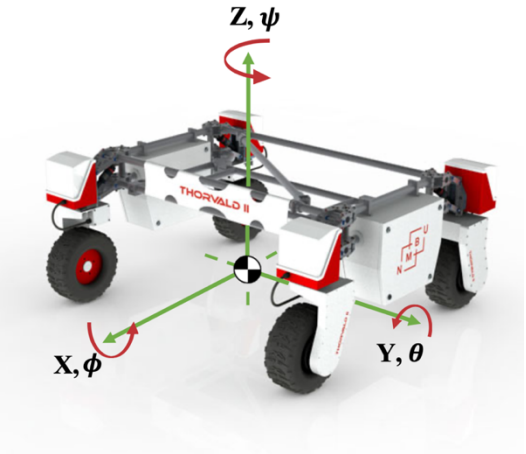


Figure 3-1: An illustration Thorvald's local coordinate system with positive directions and rotations.

3.1 Terminology

Table 3-1: An overview of the terminology and nomenclature that will be used in the thesis.

Term	Description
Accelerometer	A device that measures static and dynamic acceleration forces.
Base frame	The coordinate system related to the base link of a robot. Commonly defined with the origin set at the robot's center of rotation.
Base link	The link that defines the base of a robot.
Beacon	A device that transmits signals or is used to attract attention to a specific location.
Body-fixed frame	The coordinate frame that is locked to a particular body.
C++	A high-level, general-purpose programming language.
Client library	Code that allows developed projects to interact with an API.
Debugging	Debugging describes the process of eliminating errors from developed code.
Fiducial marker	A marker that appears in an image and can be used as a point of reference.
Frame	Short for frame of reference. Defines an abstract coordinate system.

Table 3-2: Table 3-1 continues.

Term	Description
Gyroscope	A device that measures angular velocities, the rate of change in angle over time.
Heading angle	The angle that corresponds to the direction of the longitudinal axis of the robot. For Thorvald, the heading angle is equal to the yaw of the robot ψ.
Non-holonomic system	A physical system for which the state is dependent on the path taken to achieve it.
Odometry	Use of sensor data to estimate change in position.
Pitch angle, θ	An angle that describes the rotation needed to align the longitudinal axis with the horizontal plane.
Pose	A triple that defines the position and orientation of an object.
Python	A high-level, general-purpose programming language.
Roll angle, ϕ	An angle that describes the rotation needed to align the lateral axis with the horizontal plane.
Sensor noise	Irregular variations that tend to obscure electrical signals. Also referred to as just noise.
Steering angle, δ	An angle that describes the rotation of the steering wheels with respect to the longitudinal axis of the robot.
Yaw angle, ψ	An angle that describes the rotation needed to align the longitudinal axis and the lateral axis with the XZ-plane and the YZ-plane, respectively. The yaw angle also describes the heading angle of the robot.

3.1.1 Symbols and units

Table 3-3: An overview of symbols and units that will be used in the thesis.

Index	Description	Unit, SI
ϕ	Roll angle	rad
θ	Pitch angle	rad
Ψ	Yaw angle	rad
ψ	Heading angle	rad
$\dot{\phi}$	Rate of change in roll angle	rad/s
$\dot{\theta}$	Rate of change in pitch angle	rad/s

Table 3-4: Table 3-3 continues.

Index	Description	Unit, SI
$\dot{\psi}, \omega$	Rate of change in yaw angle	rad/s
$\ddot{\phi}$	Angular acceleration about the x-axis	rad/s^2
$\ddot{\theta}$	Angular acceleration about the y-axis	rad/s^2
$\ddot{\psi}$	Angular acceleration about the z-axis	rad/s^2
δ	Steering angle	rad
v	Velocity	m/s
x	Displacement	m
d	Distance	m
c	Speed of Light	m/s
t	Time	s
r	Radius	m
μ	Mean	–
σ	Standard deviation	–

Symbols will be assigned with subscripts in relevant cases. An example is v_x, which will be used for the translational velocity along the longitudinal axis, the x-axis, of the robot.

3.1.2 Elementary formulas

Table 3-5 provides an overview of fundamental formulas for the thesis. More specific formulas will be presented for cases throughout the report and numbered continuously within the chapters they appear. The formulas will not be derived unless it is considered necessary for the relevant problem. Figure 3-2 is an illustration of the unit circle, which will be used as reference for positive and negative angles and rotations.

Table 3-5: An overview of fundamental formulas for the thesis.

Name	Formula	Index
Newton's law of motion	$F = ma$	3.1
Velocity	$v = \lim_{\Delta t \to 0} \frac{\Delta d}{\Delta t}$	3.2
Acceleration	$a = \lim_{\Delta t \to 0} \frac{\Delta v}{\Delta t}$	3.3

Table 3-6: Table 3-5 continues.

Name	Formula	Index
Centripetal acceleration	$a_c = \frac{v^2}{r}$	3.4
Law of Sines	$\frac{\sin(A)}{a} = \frac{\sin(B)}{b} = \frac{\sin(C)}{c}$	3.5
Euclidean norm	$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$	3.6
The cosine rule	$\vec{a} \cdot \vec{b} = \|\vec{a}\| \cdot \|\vec{b}\| \cdot \cos(\theta)$	3.7
Standard deviation	$\sigma = \sqrt{\frac{\sum(x-\mu)^2}{N}}$	3.8

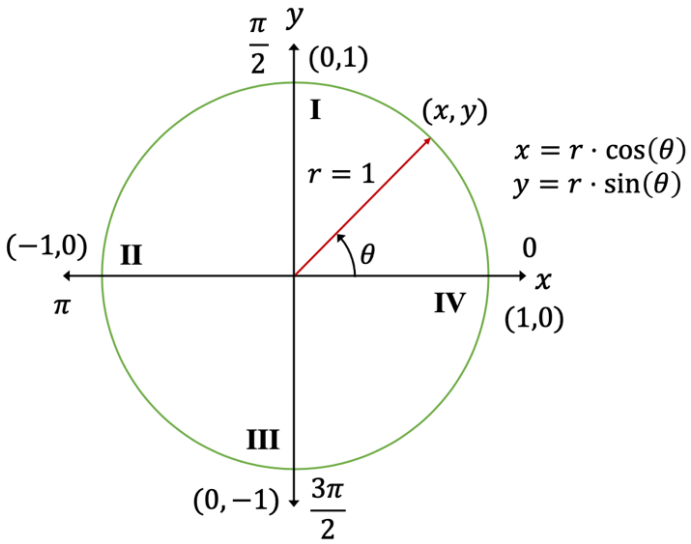


Figure 3-2: An illustration of the unit circle which defines the positive axes and angles, and polar coordinates that will be used for calculations.

3.2 Development methods

This section provides a description of four methods that will be used for the development of the docking system; IPD, Pugh's method, a modified Waterfall method and SMART.

3.2.1 Integrated Product Development

Integrated Product Development, IPD, is commonly a very hardware-focused method. However, it can be applied to any product development process, both for physical products and for processes or big systems. The general purpose of IPD is to ensure that all phases included in the development of a product are done thoroughly and in the correct order. The method has roots in the United States and was first mentioned by Mogens Myrup Andreasen in his book "Integrated Product Development" [8]. IPD promotes continuous learning and continuous improvement of a product or a process through the utilization of

data tools, data communication, and nonetheless, the collaboration between people with different expertise and background through multidisciplinary platforms.

IPD can also be used to help structure the development and to maintain focus both EHS, economy, and the use of resources so that important aspects are not left behind [9]. EHS is very important in the development of the Thorvald concept, which also contributes to NMBU's goal of adding sustainable value to the development of society through innovation [10].

In the process of developing an autonomous docking system for Thorvald, integrated product development will be utilized to structure the research and development, to ensure continuous learning and improvement, and to ensure that important aspects are not left out.

3.2.2 Pugh's Method

Pugh's method promotes three main elements; definition, metric specification, and controlled convergence. The method includes the mapping of customer needs to determine measurable specifications for a product or a process. Concept alternatives are then generated based on the specifications, and Pugh's selection method used to determine which is more optimal.

Pugh's selection method uses weighted matrices to evaluate concept alternatives based on criteria set by the specifications. In these matrices, alternatives receive scores based on how well they meet the criteria. Pugh's selection ensures controlled convergence into choosing the most appropriate solution to a problem [11]. In this thesis, the method will be used to define the system specifications and requirements and to select concepts for a final solution.

3.2.3 The Waterfall Method

The Waterfall method is commonly used in software development to ensure progress in projects. The waterfall method is known to be rigid and requires that one development phase is finished before a new phase can begin. In this thesis, a modified version of the method will be used as a guideline for software development. Figure 3-3 illustrates the Waterfall method with a modified workflow [12].

The first step of the Waterfall method involves the collection of relevant information to determine requirements for a system. The requirements are often used as the basis for a system design in the second step. When a system design is specified, an architecture for the system can be determined, and the coding process can begin.

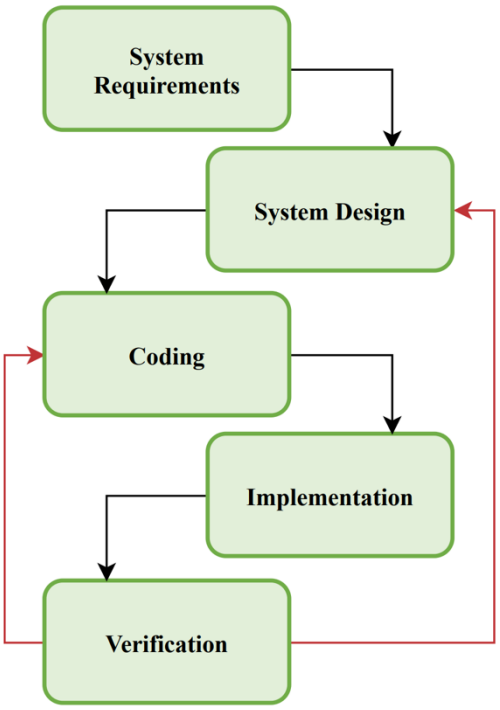


Figure 3-3: An Illustration of the Waterfall Software Development Method.

The Waterfall method suggests that a system is developed in individual parts, for which each element can be tested to ensure the desired functionality. When the software is complete, it can be verified through testing and implemented with the remaining parts of the system [13]. The classic Waterfall method does not allow going back to previous stages. However, the modified workflow is proposed because the navigation system will mainly be constructed of hardware-based software which requires continuous testing, tuning, and improvement.

3.2.4 SCAMPER

SCAMPER is a multidisciplinary method that can be utilized in almost any phase of product development. Bob Eberle proposed this method in 1971 as a modified and arranged version of Alex Faickney Osborn's brainstorming method [14], [15].

The method challenges the developer to find new solutions to the problems considered throughout the development. To do so, SCAMPER suggests, mainly, seven different strategies; *substitute (S), combine (C), adapt (A), modify/minimize/maximize (M), purpose (P), eliminate (E), rearrange/reverse (R)*. Often, many of these strategies are used without the developer being aware of it.

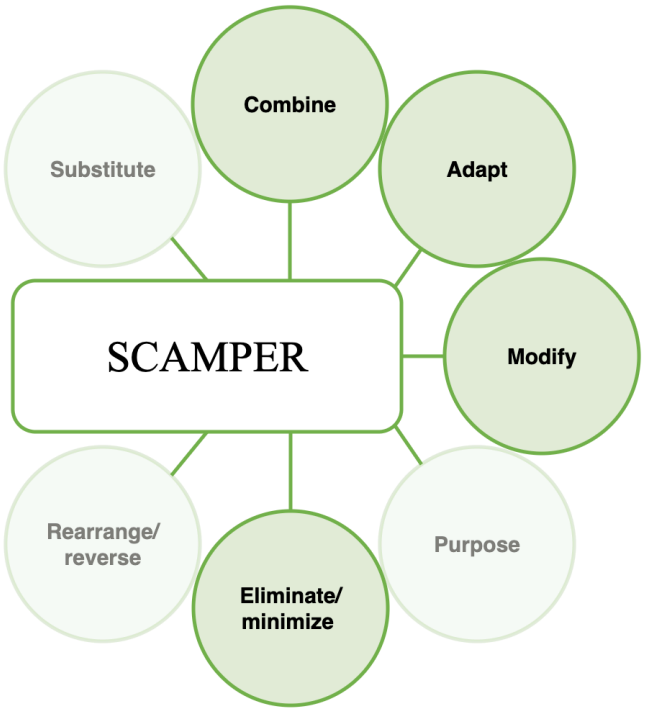


Figure 3-4: An illustration of SCAMPER with corresponding strategies. The strategies that are most relevant for this project are highlighted.

For this thesis, the four strategies highlighted in Figure 3-4 will be used. *Combine* will be used to determine if any functions can be combined or created as modules. *Adapt* is already used to adapt general product and software development methods to fit the purpose of this thesis and will be further used to determine if existing technology can be adapted to solve the main problem. *Modify* and *Eliminate* will be used in the final phase to analyze the software and reveal potential for improvements to ensure a robust and efficient solution.

3.2.5 SMART

SMART is a method often used for goal setting. The name of the method is an acronym made from criteria considered necessary for a well-defined goal. The description given in Table 3-7 on the next page will be used in this thesis. SMART will, throughout the thesis, be used as a guideline tool to make feasible and concrete goals for both system specifications and potential experiments [16].

Table 3-7: The chosen definition for the SMART abbreviation.

S	M	A	R	T
Specific	Measurable, motivational	Achievable	Reasonable	Time-based

3.3 Data tools

Canonical Ltd., Ubuntu Operating System, Version 18.04 [17]

Ubuntu is an open source operating system that will be used for software development and simulation.

Python Software Foundation, Python Version 2.7 [18]

Python is an object-oriented programming language that will be used for the development of software for the navigation system.

2020 GitHub Inc., Atom 1.43.0 [19]

Atom is an open source text and source code editor that will be used as an integrated development environment for software development.

Open Robotics, ROS Melodic Morenia [20]

ROS, short for Robot Operating System, is a framework that contains a collection of packages and tools that simplify the development of robot technology. Thorvald currently uses the version ROS Melodic [21].

Open Robotics, Gazebo 11.0.0 [22]

A stand-alone robot simulation software that is fully compatible with ROS. Gazebo allows for creation of realistic environments for simulation of robot operation.

JGraph Ltd., Draw.io, version 12.7.0 [23]

Draw.io is a web-based program that will be used to construct flow charts for the report.

Dassault Systems, SolidWorks 2019, SP 2 [24]

CAD software that will be used to design 3D models for simulation purposes. An add-on will be used to convert CAD models to *urdf*, which is the filetype of the parameter file that is read by the simulation software [25].

MATLAB version R2018a, The Mathworks, Inc [26]

MATLAB is programming platform that uses a unique matrix-based language for iterative analysis and system design. MATLAB will be used for data analysis and for control design.

Simulink version R2018a, The Mathworks, Inc [27]

Simulink is an environment for model-based design in which block diagrams are used for system-based design, simulation and continuous tests of embedded systems. The environment is integrated with MATLAB and will be used for control design purposes.

3.4 Information gathering and quality assurance

3.4.1 Resources

To ensure quality, literature will be used as the main source of information in this thesis. The literature will consist of books from the NMBU library, online books, and course readers from undertaken classes.

Web-based information will be gathered through online databases and search engines such as Research Gate, IEEE, Science Direct, and Web of Science. The databases will be used to investigate existing solutions and ongoing research within relevant fields.

3.4.2 Quality assurance

The following table provides an overview of important literature that will be used for quality assurance in the thesis:

Table 3-8: Table of used standards and quality assurance literature.

Use	Literature or source
Quality Management Systems – Fundamentals and Vocabulary	NS-EN ISO9000:2015 [28]
Quality Management Systems – Requirements Planning, realization and development	NS-EN ISO9001:2015, chapter 8 and appendix [29] NS-EN ISO9001:2000, chapter 7
Elementary physics principles	Physics for Scientists and Engineers 6th Ed., Tipler et. al [30]
Linear algebra	Linear Algebra and its Applications, 5th Ed., Lay et. al [31]
Engineering formulas	Engineering formulas, 8th Ed., Gieck [32]

3.4.3 Patents

To make sure that the technology that will be used is not protected, a brief patent search will be conducted in the databases for the organizations below. Results from the search will be listed in Appendix III.

Table 3-9: An overview of patent organizations with their corresponding purpose of use.

Purpose	Organization
Norwegian patent search	Patentstyret [33]
European patent search	The European Patent Office, EPO [34]
International patent search	World Intellectual Property Organization [35]

3.5 Process chart

Figure 3-5 and Figure 3-6 present flowcharts that show the planned process steps for the development of the autonomous docking system for Thorvald.

The first phase consists of planning the project, as well as gathering information and knowledge for the process of defining the main goal. The charts are made with symbology that corresponds to NS-EN ISO9001. An overview of the symbols is provided below along with the flowchart of phase 1 for the development.

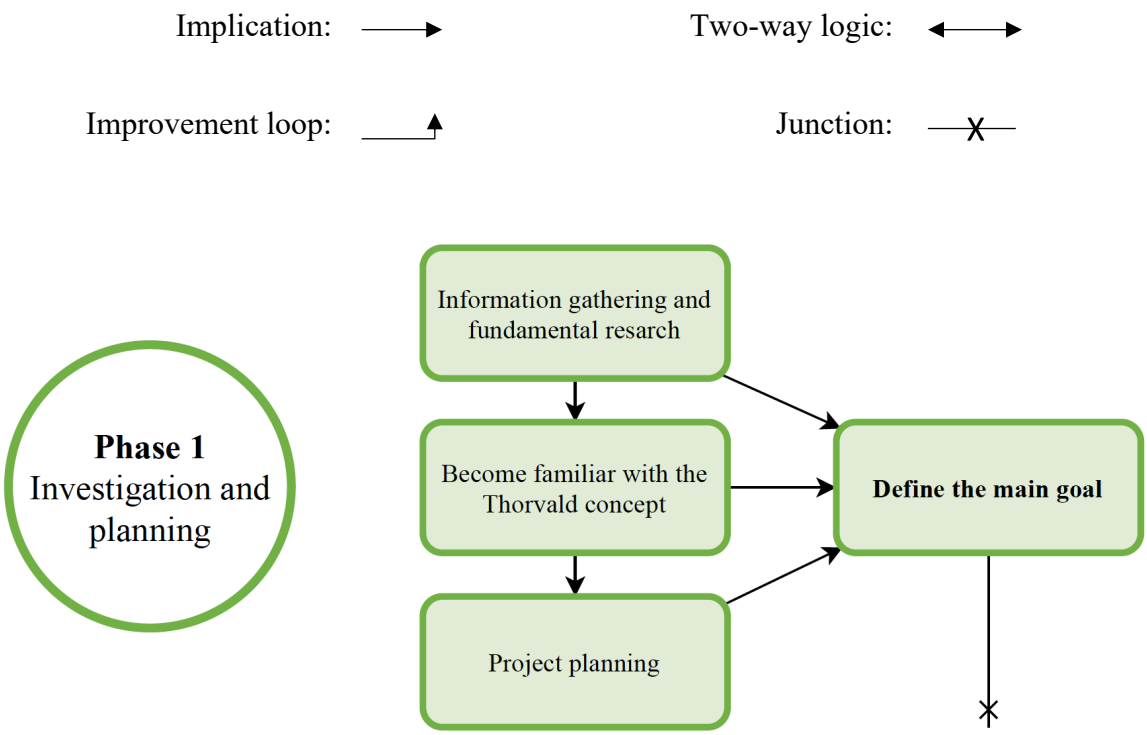


Figure 3-5: A flowchart that shows phase 1 of the process.

Phase 2 of the process involves processes related to research and development. Existing technology will be evaluated to discover potential alternatives for the various components of the navigation system. Concepts will be generated based on system specifications and evaluated in a selection process to determine which is most appropriate. The most appropriate concepts will be used to propose a complete solution to the main problem. When the final solution is proposed, software for the function will be developed using the modified Waterfall method.

In Phase 3, a series of experiments will be conducted to validate the functionality of the proposed solution. The results will be analyzed and used to improve the function as much as possible before finalizing the development. Both Phase 2 and Phase 3 are shown in Figure 3-6 on the next page.

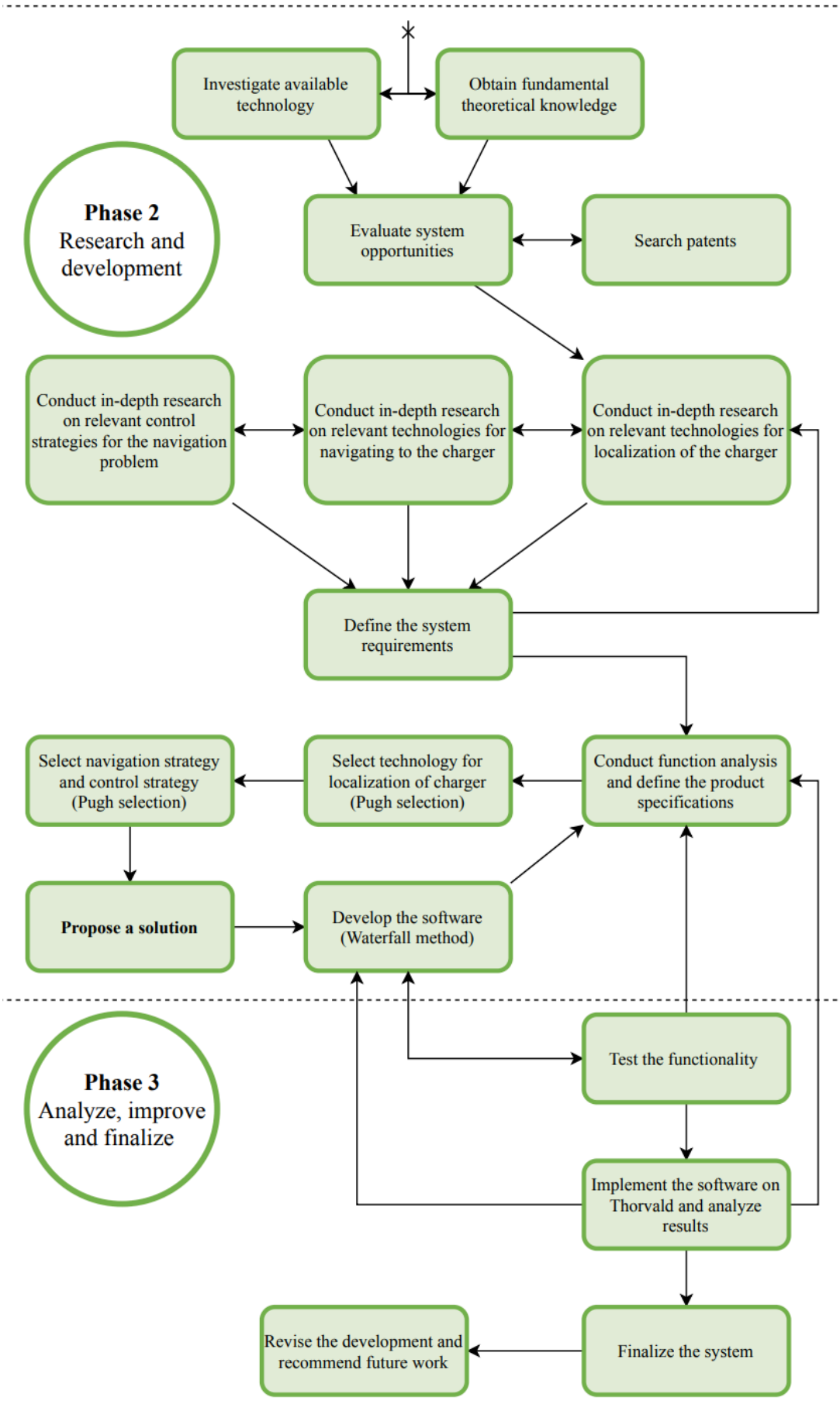


Figure 3-6: A flowchart that shows phase 2 and 3 of the process.

4 State-of-the-art and technology trends

This chapter provides an overview of existing problems and solutions for autonomous navigation of mobile robots.

Navigation is a problem that requires a large amount of attention when developing autonomous mobile robots, as both accuracy and precision are essential. To ensure precise and accurate navigation, robust localization is crucial in any environment. In many cases, localization is dependent on satellite systems, such as GPS. However, now that mobile robots and autonomous cars need to operate in both urban and indoor environments, new challenges occur [36].

Robot localization defines a robot's ability to estimate its position. The task can be performed in many ways, for instance, by using information that is provided by a map [37]. However, to map an environment, a robot must have information about its position, thus creating a coupling of mapping and localization. Several methods have been developed to solve problems due to this coupling. SLAM, simultaneous localization and mapping, is a common method that allows the robot to learn a map and estimate its position by using a continuous flow of sensor data [38].

Navigation is also necessary to enable a robot to autonomously recharge. Various existing platforms such as versatile robots, robot vacuums, and robot lawn mowers include functionality that allows them to both localize and navigate to their charging stations.

For a large number of modern robotic lawnmowers, a charged wire is used to create boundaries for the operational area of the robot. Older robot lawnmowers also use an embedded wire to guide the robot to its base station. Today, however, it is more common for lawnmowers to use, for instance, Bluetooth and cellular beacons to localize and navigate to their charging stations. Active beacons provide more precise position estimates and smarter ways for the robot to communicate with its base. Examples on robots that use such technology for docking are shown in Figure 10, where the bottom image shows the robotic lawn mower Bosch Indego 350 docking to its base station [39].

Figure 4-1: Illustrations of robots docking. Top left: WiFiBot QR docking. Top right: Roomba i7+. Bottom: Bosch Indego 350.

Indoor robot vacuums, such as the Roomba i7+ [40], also shown in Figure 4-1, use SLAM to map their environment and enable continuous localization. For autonomous docking,

most models use infrared beacons to communicate with their base station, where the robots have receivers that respond to signals from transmitting beacons mounted on the dock.

Another example of a technique that is used for autonomous docking is shown in Figure 4-1 in which the WiFiBot docks by using computer vision to recognize QR-codes. By localizing the QR-codes, the robot can generate a path back to its dock. For the final step of the docking, proximity sensing is used to ensure accurate alignment.

Other existing solutions use tactile sensors, embedded wires, heat, or magnetic tape. However, not all solutions are applicable in any environment. For instance, computer vision relies on sufficient lighting, whereas cellular, Bluetooth, and infrared communication, often rely on a free line of sight. Because the various methods are dependent on environmental conditions, it is necessary to evaluate the technique based on the intended application to determine if it is appropriate.

This chapter has presented existing solutions that can be used as inspiration for the development of the autonomous navigations system. The theory and technology that is relevant for the development is presented in the next chapter.

5 Theory and technology

This chapter explains the technologies and corresponding theoretical principles that apply to the problem of developing a navigation system for docking of Thorvald.

5.1 Navigation

Several elements are required to enable a robot to navigate freely in an environment. A map is required as it contains information that is crucial for robots to accomplish their given tasks. Maps can either be metric, topologic, or hybrid, as a combination of both. Metric maps provide information about geometric characteristics of an environment, whereas topological maps are simplified and only contain crucial information, such as the relationships between points in space. Topological maps often have lower resolution, while metric maps have a higher resolution, where higher resolution often is related to greater computational complexity.

A localization module is also needed for the robot to estimate its pose relative to the map or a frame of reference. Localization can be divided into two separate problems; pose tracking, which requires information about an initial position, and global localization, for which no initial position is known.

Given a map and a localization module, it is possible to generate a path to the desired goal location, if it exists. The motion planning problem is a crucial problem in mobile robot navigation and involves the construction of feasible paths and trajectories. The following sections describe theoretical principles that make motion planning possible.

Path and Trajectory Planning

Path planning is a task for which the aim is to determine a feasible and continuous path between two points in space. A wide variety of sensors can be used by robots to perceive their surroundings, and examples of such sensors are cameras, lidars, and proximity sensors. These devices provide data that robots can use to map obstacles and generate feasible paths [41].

While path planning enables the robot to discover paths in the operating environment, trajectory planning connects the robot's state to this path, which allows for optimization of, for instance, execution time or energy consumption.

To understand the path planning problem, it is necessary to define the following environments:

Configuration Space (C_{space}): The configuration space defines the space where any robot configuration is possible.

Obstacle-space, ($C_{obs.}$): The obstacle space defines the part of the configuration space that is occupied by obstacles.

Free-space (C_{free}): The free-space, or space of free configurations, defines the part of the configuration space in which robots can move freely. This environment is often described as:

$$C_{free} = \{C_{space} - C_{obs.}\}$$

The path planning problem considers generating a geometric path within c_{free} so that the robot can move freely from A to B without colliding with any obstacles. The various spaces are depicted in Figure 5-1.

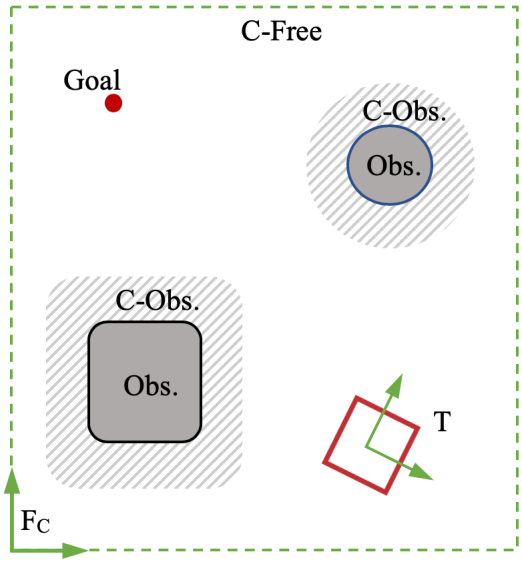


Figure 5-1: An illustration of the different environments considered in path planning. the dashed green line illustrates the boundaries of the C-space. T: Thorvald, F_C: Fixed frame for C-space.

Path planning can be divided into three different categories; *sensor-based planning*, which is planning that relies solely on the robots' sensors, *map-based planning*, which is planning based information provided by a map, and *combined planning*, which is a combination of the two previous. There are several ways to obtain the desired geometric path, and techniques are usually divided into three main techniques; *roadmap techniques*, *cell decomposition algorithms*, and *artificial potential methods* [42], [43] .

5.2 Positioning and localization

Accurate information about the robot's position is a main problem for mobile robots, and numerous techniques have been developed to enable robots to get exact information about their positions [44], [45]. Positioning and localization are both crucial for mobile robot navigation.

Positioning provide data containing information about the robot's coordinates in a local or global frame, whereas localization defines the process of locating these coordinates in a frame or a map. The three main parameters that describe the pose of a robot in a 2D world are x, y, and ψ, respectively the x-coordinate, the y-coordinate and the heading. For straight-line motion, incremental displacement can be used to update the robot's coordinates, as is depicted in Figure 5-2. In a two-dimensional world, the pose of a simple kinematic model can be updated with the relationships given by the two equations on the next page.

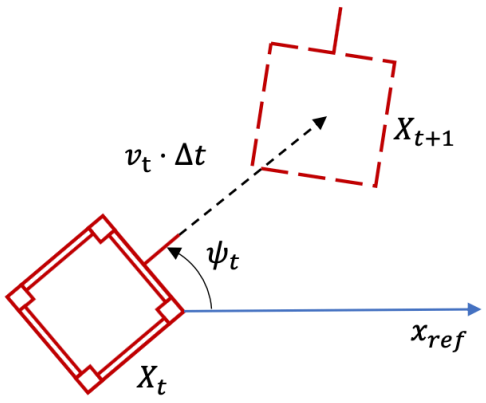


Figure 5-2: An illustration of pose estimation over time. X_t is the initial pose, ψ_t is the heading and v_t is the velocity at time t. X_{t+1} is the pose at time $t + 1$.

In Figure 5-2, X_t and X_{t+1} represent the pose of the robot given by its x- and y-coordinates, and its heading at time t and $t+1$, respectively. v_t represents the velocity at time t.

$$X_t = \begin{bmatrix} x_t \\ y_t \\ \psi_t \end{bmatrix} \tag{5.1}$$

$$X_{t+1} = X_t + \begin{bmatrix} v_t \Delta t \cos(\psi_t) \\ v_t \Delta t \, sin(\psi_t) \\ \dot{\psi} \Delta t \end{bmatrix} \tag{5.2}$$

The heading information for the robot can be obtained using multiple methods, where the most common methods involve using steering angle sensors, magnetic compasses, or calculations from differential odometry [46].

It is common to divide positioning methods into two different categories; *Relative position measurements* and *absolute position measurements*, that both are explained in the following sections.

5.2.1 Relative position measurements

Dead reckoning is a localization method that uses a simple mathematical procedure to perform relative position measurements. Advancement of previous knowledge about the position with velocity, bearing, and time information is used to determine the position of the robot. Odometry is an example of an implementation of dead reckoning and suggests that the distance traveled by an object can be derived directly from an onboard odometer. However, as dead reckoning methods rely on previous estimates, there is a high probability for cumulative errors that may propagate and decrease the accuracy of the positioning [46].

Another dead reckoning method uses an INS, Inertial Navigation System. An INS integrates data from its IMU, Inertial Measurement Unit [47], which with its accelerometers and gyroscopes measures linear acceleration and angular rate. By integrating this information, the inertial navigation system provides estimates for position, velocity, and attitude. However, as a result of integration, inertial measurements may drift over time, making inertial navigation less appropriate for long-time operation. Accelerometers are also subject to noise, especially for low accelerations, and may require robust filtering to provide accurate estimates [44].

5.2.2 Absolute position measurements

In contrast to dead reckoning, absolute position measurements do not rely on previous estimates, which reduces the risk of error cumulation. Two common techniques for absolute positioning are trilateration and triangulation.

Trilateration

Trilateration is a technique that provides accurate information about a position by using distance measurements between a reference and a set of known objects or beacons. For instance, three or more transmitters with known positions can be used as anchor points,

while a receiver can be mounted on a robot to collects signals. By using time-of-flight calculations, information about the distance to the anchor points can easily be obtained. The system can also be mounted conversely, with a transmitter on the robot and three receivers mounted at anchored positions. Three anchored nodes with known locations allow for 2D positioning, whereas four nodes are needed for positioning in 3D [44], [48].

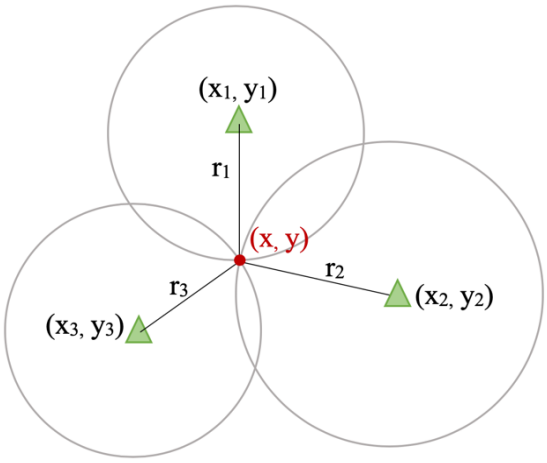


Figure 5-3: An illustration of the trilateration principle. The green triangles represent anchored points with coordinates (x_i, y_i) and distance r_i for $i \in (1,3)$ to a red point with coordinates (x, y) which is the point for which the position needs to be

The trilateration technique is depicted by Figure 5-3, where green triangles represent anchored transmitters. The signal from a transmitter can, in free 2D space, be visualized as a circle with an increasing radius. Let the red dot in the figure represent an object that needs to be localized. When the radius of the circle has increased so that the object is placed somewhere along its circumference, the distance to the object is known. By utilizing three transmitters, the position of the object in 2D space, (x, y), can accurately be determined by solving the equations given by the relationships below.

$$\begin{bmatrix} (x_1 - x)^2 + (y_1 - y)^2 \\ (x_2 - x)^2 + (y_2 - y)^2 \\ \vdots \\ (x_n - x)^2 + (y_n - y)^2 \end{bmatrix} = \begin{bmatrix} (r_1)^2 \\ (r_2)^2 \\ \vdots \\ (r_n)^2 \end{bmatrix} \tag{5.3}$$

Trilateration is a principle that is, for instance, used by a GNSS, Global Navigation Satellite System [49], where four or more satellites work as transmitting anchored nodes. Although trilateration provides good position estimates, the calculation does not directly provide information about the heading of the robot. In other words, the orientation remains uncertain and other techniques are necessary to obtain sufficient information.

For GNSS, mounting a second receiver somewhere in the XY-plane of the robot, with sufficient spacing, will make it possible to calculate the heading angle of the robot. However, robots tend to be rather small and will, hence, require Real-Time Kinematics GNSS receivers (RTK-GNSS) that provide high precision (1-2 cm). These sensors are extremely expensive and require correction data from a base station. In many cases, the use of other techniques may, therefore, be more beneficial than implementing additional sensors.

Triangulation

Triangulation is a method that uses both the distance and angle to known landmarks to obtain information about the pose of a robot, or an object of interest. Compared to

trilateration, triangulation utilizes not only the distances to the landmarks but also the angles and, thus allowing for determination of orientations [44].

Similar to trilateration, triangulation requires at least three anchored references. The technique can be performed, for instance, by mounting a rotating sensor, such as a laser scanner or an infrared receiver, onboard a robot to collect relative position information about the anchored references. These sensors can provide vectorial information from which angles and distances to the reference nodes can be extracted.

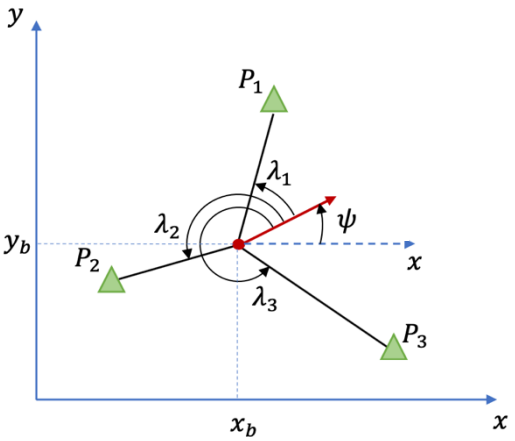


Figure 5-4: An illustration of the triangulation problem. Three landmarks marked by P_i are sensed at angles λ_i for $i \in (1,3)$ with a sensor on a robot with pose (x_b, y_b, ψ).

Figure 5-4 provides an illustration of the triangulation principle for determination of the pose of a robot. The position of the robot is given by (x_b, y_b), the heading by ψ, three references are given by P_i for $i \in (1,3)$, and the angles between these and the robot's longitudinal axis by λ_i for $i \in (1,3)$ [48].

The technique can also be performed with computer vision, where distinctive features can be used to define landmarks. The distance and angles can be calculated by using analyzing relationships in images, or by extracting data from a point cloud provided by a depth camera.

When utilizing triangulation, it is important to be aware of its limitations, which, for instance, are reduced accuracy with higher ranges, angle measurement errors, and shadowing, which means that reference nodes remain invisible.

Landmark-based positioning

Another absolute positioning technique considers the use of landmarks. QR-codes, physical geometries are examples of landmarks with distinct features that can be easily recognized. Landmarks used in robot navigation usually have fixed locations and can, hence, be used by the robot to locate itself or to localize points of interest. A great challenge in landmark navigation is to ensure that the landmark recognition is robust so that, for instance, the robot's position accurately can be determined. Landmark navigation is, however, often used alongside other localization methods to provide more accurate and precise pose estimates [44].

Map-based localization

The position of a robot or an object can also be determined by using map-based localization. Map-matching is a technique where sensors, often lasers, are used to map smaller parts of the robot's surroundings. A small, local map is created and then compared to an existing

map of the same environment in search of matching features. If matching features exist, these are used to determine the robot's location on the map.

5.2.3 SLAM

SLAM, simultaneous localization and mapping, is a method that enables a robot to generate a map of the surrounding environment and estimate its position simultaneously. A common way to perform SLAM for a robot is by using laser scanners to map the environment and odometry to track the pose of the robot. The pose is tracked and refined by laser measurements and then expressed in the constructed map. Simultaneous localization and mapping can be done with many different approaches, and multiple methods can be used for both mapping and pose tracking. Common for most SLAM algorithms are, however, that they use probabilistic approaches to learn a map and to track the pose.

5.3 Sensors for navigation

Perception and localization are crucial for safe navigation. A sensor is a device that measures physical quantities such as pressure, acceleration, or humidity. Sensors can be divided into two main categories; active sensors and passive sensors, for which active sensors use an external power source to produce signals that it can measure reflections of. Examples of active sensors are lidars and ultrasonic sensors that use the reflection of electromagnetic and mechanical waves to perceive the environment. Passive sensors do not transmit signals and only measure what is already in the environment, such as, for instance, a camera [46].

Sensors can be further classified as proprioceptive and exteroceptive, where proprioceptive sensors are sensors that measure internal values, and exteroceptive collect information from the surrounding environment [50]. In the following sections, a general description of a handful of sensors will be provided.

5.3.1 IMU

IMU, or Inertial Measurement Unit, is a complete system that allows for the measurement of linear and angular motion [47]. A common assembly for an IMU is a triad of accelerometers along with a triad of rate gyroscopes. The rate gyroscopes measure the angular velocity in degrees per second and can output angular motion $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ of a body. Examples of IMUs are shown in Figure 5-5.

Figure 5-5: The ISM product line from XSens (xsens.com).

Accelerometers, on the other hand, measure linear accelerations, for which the integrated measurements can be used for estimates for linear velocity, and the double integrated values for displacement. However, signals from both accelerometers and rate gyroscopes are often

noisy and result in inaccurate measurements. An IMU is, therefore, not optimal for state estimation on its own, and is often part of complete systems.

Inertial navigation systems (INS) are systems that are constructed to keep track of the measurements from inertial sensor units. An INS includes a processing unit that keeps track of, for instance, the integrated measurements from the IMU. For indoor navigation, complete systems, such as INS and odometry, are often used together with either computer vision or lidars to solve problems due to the absence of satellite signals [51]. For outdoor navigation, INS and odometry are often used alongside GNSS to provide robust position estimates.

5.3.2 Odometers

Odometers are sensors that provide information about the relative position and translation of an object. Common odometers are optical encoders, which are devices that enable angular displacement to be converted into digital data. Optical encoders focus beams of infrared light, for instance, towards a rotating disk connected to an axle or the inside of the wheel and use the reflection to measure the angular displacement. The sensors can also transmit light through a grated disk and operate in the company of a receiver. Given knowledge about the radius of the wheel or axle, it is possible to transform the angular displacement into an estimation of the traveled longitudinal distance. The distance traveled by one wheel can be derived from the following equation:

$$D = 2\pi r \cdot \frac{n}{n_{tot}} \tag{5.4}$$

where D is the estimated distance, r is the radius of the wheel, n is the number of measured reflections, and n_{tot} is the total number of detection points. By including the encoder data for all wheels, the relative displacement of an object can be calculated.

5.3.3 Lidar

Lidar is an acronym for Laser Image, Detection and Ranging, or Light Detection and Ranging. Lidar is an active sensor that uses the reflection of optical laser pulses to collect data about the environment. The frequency of the pulses depends on the lidar specifications, but for some lidars, the frequency can be as high as 150.000Hz. The physical and mathematical theory that is the base for lidars' functionality is rather straight-forward and revolves around the following equation:

$$d = \frac{c \cdot t}{2} \tag{5.5}$$

Where d is the measured distance to an object, c is the speed of light and t is the time of flight for the laser pulse. Lidars are often used for 2D- and 3D-mapping, digital modeling and in navigation, and often designed to measure with a view of as much as 270 degrees.

Figure 5-6 depicts how lidars work. The laser emits pulses of light at a very high rate, often millions of pulses per second, towards the surrounding environment. As long as there are no obstacles, the photons will travel freely through the air. However, when the light encounters a physical obstacle, the photons reflect.

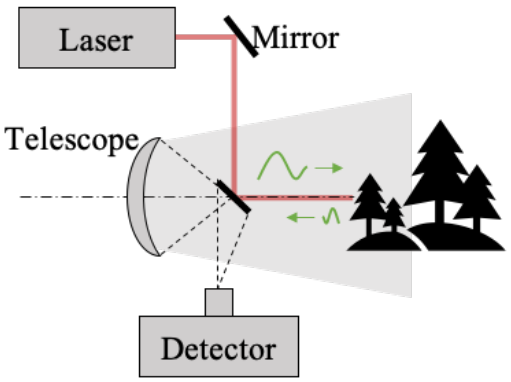


Figure 5-6: An Illustration of how lidars work. A beam is emitted from the laser and redirected by mirrors towards the environment. The reflected beams are caught by a telescope that directs the reflections towards a detector before the necessary calculations are done.

A telescope in the lidar is constructed to collect the reflected photons and direct these towards a detector. Based on a timestamp, and Equation (5.5), the distance traveled can easily be calculated. 3D-lidars translates the millions of distance measurements into a 3D point cloud, which can be described as a complex 3D map of the environment. 2D-scanners also generate point clouds, but only in two dimensions, thus often used for ranging and 2D mapping [52].

Lidars can determine positions with high precision, making the sensor very suitable for positioning in the absence of satellite systems [53]. For outdoor operation, however, weather conditions such as rain and heavy fog may pose challenges to the capabilities of the lidar making accurate measurements. Current state-of-the-art lidars run with an electromagnetic wavelength of either 905nm or 1550nm, but tests show that a wavelength of 905nm is a more weather-proof choice as water absorption is greater for 1550nm [54].

5.4 Computer vision

Computer vision is a field within computer science in which various techniques are used to enable computers to understand the content of digital images. Computer vision aims to reproduce the capabilities of human vision by processing digital image data. Three-dimensional metric information and semantic information can be extracted from an image and related to what is seen by a processing device, which allows for a robot to recognize objects or geometries in the environment. Recognized features may be used to, for instance, guide the robot, or to localize landmarks [55].

In advanced driver assistance systems, ADAS, computer vision is, for instance, used to recognize lanes for lane-keeping assistants (LKA). Images provide information about a car's position relative to the lane so that potential position or heading errors can be detected and corrected by control commands.

Computer vision has a wide range of applications and can be used, for instance, in industrial inspection or medical image analysis. For mobile robot navigation, computer vision can be applied to perform various tasks, such as obstacle detection, ranging, or localization of landmarks.

5.4.1 Localizing objects with shapes and computer vision

Object detection is a set of techniques within computer vision that allows for the recognition of instances or objects in images or videos. Detection is used when the desired result is to localize a known feature in an image. The task can be performed either by using classical computer vision approaches such as Hough transforms [56], or through machine learning or deep learning. Deep learning is a subset of machine learning and involves techniques that utilize artificial intelligence and sets of neurons to mimic the human brain for decision-making [57]. Handcrafted computer vision approaches can also be combined with deep learning strategies for increased robustness in high-performance systems [58]. Examples of detectors that use deep learning are R-CNN [59] and Yolo v3 [60], [61].

A general workflow for computer vision systems can be formulated given by Figure 5-7 below:

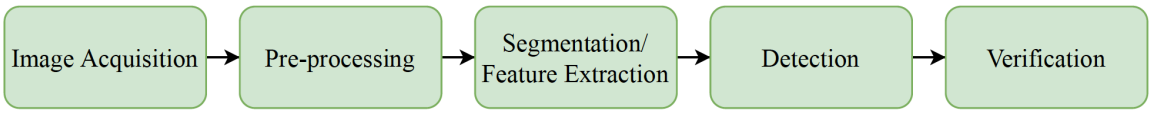


Figure 5-7: An illustration of a general workflow for a computer vision detection system.

5.4.2 RGB-D camera

RGB is an acronym for red, green and blue, and represents a color space. An RGB image may contain thousands to millions of pixels, which are all assigned with an RGB code (R, G, B), for which each element usually ranges from $(0, 255)$. For instance, a completely white pixel will be represented as $(255, 255, 255)$, whereas a red pixel will be represented as $(255, 0, 0)$ [56].

RGB cameras store images as pixels with coordinates, (x, y), in a 2D plane. RGB-D cameras, however, also include the z-axis which represents the depth in the image. The z-coordinate allows for the creation of 3D images and point clouds, which can be useful for tasks such as 3D-mapping, robot navigation, and augmented reality [62].

The depth in an image is often obtained by using active stereo vision, which is a technique that utilizes two image sensors to mimic human binocular vision. By comparing two images, one obtained by the right image sensor, and one by the left, disparities can be mapped, and with information about the focal length (f) and the baseline between the two cameras (B), triangulation can be used to extract distance information from the measured parallax. The process is depicted by the illustration in Figure 5-8 on the next page [63].

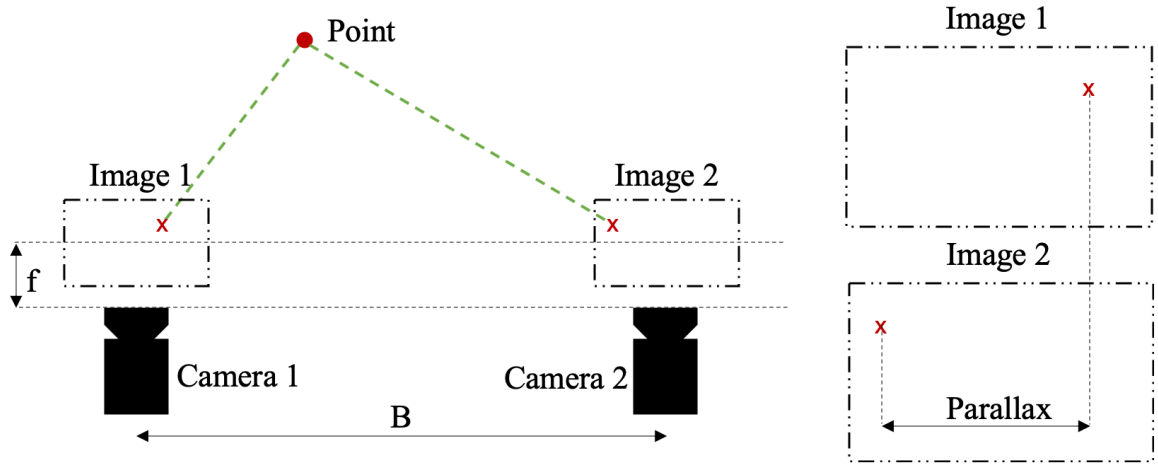


Figure 5-8: An illustration that shows the principle of stereoscopic vision, where two cameras capture images of the same point. f: focal length of the camera. B: Baseline distance between the two cameras. Parallax: The distance between the pixels that represent the relevant point in the respective two images.

5.5 Beacons

Beacons are devices that are made to attract attention to a specific location. For robot navigation, beacons can be used for robot positioning or determine the position of relevant targets.

In similar to sensors, beacons can be both active and passive, for which active beacons need transmit signals, and passive beacons do not. Bluetooth, infrared, and sonar beacons are examples of active beacons, and reflective and optical markers are examples of passive. Reflective markers can, for instance, be used to intensify the reflection of laser beams from lidars to make targets distinguishable. Beacons are commonly used to mark targets to use for trilateration or triangulation [44], [64].

5.6 Magnetic guidance path

In robotics, guidance paths are designed to guide robots to desired locations. By calculating the lateral error between a robot and the path, control commands can be generated to guide the robot.

Magnetic tape is a tape that generates a magnetic field around it. By using, for instance, a hall-effect sensor to measure the difference in the magnetic field density, the lateral distance to the center of the magnetic field can be calculated. Magnetic tape is inexpensive and can be easily mounted on floors with adhesives. The magnetic field generated by the tape is robust to both dirt and lighting conditions, which makes it appropriate for applications in various fields. Robots that navigate based on guidance paths are, however, very dependent on the path that is constructed and cannot easily change their path, for instance in the presence of an obstacle [46], [65].

5.7 Control theory

This chapter provides an overview of basic control theory and describes relevant controllers for the docking system. In this chapter, "vehicle" and "robot" will be used alternately.

5.7.1 Dynamic systems and feedback

A dynamic system is a system for which the behavior changes over time, often due to external forces or interferences. The aim is to control this behavior so that external stimulation does not affect the process of the system. The system whose behavior is desirable to control is often referred to as the *plant.* However, the terms *system, process,* and *plant* are used interchangeably. External impacts on dynamic systems are usually expressed as inputs, which can be generated either by an operator as control inputs or as non-controllable, external disturbances. Systems that consist of two or more coupled processes are often referred to as *open-loop feedforward systems* or *closed-loop feedback systems*.

The term *control* describes the use of control strategies, or control laws, to manipulate inputs so that a system acts as desired. Control strategies are used to drive the parameters of a system to desired values. For instance, regulate the temperature in a refrigerator, the velocity of a car, or the attitude of a drone.

It is important to distinguish between open-loop systems and closed-loop systems when designing a controller. Open-loop systems are simple, and control input values are decided ahead-of-time. These systems rely on calibration and cannot deal with real-time disturbances or changes. In other words, the input of the system is only a function of the reference signal, *r*, and the time, *t*.

$$u = f(r, t) \tag{5.6}$$

Closed-loop systems are, on the other hand, constructed with as a loop, in which every subsystem is dependent on another. Closed-loop systems allow for robust control that is less affected by uncertainties and external non-controllable disturbances. In closed-loop systems, the output is continuously measured by sensors to map the error between a desired and an actual value. The error is used to generate control commands to regulate the relevant process. Compared to open-loop systems, the input of a closed-loop system is a function of also the system output, y_m [66], [67]:

$$u = f(r, y_m, t) \tag{5.7}$$

This thesis will only consider closed-loop systems, and in Figure 5-9 on the next page, a simple schematic of a closed-loop system is shown. In the figure, r is the desired output of the system, C is the controller, u is the control command, and d is the external disturbances that affect the system. P is the process that is controlled, and n is the noise in non-ideal sensors that adds to the output, y, which is measured and fed back as y_m. y_m is then compared with the desired value, r, to determine the error, e, which closes the loop.

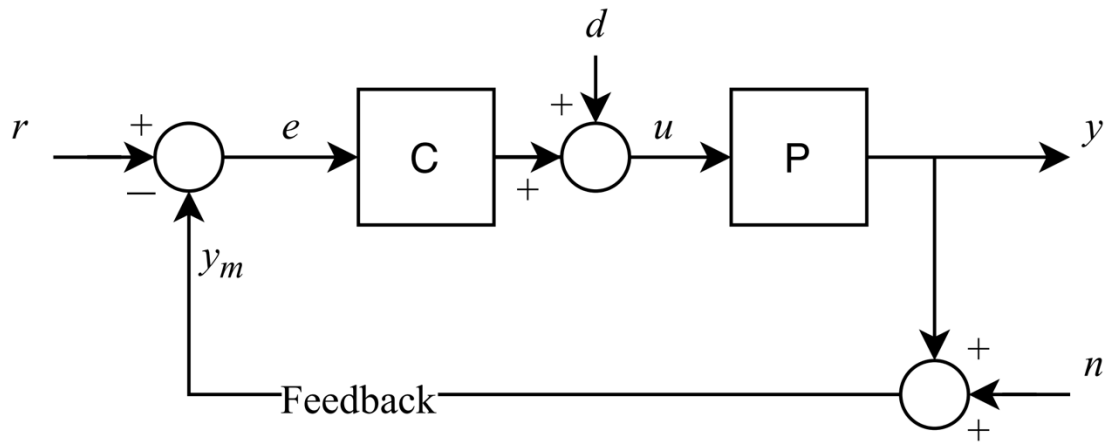


Figure 5-9: An illustration of a simple closed-loop system. The measured sensor value, y_m, is subtracted from a reference value which leads to an error. This error is used by the controller, C, to generate commands, u, to regulate the process, P. External disturbances, d, add to u and affects the output value of the process, y. Through a feedback loop, the measured values generate a new error. Sensor measurements are also subject to noise, here represented by n.

The aim of a closed-loop system is to drive the error, e, to zero. A descriptive example can be given through the principle of a cruise controller, which regulates the velocity of a car so that the actual velocity corresponds to the desired input from the driver. The vehicle system and output equations can be expressed as:

$$\ddot{x} = -\frac{k}{m}\dot{x} + \frac{1}{m}u \tag{5.8}$$

$$y = \dot{x} \tag{5.9}$$

where $\ddot{x}$ is the acceleration of the car, m is the mass, k is a constant multiplied with the velocity to consider, assumedly constant, forces from drag and rolling resistance, u is the throttle command, and y is the output of the system, which in this case is $\dot{x}$, the vehicle velocity. Imagine that the driver's desired velocity is the reference, r, for the system and y_m is the measured velocity that is calculated with encoders on each of the wheels. From y_m, the error is calculated as the measured velocity subtracted from the reference value.

$$e = r - y_m \tag{5.10}$$

If the measured velocity of the vehicle does not correspond to the desired velocity, the controller will generate commands to increase or decrease the speed.

Now, imagine that the car, for instance, drives towards a steep hill. The increase in inclination will increase the external forces working on the system and cause a reduction in velocity. The hill can be interpreted as a disturbance, d, that directly affects the process. However, due to feedback from the wheel encoders, the error increases, thus making the controller generate more aggressive throttle commands to maintain the desired velocity.

5.7.2 Controllers

A wide range of controllers exist and can be used to perform a great variety of tasks. For instance, to control the longitudinal motion of a robot, a simple option is to use a P-

controller, short for a proportional controller. P-controllers are often used in linear feedback control systems and generate their output by simply multiplying an input, usually, an error, with a constant, or gain. With a well-tuned constant, the P-controller can become suitable for longitudinal velocity control in simple robot operations.

The next equations describe a proportional control law for a simplified time-variant vehicle system given by the system equations as the ones used in the previous section with ideal values ($m = 1$, $k = 0$). The control command is given by the error between a desired velocity, r, and a measured velocity, y_m, as follows:

$$e(t) = r(t) - y_m(t) \tag{5.11}$$

$$u(t) = K_p e(t) \tag{5.12}$$

where $e(t)$ represents this error at time t, $u(t)$ represents a control command, a value for throttle or braking, and K_p represents a proportional gain. If the reference, r, is constant and the conditions are ideal ($d = 0$, $n = 0$), the control law $u(t)$ will ensure asymptotic convergence of the error $e(t)$ to zero, given that K_p is chosen to be positive.

Many different laws or strategies can be used to design controllers for both the longitudinal and lateral motion of robots. These controllers may be model-based and non-model based, describing their dependence on a model that represents the system. Common examples on control strategies are P-controllers, PI-controllers, PID-controllers, Linear Quadratic Regulators (LQR) and Model Predictive Control (MPC) [66], [68]. However, a study on each of these controllers goes beyond the scope of this thesis.

5.7.3 Kinematic bicycle model

The virtual kinematic bicycle model is a simplified kinematic model that can be used as a representation of a four-wheeled vehicle. Assuming that the vehicle has low-speed and slow-acceleration planar motion in 2D space and that the no-slip condition holds, a kinematic model can be constructed for a vehicle without any consideration of its dynamics. The kinematic model relies solely on the geometric relationships that govern the system. A kinematic bicycle model has shown to be a suitable model for the control of vehicle motion. An example of a kinematic bicycle model for a vehicle with Ackerman steering, meaning that only the front wheels can steer, is depicted in Figure 5-10. The model has its name from how it combines the two front and two rear wheels into one wheel, resembling the geometry of a bicycle. However, this simplification also involves assuming that the two front wheels steer with the same angle, which

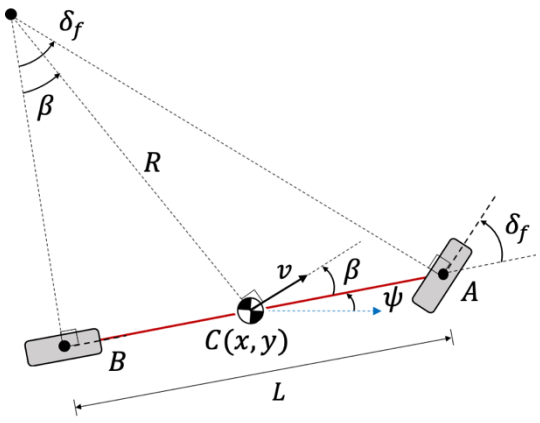


Figure 5-10: An illustration of the virtual kinematic bicycle model for lateral motion. A description of the parameters is given on the next page.

in reality is not true because they have different distances to the path's instantaneous center of rotation.

In Figure 5-10, both front wheels of the vehicle are combined and represented as one wheel at A, and both rear wheels at B. The front wheel steers with an angle δ_f to the longitudinal axis of the model. The pose of the vehicle can be represented by a vector $[x, y, \psi]^T$, where x and y represent the position of the center of mass, C, in the Cartesian plane, and ψ represents the heading of the vehicle in the reference frame. The vehicle drives on a curved path with O as the instantaneous center of rotation. The velocity vector for the center of mass is depicted by v, which makes an angle β with the longitudinal axis of the vehicle.

In reality, the direction of the velocity vector at wheel A is slightly different from the steering angle. This difference is due to wheel slip angles which occur when the direction of the velocity at each wheel differs from the direction of the wheel heading. However, vehicle dynamics theory, Rajamani [68], states that a no-slip condition can be assumed valid for low-speed operation, $v < 5\ m/s$, and, hence, that the velocity vectors located at each wheel are in the same direction as their orientation. The assumption is reasonable to make because the total lateral force working on each of the wheels is given by Newton's first law of motion for centrifugal acceleration:

$$F_L = \frac{mv^2}{R} \tag{5.13}$$

where F_L is the lateral force, m is the mass of the vehicle, v is the velocity at the center of mass, and R is the distance to the instantaneous center of rotation. At low speeds, the lateral forces are small and have no significant effects on the lateral motion. Because of this, F_L is often neglected for control design purposes.

When the no-slip assumption is valid, the following equations can be used to describe the motion of the vehicles center of gravity in the cartesian plane:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v \cdot \cos(\psi + \beta) \\ v \cdot \sin(\psi + \beta) \\ \frac{v}{L} \cdot \cos(\beta) \cdot \tan(\delta_f) \end{bmatrix} \tag{5.14}$$

Here, $\dot{x}$ and $\dot{y}$ represent the velocity along the x- and y-axis, respectively, whereas $\dot{\psi}$ represents the angular velocity, or yaw rate, and β is given by the following geometric relationship:

$$\beta(\delta_f) = \tan^{-1}\left(\frac{l_r}{l_f + l_r} \cdot \tan(\delta_f)\right) \tag{5.15}$$

where l_f and l_r are the distances from, respectively, A and B to the vehicle center of gravity.

5.7.4 Kinematic unicycle model

The kinematic unicycle model is a simplified construction of the bicycle model where the cycle has one instead of two wheels. The pose of the unicycle is complete when given the

vector $[x, y, \psi]^T$, in which x and y represent the unicycle's Cartesian coordinates and ψ its heading in the frame of reference. The kinematic model of the unicycle is given by the equation below, where v is the velocity of the center of gravity and ω is the angular velocity around the vertical axis which, for this model, is equivalent to the yaw rate, $\dot{\psi}$.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos(\psi) \\ \sin(\psi) \\ 0 \end{bmatrix} \cdot v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdot \omega \tag{5.16}$$

The kinematic unicycle model is, however, not only applicable to a system with one wheel, which in reality will have problems balancing under static conditions. The kinematic unicycle can also be used to represent differential drive robots [69]. The relationships given by the equation above for a robot with differential drive are depicted by Figure 5-11. Here, the pose of the robot with center of gravity at C is given by $[x_c, y_c, \psi]^T$, the velocity vector by v, while ω is equivalent to the yaw rate of the robot.

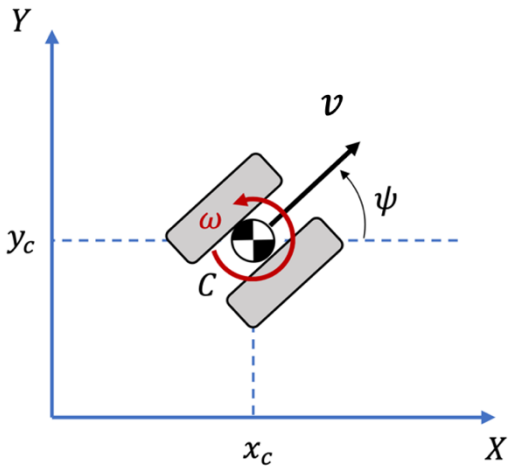


Figure 5-11: An illustration of the unicycle model for a robot with differential drive.

5.7.5 Geometric controllers

A geometric controller is a controller that relies solely on the kinematic model of a vehicle or robot. By designing a controller based on state feedback and geometric relationships in the kinematic model, it is possible to obtain the desired regulation of the motion of a vehicle [70]. However, to use geometric controllers, it is required for the vehicle or robot to have planar motion, and that the no-slip condition holds. In the following two sections, two geometric controllers are presented. First, a pure pursuit controller that relies on a reference trajectory, lookahead distance and curvature, and secondly, a pose regulator that enables the movement from one pose to another.

5.7.6 Pure pursuit control

A pure pursuit controller is a geometric lateral controller that is used for vehicles and robots to follow desired paths. The controller generates steering commands based on a cross-track error between the vehicle and the desired path. As the pure pursuit controller only regulates lateral motion, an additional control strategy must be added for longitudinal motion. Pure pursuit controllers ignore the dynamics of the vehicle and merely rely on geometric relationships. The controller is often used with the kinematic bicycle model and can, hence, only be used when the assumptions that were described in section 5.7.3 hold [68].

Figure 5-12 depicts the geometric relationships that are used to design a pure pursuit controller for a vehicle with Ackerman steering. Here, L represents the length of the vehicle, α represents the angle between the heading of the vehicle, ψ, and the target point, P_{target}, on the reference trajectory. x_{LA} is the lookahead distance which defines the distance from the reference point on the vehicle and the target point and defines the curvature of the road. δ is the steering angle of the front wheel, e represents the cross-track error, and R the radius to the instantaneous center of rotation in the system, C_R. v_f is the velocity vector of the front wheel.

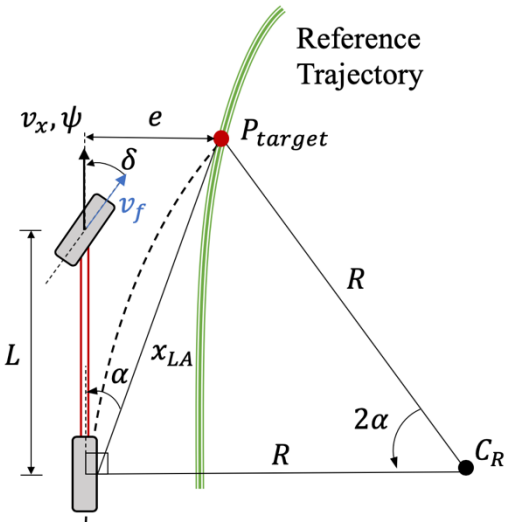


Figure 5-12: An illustration of principles for a path pursuit controller.

The lateral controller operates alongside a longitudinal controller to reduce the heading and cross-track errors:

$$\dot{\psi}_{des} - \dot{\psi} = \frac{v_f \cdot \sin(\delta)}{L} \tag{5.17}$$

$$\dot{e} = v_f \cdot \sin(\psi - \delta) \tag{5.18}$$

Let κ be the curvature of the path given by the law of sines (Formula 3.5) as:

$$\kappa = \frac{1}{R} = \frac{2 \cdot \sin(\alpha)}{x_{LA}} \tag{5.19}$$

and the steering angle, δ, define the arc radius so that:

$$\tan(\delta) = \frac{L}{R} \tag{5.20}$$

The performance of this controller is very dependent on the relationship between the look-ahead distance and the velocity. However, the look-ahead distance, x_{LA}, can be set as a linear function of the velocity, v_x, leading to the following relationship when combining (5.19) and (5.20):

$$\delta = \tan^{-1}\left(\frac{2 \cdot L \cdot \sin(\alpha)}{K_{pp} v_x}\right) \tag{5.21}$$

where δ is the steering angle, L is the length of the base, α is the angle to the target point from the reference point, K_{pp} is a constant positive gain and v_x is the longitudinal velocity of the vehicle [71], [72].

The pure pursuit controller provides lateral motion with two controllers, one for longitudinal velocity and one for steering angles. However, for many robots, it is desirable

to directly control the yaw rate, $\dot{\psi}$ (or ω). In the next section, a pose regulator that controls the longitudinal velocity v_x and the angular velocity $\dot{\psi}$ will be described.

5.7.7 Pose regulator

A pose regulator allows for regulation of the complete configuration vector of the robot, including both position and orientation. Due to the non-holonomic characteristics of the unicycle, a controller designed in Cartesian coordinates can only guarantee asymptotic stabilization for an arbitrary position. Hence, we use polar coordinates to design a control law that also allows for specification of a final orientation [69], [70].

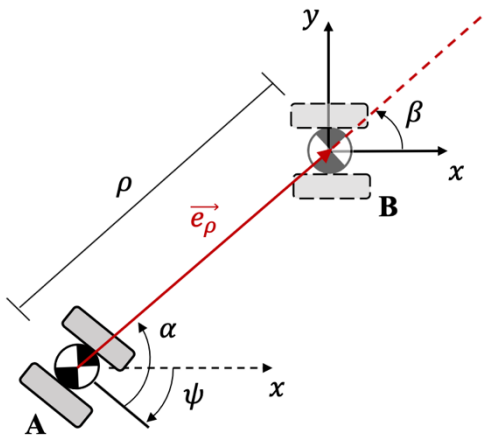


Figure 5-13: Definition of polar coordinates for the unicycle model. A: Initial configuration. B: Goal configuration at $[0, 0, 0]^T$.

Figure 5-13 depicts how the unicycle can be represented with polar coordinates, where ρ is the Euclidean distance between the reference point of the unicycle and the origin of cartesian plane, α is the angle between the vector $\overrightarrow{e_\rho}$ and the longitudinal axis of the robot, while β is the angle between $\overrightarrow{e_\rho}$ and the global x-axis. ψ is the heading of the robot with respect to the global x-axis. The following set of equations on the top of the next page describe the problem in polar coordinates:

$$\rho = \sqrt{x^2 + y^2} \tag{5.22}$$

$$\alpha = atan2\left(\frac{y}{x}\right) - \psi \tag{5.23}$$

$$\beta = \alpha + \psi \tag{5.24}$$

Based on these coordinates, the kinematic model of the unicycle can be expressed as:

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos(\alpha) \\ \dfrac{\sin(\alpha)}{\rho} \\ \dfrac{\sin(\alpha)}{\rho} \end{bmatrix} \cdot v + \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \cdot \omega \tag{5.25}$$

The kinematic model allows for control of the polar coordinates through control commands for longitudinal and angular velocity, v and ω respectively. This allows for the design of the following non-linear feedback control law which regulates the pose of the robot from $[\rho, \alpha, \beta]^T$ towards a desired pose at $[0, 0, 0]^T$:

$$v = k_\rho \cdot \rho \cdot \cos(\alpha) \tag{5.26}$$

$$\omega = k_\alpha \cdot \alpha + k_\rho \cdot \frac{\sin(\alpha) \cdot \cos(\alpha)}{\alpha} \cdot (\alpha + k_\beta \beta) \tag{5.27}$$

For small angles of α, $\alpha < \pi/10$, it is reasonable to make the following trigonometric assumptions:

$$\cos(\alpha) \approx 1 \tag{5.28}$$

$$\sin(\alpha) \approx \alpha \tag{5.29}$$

which allows for the use of the following linearized control law for small α:

$$v = k_\rho \cdot \rho \tag{5.30}$$

$$\omega = k_\alpha \cdot \alpha + k_\beta \cdot \beta \tag{5.31}$$

in which k_ρ, k_α and k_β are constant gains that are proven stable, by the Routh-Hurwitz theorem, under the following conditions [66], [69]:

$$k_\rho > 0 \tag{5.32}$$

$$k_\beta < 0 \tag{5.33}$$

$$k_\alpha - k_\rho > 0 \tag{5.34}$$

However, for greater angles, the non-linear strategy must be used, and in this case, the Lyapunov stability theorem proves the system stable for only positive values for the three gains; k_ρ, k_α and $k_\beta > 0$ [67], [69].

In this chapter, a brief explanation of relevant theory for the thesis has been presented. The next chapter provides a description of the software tools are utilized for this project.

6 Software tools

This chapter explains the software tools that will be used for the development of the autonomous docking system for Thorvald.

6.1 ROS, Robot Operating System

ROS, Robot Operating System, is the name of an open-source framework that is commonly used for robot software development. ROS was developed in 2007 by the Stanford AI Laboratory and has since 2013 been managed by Open Robotics [73].

The framework is built upon the four following elements:

- *Plumbing:* multiple programs can run simultaneously and communicate with each other. This feature allows for simple inter-process management and construction of distributed systems, which are systems constructed with several independent computers linked through a network.
- *Tools:* a wide range of tools are provided for various processes such as visualization, configuration, and debugging.
- *Capabilities:* ROS allows software developers to implement a wide range of functionality in their robots, such as controls, manipulation, mapping, planning, and perception.
- *Ecosystem:* ROS is open-source and has a large community that contributes to the development of the framework. There is a big focus on documentation and integration. ROS also includes tutorials that enable users to quickly become familiar with the system.

ROS is a peer-to-peer system which defines a distributed application architecture in which individual programs can communicate over a defined API (Application Programming Interface) through messages and services. Because of this, programs can be run on multiple computers, which reduces problems due to, for instance, storage and computational capacity. ROS is multi-lingual and comes with client libraries for C++, *roscpp* [74], and Python, *rospy* [75], but can be used with any language for which a client library can be made. ROS' overall aim is to simplify the process of creating truly robust, general-purpose software for a large variety of robots [76]. The framework is graphically depicted by Figure 6-1 below.

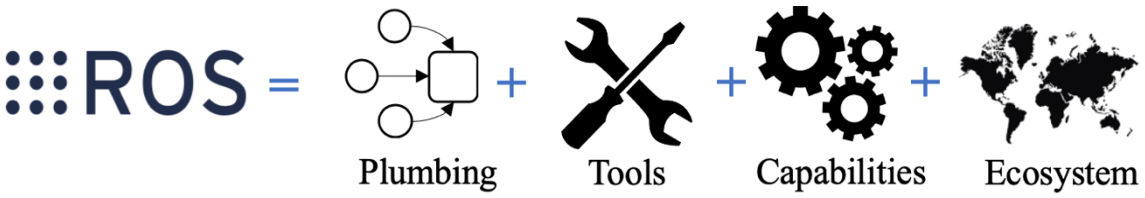


Figure 6-1: An Illustration of Elements in ROS with inspiration from ros.org.

6.1.1 ROS fundamentals

In ROS, nodes represent single-purpose executable programs that are individually processed [77]. The nodes are organized in packages and managed by the *ROS Master*, which controls the communication between active nodes. The flow of communications runs through topics. Topics represent streams of messages, to which the nodes subscribe and publish. These messages can include various types of information, such as, for instance, integers, floats, strings, or booleans.

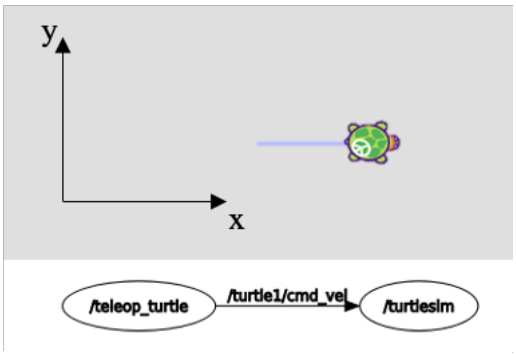


Figure 6-2: Illustration of a teleoperation node publishing messages to the turtlesim node through the cmd_vel topic. The teleop command received by the turtlesim node has, in this case, made the turtle move.

Figure 6-2 illustrates how nodes communicate in ROS. The *teleop_turtle* node is a node that publishes messages to the *cmd_vel* topic. *cmd_vel* is a topic that accepts *Twist* messages, which is a type of message that contains values for linear and angular velocity. The turtlesim node subscribes to the velocity topic and executes commands based on the obtained information. The structure of *teleop_turtle* - allows for physical control of *turtlesim*, where a pressed key on the keyboard changes values in the velocity commands published to *cmd_vel*. The white line in the illustration represents the distance the turtle has traveled in the x-direction as a result of a single push on the *up arrow* on the keyboard.

6.1.2 ROS actionlib

The ROS *actionlib* allows for the use of *ActionServe*rs and *ActionClients*. An ActionServer is an executable program that communicates with an ActionClient via the *Action Protocol*. The Action Protocol allows for communication through actions, which are messages that contain a *goal*, *feedback*, and a *result* that also allows for the client to receive continuous updates on the execution of an action, or to cancel requests [78].

A goal is defined by the client and sent to the server, and can, for instance, contain information about a desired pose or position. The feedback consists of a stream of messages from the server to provide the client with continuous updates on the action's progress. When the goal is achieved, a result is sent to the client. An advantage to ActionServers over ROS services, is that ActionServers do not restrict the robot from performing other tasks while the action is taking place.

6.1.3 Other relevant ROS packages

This section provides a brief overview of the contents of important ROS packages.

ROS Navigation Stack [79]

The ROS *Navigation Stack*, or *nav stack,* is a module that outputs translational and angular velocity commands based on input from odometry data, sensors, and a goal position or pose. The navigation stack is constructed of many nodes for which the main goal is to collaborate to navigate a robot from an initial to a final pose. Given a map, the *move_base* node can be used either with its internal planners, alongside nodes for robot localization to achieve a motion goal. In general terms, the navigation stack provides a complete system for safe navigation of mobile robots.

gmapping [80]

gmapping is a package that enables the use of SLAM to make maps and 2D occupancy grids. Maps can be saved and later used for navigation, for instance, along with the *navigation stack.* The node *slam_gmapping* uses laser scanner data and robot pose estimates to create grid-maps.

ROS tf: The transform library [81]

A package that allows the user to maintain an overview of existing coordinate frames in a system. *tf* provides the relationship between frames and allows for the transformation of points and vectors in between them. For example, coordinates related to a sensor's frame can easily be transformed into map coordinates by using ROS tf. Transformations can be obtained through the use of constructed broadcasters and listeners, in the same manner as ROS operates with publishers and subscribers.

rosbag [82]

rosbag is a package that provides tools for recording and playback of topics in ROS. The stream of data is stored in *bag* files, which is a ROS file format for storage of message data.

6.1.4 AMCL

AMCL, Adaptive Monte Carlo Localization, is a probabilistic localization approach that is commonly used for robots working in 2D environments. The strategy uses Monte Carlo sampling methods and a particle filter to estimate the pose of the robot with laser scan data that is compared to an existing map. A node that allows for adaptive Monte Carlo localization is also included in the ROS Navigation Stack [83].

6.1.5 ROS and Thorvald

Thorvald is fully compatible with ROS and is therefore also capable of navigating by using the Navigation Stack and its constructed nodes with a pre-defined map. By including the localization module *amcl,* Thorvald can localize itself by comparing sensor data from lasers to the given map. Using *amcl* allows for robust navigation with few errors as the localization modules update the pose of the robot continuously.

Although it is possible to use existing packages in ROS to enable Thorvald to navigate, it is also possible to design customized systems with desired sensors and functionality. The motion of the robot bases of Twist messages, which are messages that contain commands for linear and angular velocities. The message defines commands for linear velocities along the x, y and z axis (v_x, v_y, and v_z) and angular rates around them ($\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$) [84].

6.2 Simulation

6.2.1 Gazebo

The Gazebo simulator allows for the construction of complex, realistic worlds for simulation of mobile robots. Below, in Figure 6-3, is an illustration of a simulation environment that has been constructed in Gazebo. The environment is constructed with 3D models that are designed in SolidWorks and then exported to Gazebo models. The model of Thorvald is an example file provided by the robotics group at the University, and the trees are downloaded from Gazebo's model database. Simulations in Gazebo allow for efficient testing of new features and will, hence, be used to test software on Thorvald throughout the development process [85].

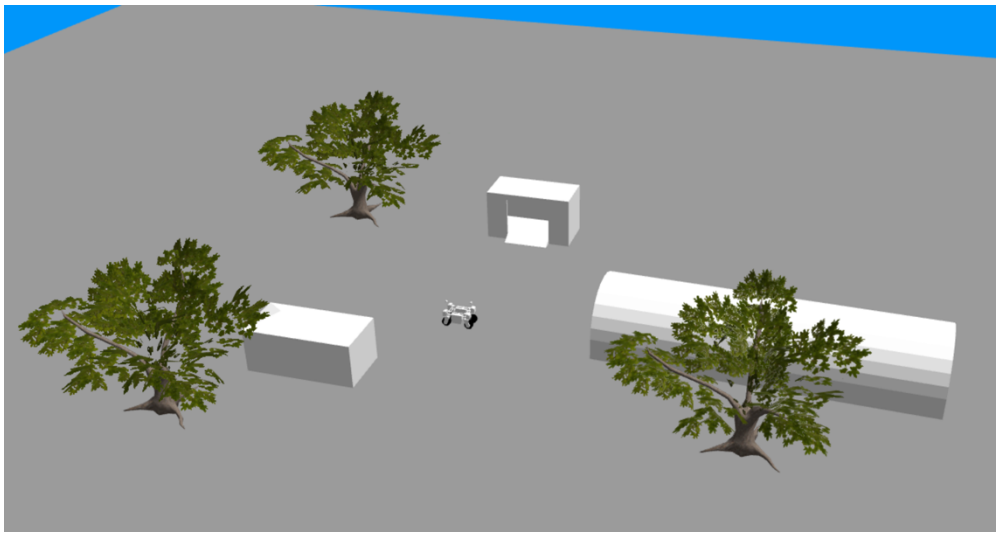


Figure 6-3: An illustration of a simulation environment made with inspiration from the test field at the University.

6.2.2 RViz

RViz is a tool for 3D visualization of robot operation. The tool allows for monitoring of what the robot perceives and does to make debugging easier and to be able to understand what the robot thinks. Furthermore, RViz also enables the user to maintain an overview of existing coordinate frames and topics that are active during operation [86].

6.3 OpenCV

The Open Source Computer Vision Library, OpenCV, is an open-sourced library for machine learning and computer vision [87]. The library includes a large number of

machine learning and computer vision algorithms that simplify perception tasks such as object detection and tracking, classification and face recognition. OpenCV also has a built-in interface that allows for development using Python.

HoughCircles

HoughCircles is a computer vision detection algorithm that can be used with OpenCV to detect circular geometries. To detect circles uses a modified version of the Hough Transform named the Hough Gradient, and Canny edge detectors to detect edges from gradient information in a photo [88]. The input to the function is a greyscale, preferably blurred photo, and the output is an array of the detected circles with their center given with coordinates in the camera frame [89].

Chapters 5 and 6 have explained theory, technology and software tools that will be used to develop the autonomous docking system for Thorvald. In the next chapter, specifications and goals will be defined for the navigation system.

7 Early system specifications

To select system components for the docking system that provide satisfactory performance, a set of requirements and specifications are presented in this chapter.

7.1 System goals and requirements

This section presents the main and intermediate system goals and the key system requirements for the docking system.

7.1.1 System goal

The main goal for the docking system is to enable Thorvald to autonomously navigate to its charger. Therefore, the system needs to include components for perception, path planning, and motion. In the next section, the system goal is broken down into sub-goals and system requirements.

7.1.2 Requirements

Table 7-1 provides an overview of system requirements and goals that have been set for the docking system. The requirements have also been assigned weight coefficients that represent their importance with respect to the system goal. A scale of 1 to 5 has been used, for which 1 represents non-critical, and 5 represents crucial requirements.

Table 7-1: An overview of the key qualifications for system and their corresponding goals.

Key system requirement	**Goal**	**Weight Coefficient**
Robustness	Make sure that the docking system is robust and that it can deal with disturbances.	2
Sensor technology	Make sure that the sensor technology selected for the docking system is appropriate, and that it provides adequate data for its given task.	4
Functionality	Make sure that the docking system provides the desired functionality, which, from the main goal, is to enable the robot to localize and navigate to the charger and align with it.	5
Transferability	Make sure that the docking system can be transferred to various configurations of Thorvald.	1
Integration	Make sure that the function can be easily integrated in Thorvald's system.	2
Feasibility	Make sure that the function can be developed within the given timeframe.	3

Table 7-2 provides an overview of the criteria that have been formulated for each system requirement. These criteria are formulated with *must, should* and *could* and to indicate their degree of importance in the development.

Table 7-2: An overview of the key system requirements for the docking system and their corresponding criteria.

Key system requirement	**Criteria**
Robustness	The function **should** be robust, thus not significantly affected by external disturbances such as incline, bumps or environmental forces.
Sensor technology	The selected sensor technology **must** be able to detect and track important objects or points. The selected sensor technology **must** be reliable for its operating environment.
Functionality	The function **must** be able to localize the charger. The function **must** navigate Thorvald to the charger. The function **must** align Thorvald with the charger within the tolerances that have been set. The function **must** navigate Thorvald through the gate of the charging station. The function **could** include a signal to open the gate of the charging station in cases where this is possible.
Transferability	The function **must** be transferable to different robot operating environments. The function **should** be transferable between different configurations of Thorvald with minimal adjustments. The function **could** be useable regardless of the charging station design.
Integration	The function **must** be integrable in Thorvald's existing system architecture. The function **should** be tested through simulation before deployed on the robot. The function **could** be made as an ActionServer that can be executed by Thorvald when necessary.
Feasibility	The construction of the function **must** be realistic within given timeframe.

7.2 Metric specifications

The following metric specifications hold for the Thorvald concept in general:

Table 7-3: Metric ranges for the dimensions of the Thorvald concept.

Specifications	Minimum	Maximum
Velocity	$0\ m/s$	$1.50\ m/s$
Acceleration	$0\ m/s$	$2.0\ m/s^2$
Dimension, length	$1500\ mm$	$1750\ mm$
Dimension, width	$1000\ mm$	$>3000\ mm$
Dimension, height	$825\ mm$	$\sim 2100\ mm$

It will be challenging to consider all the robot configurations in the design process. Therefore, it is decided to only focus on the standard configuration, but to maintain focus on transferability. This standard configuration of Thorvald holds the following specifications:

Table 7-4: Metric dimensions for the standard configuration of Thorvald.

Specification	Range/dimension
Longitudinal velocity (min – max)	$0 - 0.40\ m/s$
Angular velocity (min – max)	$0.1 - 0.5\ rad/s$
Acceleration (min – max)	$0 - 2.0\ m/s^2$
Dimension, length	$1500\ mm$
Dimension, width	$1500\ mm$
Dimension, height	$825\ mm$

Specifications are also needed for the operational environment to define metric tolerances for the system. The charging station that will be used as a reference when designing the system has the following metric specifications:

Table 7-5: An overview of relevant dimensions for the charging station.

Specification	Dimension
Gate width	$2300\ mm$
Gate height	$2000\ mm$
Inside minimum width	$2250\ mm$

Figure 7-1 depicts a potential design for a charging dock. This dock is constructed with pillars that, for instance, can be marked with reflective tape to be distinguished by a laser scanner. The specifications for the illustrated dock are given by Table 7-6.

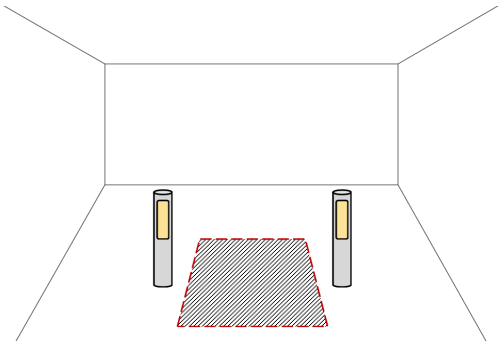


Figure 7-1: An illustration of the charging dock without the charging mechanism. The yellow fields on the pillars represent reflective tape.

Table 7-6: An overview of relevant dimensions for the charging dock.

Specification	Dimension
Distance between pillars	2200 mm
Pillar height	1200 mm

7.3 Tolerances

It is hard to design functionality with high accuracy and precision. Because of this, it is necessary to set acceptable tolerances for the system. Tolerances have been set for Thorvald's position and heading to ensure that a goal pose can be reached without the need for minimal and unnecessary final adjustments. The tolerances are provided in Table 7-7. Offset errors are given as radii of circles to define a tolerance field surrounding the desired or actual position.

Table 7-7: An overview of the tolerances set for the system.

Specification	Error
Heading error	$\pm 10\,°$
Position offset	200 mm
Detection offset	100 mm

The next chapter presents a functional analysis for the docking system that has been conducted to obtain an overview of the necessary functions that the system must include.

8 Function requirements and concept generation

This chapter presents a function analysis that has been conducted to determine the specifications for the system. These specifications are later used as criteria for a selection process where the aim is to propose a final and complete system solution.

8.1 Functional analysis

Function analyses are often used to reveal the necessary functions for a system to obtain the desired functionality. The function analysis for the docking system maps the necessary tasks that need to be performed by the system. The flow chart in Figure 8-1 provides an overview of the workflow for the autonomous docking of Thorvald. The grey-colored boxes represent tasks that are not considered in this development.

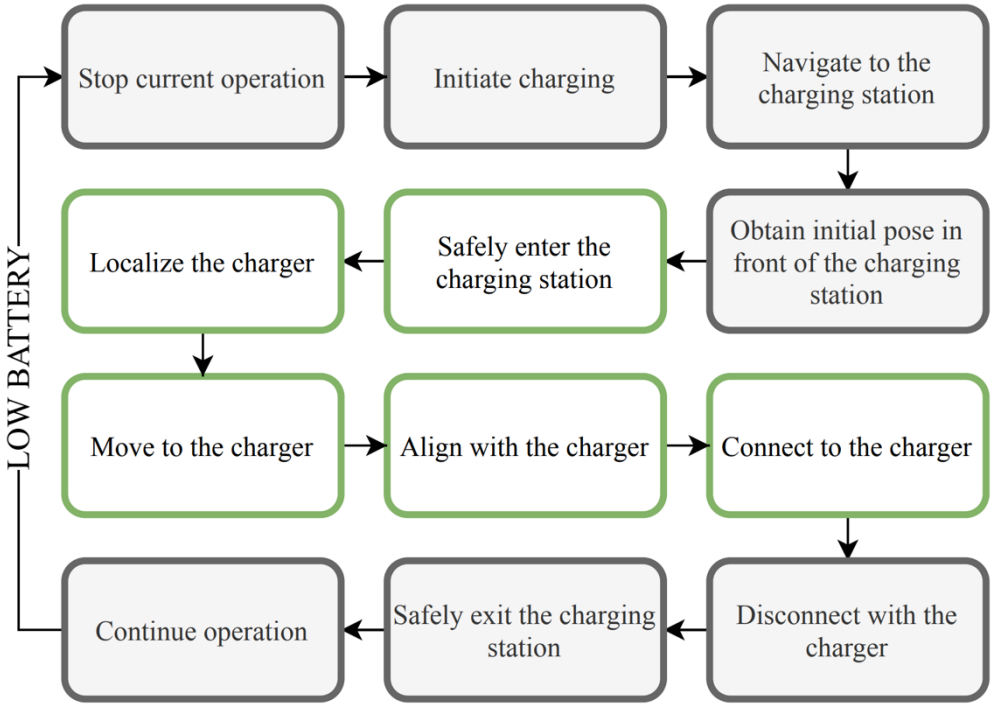


Figure 8-1: Functional step-to-step chart of Thorvald's workflow when the robot become low on battery.

The function analysis presented on the next pages is conducted to reveal the primary and secondary functions of the system, where the main goal is for Thorvald to autonomously dock inside the charging station. The analysis is presented with downwards-facing arrows to represent relationships between the various functions.

In the first part of the analysis, the primary goal has been broken down into manual and autonomous docking, as both methods can be used. However, since the system developed in this project is autonomous, manual docking has not been considered, which is indicated by the three dots in the first part of the analysis depicted in Figure 8-2 on the next page. The cross at the bottom of the figure represents a junction that connects part one with part two in Figure 8-3.

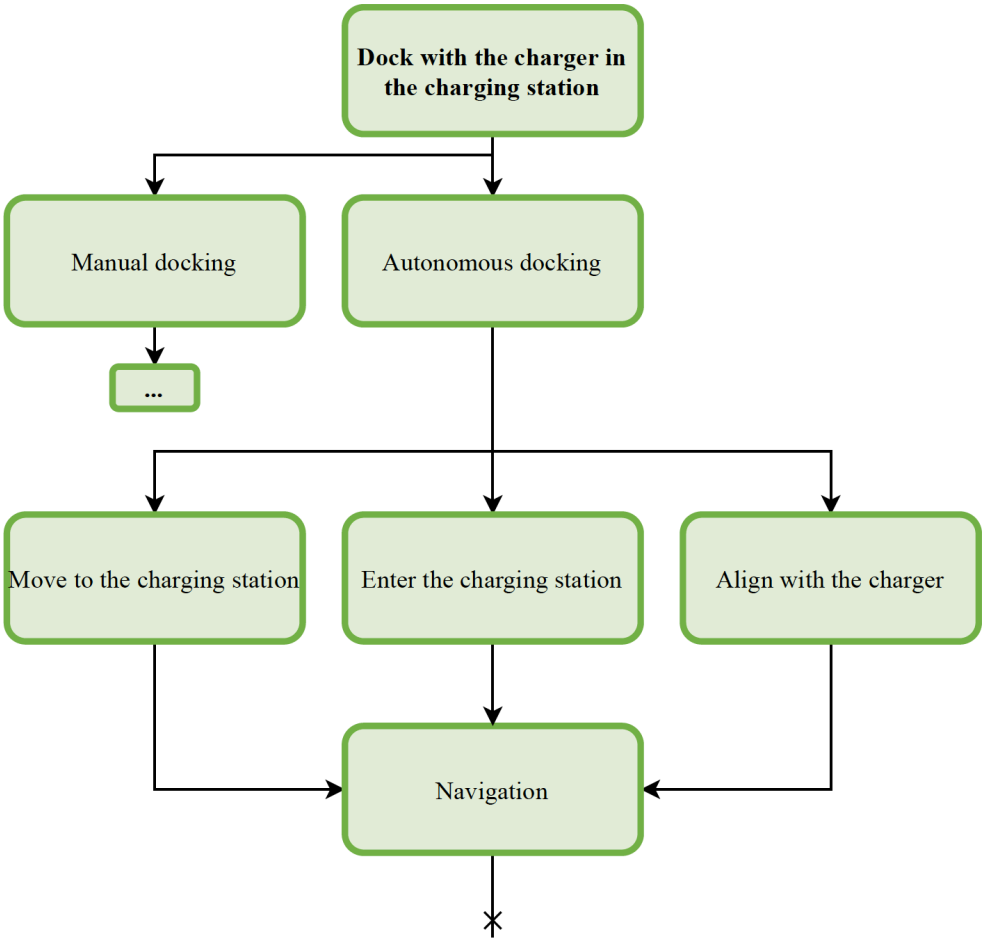


Figure 8-2: A flowchart showing part 1 of the function analysis.

Autonomous docking is the primary function of the system and can be split into three secondary functions; Move to the charging station, enter the charging station, and align with the charger. These secondary functions are all dependent on solving one common problem, namely navigation, which, in the second part of the function analysis, is broken down into two actions; path planning and path pursuit. These actions are broken down further into necessary tasks that need to be performed by the system. The final part of the analysis presents the fundamental necessities for the primary function.

The second part of the function analysis is depicted in Figure 8-3 on the next page before the secondary functions are described in more detail in the sections that follow the figure.

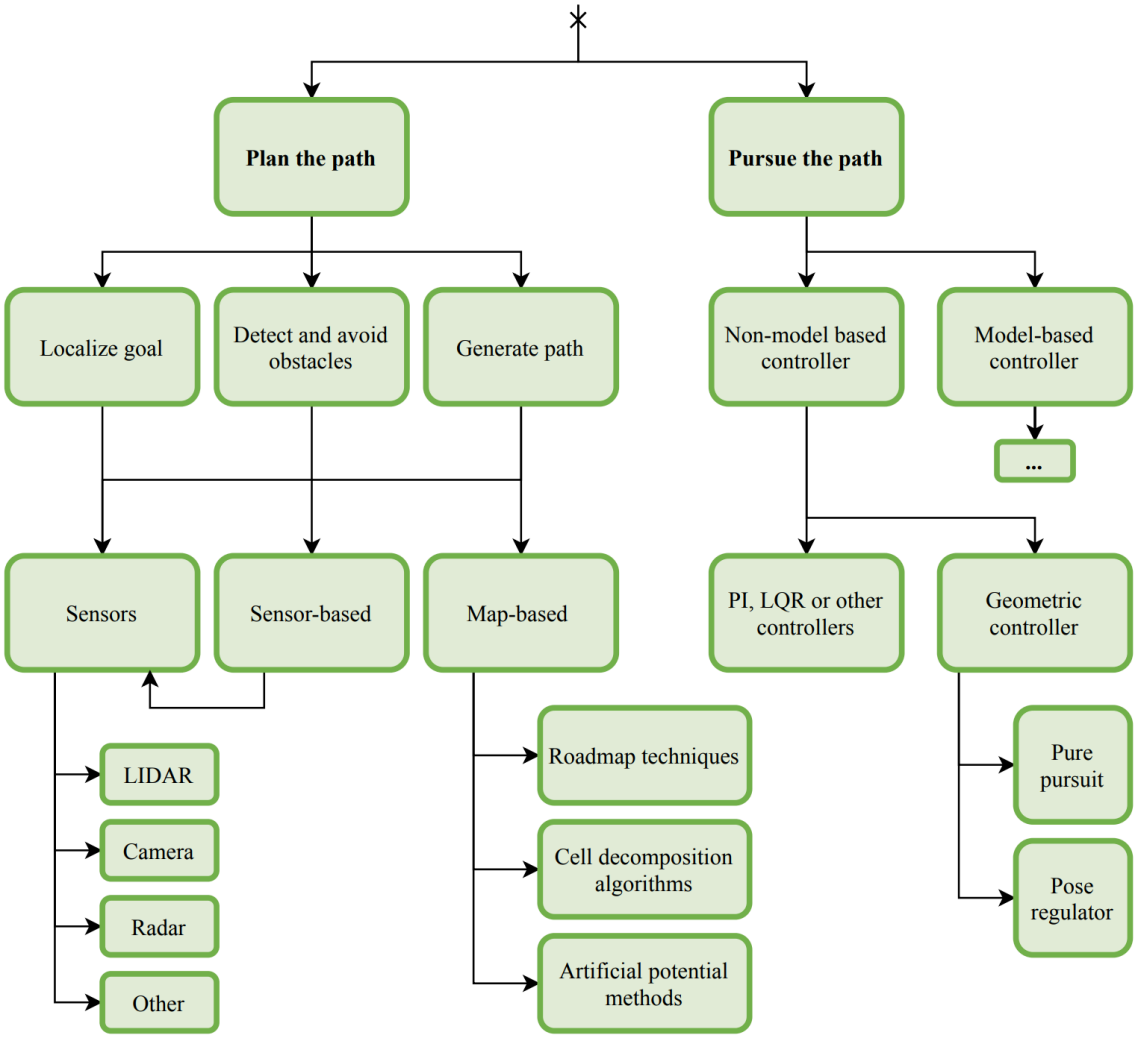


Figure 8-3: A flowchart showing level 2 and level 3 of the function analysis.

8.1.1 Navigating to the charging station

An existing navigation system that relies on a topological map will be used to perform the task of navigating to the charging station when Thorvald needs to recharge. A desired initial pose in front of the station will be defined as a node and achieved by the topological navigation system.

Table 8-1: Goal and needs for the action of moving to the charging station from an arbitrary point.

Move to the charging station	
Goal	Enable Thorvald to safely navigate to a desired pose in front of the charging station.
Needs	• Define the desired pose in front of the charging station. Topological navigation will take the robot to a pose that is between $5 - 10\ m$ away from the station and heading towards the entrance.

8.1.2 Entering the station

When the robot has obtained the desired position in front of the charging station and is facing the correct way, it can start navigating into the station. As the station considered in this thesis has given dimensions, the biggest challenge for this task will be to maintain clearance with the edges of the gate. Thorvald should be able to generate and follow a trajectory that leads it safely through the gate.

To ensure safe navigation through the gate, Thorvald should be able to recognize the boundaries of the gate and generate a path based on the middle point.

Table 8-2: Goal and needs for the action of entering the charging station.

Enter the charging station	
Goal	Enable Thorvald to safely navigate through the gate of the charging station to a pose from which it can localize the charger.
Needs	• Sensors for detection of the gate boundaries. • A path planner that can generate a path based on the sensor data. • A controller that allows Thorvald to follow the path.

8.1.3 Aligning with the charger

When the robot has entered the station, it first needs to localize the charger, and then plan a path to align with it. To obtain knowledge about the orientation of the charger and where it is located, several techniques can be used, such as, for instance, a static map in which the robot can localize itself and the charger's pose is known. However, the alignment with the charger is very important, and a static map is not necessarily a robust solution. It is, therefore, considered necessary to include functionality that ensures continuous knowledge about the exact location and orientation of the charger.

It is assumed that the actual charging mechanism will perform the task of connecting with the robot. Hence, the only goal for Thorvald is to align with the charger.

Table 8-3: Goal and needs for the action of aligning with the charging mechanism.

Align with the charger	
Goal	Enable Thorvald to align with the charging mechanism so that the charger can be connected easily.
Needs	• Sensors for localization of the charger. • A path planner that can generate a path that will align the robot with the charger. • A controller that enables Thorvald to follow a trajectory.

As the task of entering the station and aligning with the charger share the same needs, the concepts that will be considered as solutions may include versatile functions that can be used for both tasks.

8.2 Concepts for detection

This section presents concepts that will enable Thorvald to localize the charging station gate and the dock itself. The aim is to use the same localization concept to localize the gate of the charging station and the charger. Multiple techniques can be used to solve the main problem, but due to the strict timeframe, only four concepts are considered.

8.2.1 2D laser scanner and reflective tape

This concept uses a 2D laser scanner to recognize two poles that mark the charging dock, and possibly the gate for safe navigation into the station. The points are made distinguishable by using reflective tape as passive beacons. The reflective tape increases the intensity of the reflection measured by the lidar, and only the measurements above a certain threshold are used to define two points for the poles. From these points, the baseline between them is calculated to generate the orthogonal from its middle point. The reference line will be used to determine the orientation of waypoints and a goal pose to ensure safe navigation and alignment. The concept is depicted by the illustration in Figure 8-4, where A is the 2D lidar, B1 and B2 represent the pillars with reflective tape that mark the dock, *m* is the middle point of the baseline between the pillars, and the line *ref* represents reference path to align with the dock. Calculations will involve trilateration and triangulation based on distance and angle information from the laser.

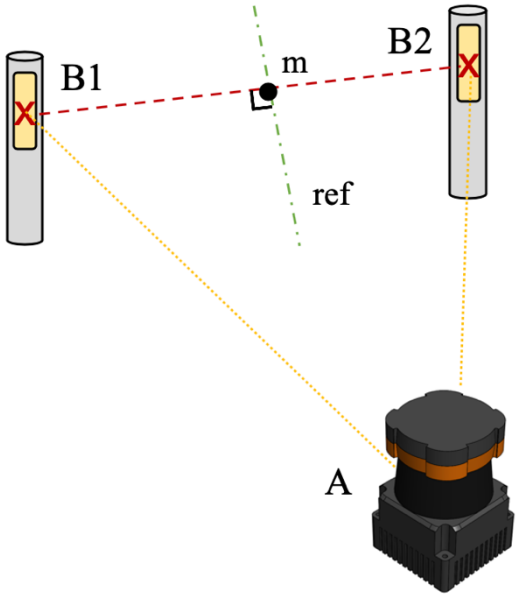


Figure 8-4: An illustration of the concept that considers the use of a 2D lidar and reflective tape beacons for recognition of the dock's boundaries. A: Hokuyo UTM 30LX-EW, B: Pillar with reflective tape that marks the docking area.

Pros

- Two Hokuyo UTM LX30-EW lasers are available and compatible Thorvald, which also means that the technology can be tested.
- Reflective tape is a rather mobile option and can be applied to various charging dock designs, and station gate sizes.
- The concept can easily be implemented.
- The Hokuyo laser provides very accurate measurements [90].

Cons

- The intensity of the relevant tape needs to be tested to ensure correct thresholds and robustness to environmental conditions.
- It is necessary to take into account the effects of outliers when detecting the pillars.
- Lasers are rather expensive and can easily be affected by, for instance, sunlight, rain, and fog.

8.2.2 RGB-D camera and visible landmarks

Similar to the previous concept, RGB-D cameras can also be used to recognize the pillars and the gates. This concept consists of using a depth camera to recognize either colored geometries to extract the position of the points of interest. Figure 8-5 illustrates the concept with yellow spheres that are recognized by an RGB-D camera. The pixels that represent the center of the circles are extracted from the image and used to calculate the position from the robot.

As the RGB-D camera can measure depth, the reference path or orientation of the charger can be calculated the same way as for the laser concept, given that the transformation between the camera frame and the body-fixed frame of the robot is known.

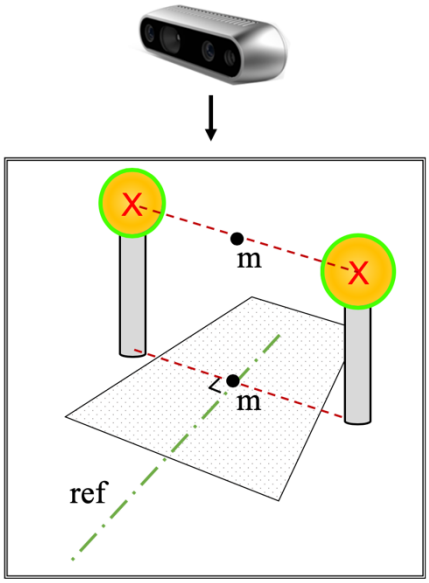


Figure 8-5: An illustration of the concept that considers the use of an RGB-D camera and colored geometries for recognition of the dock's boundaries (camera: realsense.com).

Pros

- Existing computer vision algorithms can be used to achieve the desired result.
- A RealSense RGB-D camera is available and is compatible with both ROS and Thorvald's system.
- Python has several libraries that can be used to perform the computer vision task.

Cons

- Computer vision is highly subject to light conditions.
- The camera has a limited field of view.
- The camera can easily be affected by dirt and rain.

8.2.3 Signaling beacons

This concept uses active beacons to provide information about the position of the two pillars by the dock. Depicted in Figure 8-6, are two beacons, B1 and B2, located at the pillars of the charging dock. These active beacons transmit signals to the receiver, A, that allows Thorvald to localize them and generate a reference trajectory similar to the previous concepts.

The reference trajectory intersects the middle point of the baseline, which is marked with *ref*. Waypoints and a goal pose can be determined by using trilateration and triangulation on the detected pillar locations.

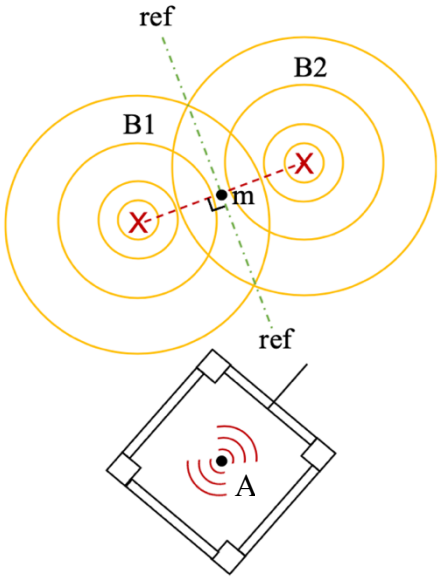


Figure 8-6: An illustration of the concept that considers the use of infrared beacons for recognition of the dock's boundaries. B1 and B2 represent beacons located at the docking boundaries. "A" represents a receiver that is mounted on Thorvald.

Pros

- Inexpensive and easy to set up.

Cons

- Noisy signals can pose challenges for accurate and precise localization.
- Bad precision increases the need for robust filtering.

8.2.4 Wire guidance

This concept involves using magnetic tape to set up the desired path for Thorvald to follow. The magnetic tape is attached to the charging station floor from the gate to the dock. A hall-effect sensor is mounted on the robot and tracks lateral errors between the robot and the path. Based on these errors, lateral motion commands are generated to guide the robot along the path. An illustration of the concept is shown by Figure 8-7.

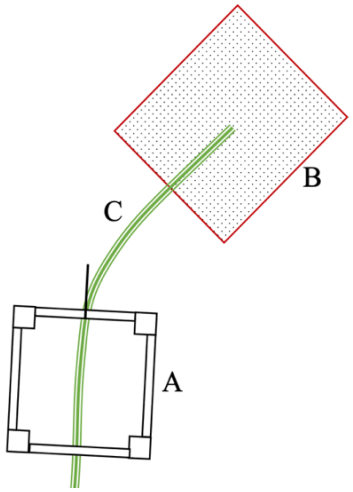


Figure 8-7: An illustration of the concept that considers the use of wire to guide Thorvald to the charging dock. A: Thorvald, B: the charging dock, C: Guidance path.

Pros

- Immune to dirt and lighting conditions.
- Inexpensive and easy to set up.
- The desired path can be chosen.

Cons

- The robot will be very dependent on the line.
- The wire is likely to be fragile.

This chapter has presented a function analysis for the docking system where the primary and secondary functions were revealed, and ways to obtain them were mapped. Also, concepts for detection and localization of the station gate and the dock were generated and described. In the next two chapters, selection processes will be conducted to select one concept for detection and an appropriate motion controller.

9 Selection of concept for detection

In this section, a weighted matrix is constructed to determine the most appropriate concept, or concepts, from the previous section. To use a weighted matrix for selection, there are three main requirements; A well-defined set of criteria, weights that describe the importance of each criterium, and a collection of alternatives to evaluate.

9.1 Selection criteria

The following weighted matrix is based on the system requirements described in Chapter 7. Not all system requirements are considered relevant for the selection of a detection concept and, therefore, not all criteria have been used. However, two concept specific criteria have been added and an overview of the criteria is provided in Table 9-1.

Table 9-1: An overview of the selection criteria that has been set for the selection of detection concept.

Key system requirement	**Selection criteria**	**Weight Coefficient**
Robustness	The concept can withstand disturbances, such as bumps, obstacles, and environmental conditions.	2
Sensor technology	The sensor technology used in the concept is accurate and precise.	4
	The sensor technology used in the concept is reliable for its operating environment.	
Functionality	The concept will enable the robot to localize and align with the charger.	5
Transferability	The concept can be used by various configurations of Thorvald.	1
Feasibility	The sensors are available.	3
	The concept can be tested within the timeframe.	
	The function can be tested without the robot.	
Installation costs	There are large installation costs related to the concept.	1
Accuracy and precision	The detection concept provides both accurate and precise pose estimates for the dock.	3

9.2 Screening matrix

Based on Pugh's selection method, the concepts have been ranked based on the criteria from Table 9-1. A score between 1 and 5, where 5 is best, has been assigned each of the concepts based on their capability of meeting the criteria. The score has been multiplied with the corresponding weight and summarized with the scores from the other criteria. Table 9-2 provides the screening of the various sensors.

Table 9-2: Screening matrix for selection of concept for detection.

Criterium	**Weight**	**Concept**			
		Lidar detection	Camera detection	Bluetooth beacons	Wire guidance
Robustness	**2**	4	4	4	3
Sensor technology	**4**	4	4	3	4
Functionality	**5**	4	4	3	3
Transferability	**1**	5	4	4	2
Feasibility	**3**	5	5	2	1
Installation costs	**1**	5	5	1	4
Accuracy and precision	**3**	3	4	2	4
	Sum:	**78**	**80**	52	58

The lidar concept and the camera concept received very similar scores in the screening. Because both of these concepts are based on the same principles of detecting two reference points to generate the path, it is considered necessary to conduct small-scale experiments to determine which of the two concepts is more appropriate. The next sections present two small-scale experiments, one for each concept.

9.3 Small-scale experiment with a camera for landmark detection

This section describes a small-scale experiment that has been conducted for the computer vision concept. A brief explanation of the experiment is provided alongside important takeaways.

9.3.1 Camera test goal

The main goal for the experiment was to see if an RGB camera and a computer vision algorithm can be used to extract two points that can be used to localize the charging dock.

9.3.2 Steps

Distinguishable objects were placed to mimic the pillars of the dock. A video was then recorded of the objects from different angles and in different environments, on which a computer vision algorithm was used to determine if the objects can be recognized.

9.3.3 Experimental set-up

Hardware

To mimic the two pillars of the dock, interior lamps and orange balloons were used. The balloons were orange so that they could easily be distinguished from the surrounding environment. Videos are captured with a personal cellphone (Google Pixel 3 [91]).

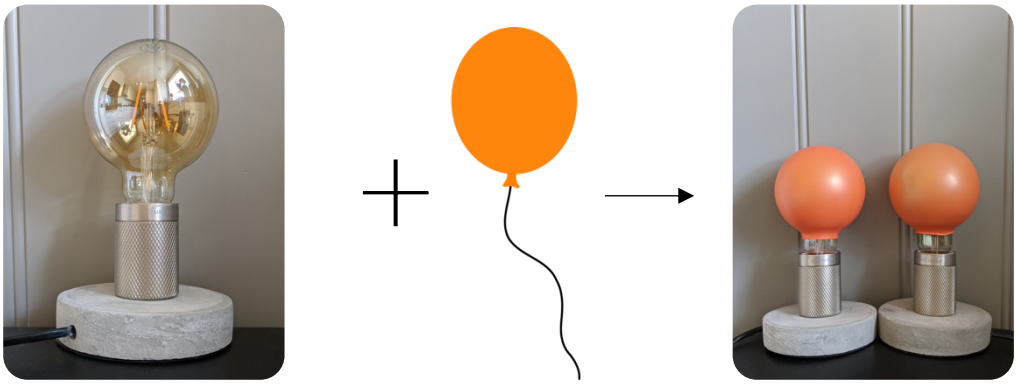


Figure 9-1: An overview of the experiment material used to conduct a small-scale experiment of the camera concept.

Software

The software that was used was developed based on packages included in OpenCV. For the detection of the circles, pre-processing was done with masks to extract orange colors in the image. The masked photo was then processed with an algorithm named HoughCircles, which uses Canny edge detectors and Hough transforms to detect circles.

9.3.4 Results from small-scale experiment of computer vision

Experiments ware conducted with various tuning for the algorithm parameters. These parameters include edge detection sensitivity, image brightness, and blur, and internal parameters for the HoughCircles algorithm. Figure 9-2 shows results from the tests, where green circles mark the edges of detected geometries in the images. Blue diamond geometries mark the center of the circle from which a red line is drawn to connect two detections. The middle point of the red line is marked with a cyan diamond and represents a potential way to determine the position of the charger dock. In Figure 9-2, C and E mark situations where the algorithm fails to detect the geometries.

Key takeaways from the camera experiment

- Computer vision can be used to detect points of interest.
- Simple algorithms can be used, but tuning is necessary to reduce errors.
- Cameras are, as expected, very dependent on the lighting conditions.
- Color segmentation and edge detection are good alternatives for detecting simple geometries.

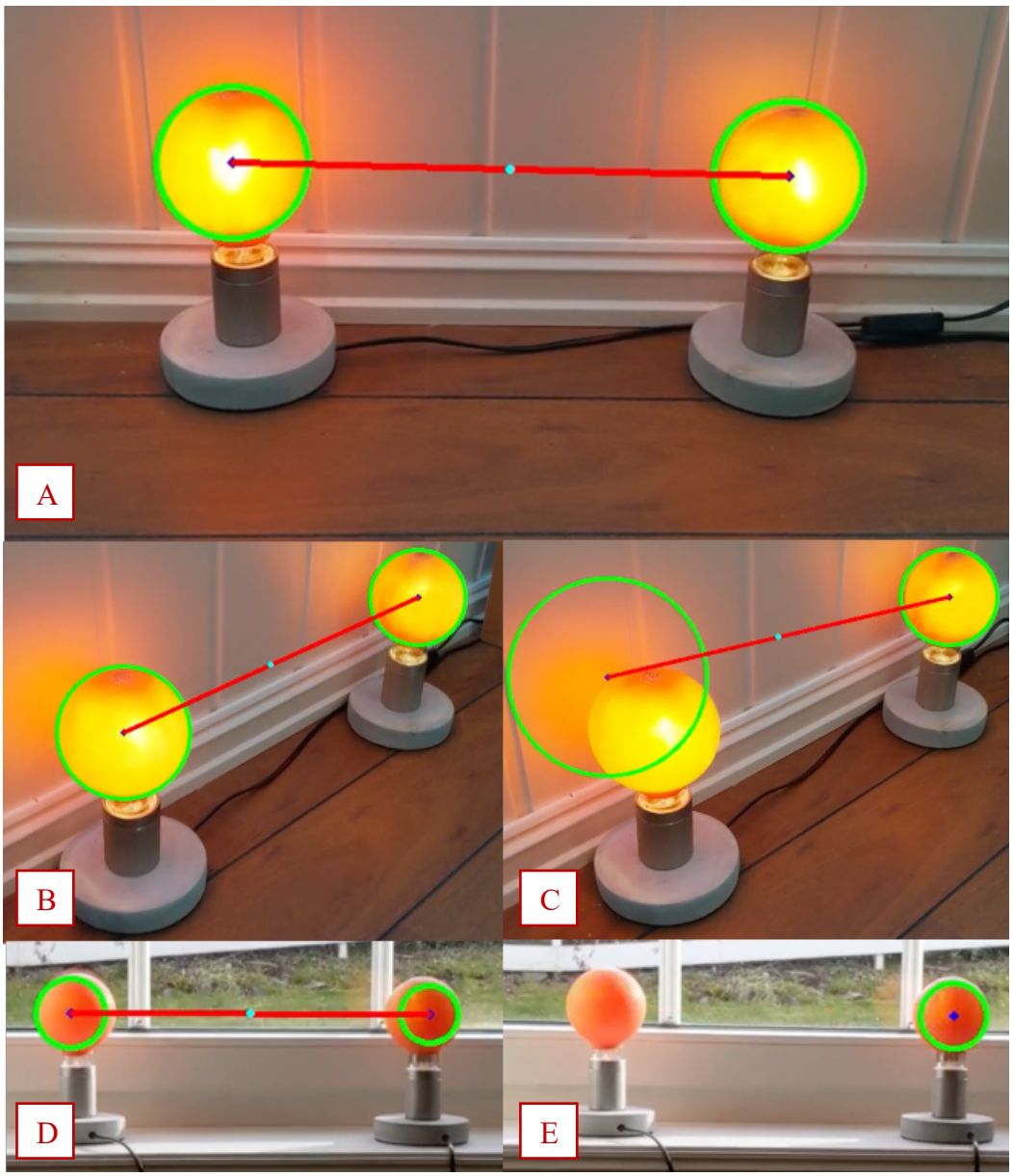


Figure 9-2: Results from small-scale experiments for using computer vision to detect the charging dock. A: Pillars located in front of the camera. B: Pillars seen from the side. C: Error in detection with the pillars seen from the side. D: Pillars located by a window. C: Error in detection with the pillars located by a window.

9.3.5 Discussion of the results from the camera test

The errors can be reduced by tuning the parameters of the algorithm for the desired distances, lighting conditions, and color ranges. However, as these properties are dynamic, tuning will have to be done continuously. Although this small-scale experiment gave a proof-of-concept, the used algorithm will not be sufficiently robust for the docking system, and other alternatives must be considered. However, both OpenCV and ROS provide more robust algorithms that may provide satisfactory functionality.

9.4 Small-scale testing of lidar and reflective objects

This section presents a small-scale experiment that has been conducted with one of Thorvald's 2D laser scanners.

9.4.1 Laser scanner test goal

The main goal for this experiment was to see if intensity measurements from a laser scanner can be used to distinguish landmarks in an environment and if two points can be extracted to localize the charging dock.

9.4.2 Steps

Reflective objects were placed at a given distance away from a laser scanner. A program was made to save laser measurements in datasets, while RViz was used to monitor the readings. The objects were scanned two times each, at two different distances; 180 cm and 60 cm. An extra test was conducted on the object that provided the highest intensity measurements, where two samples of the object were placed to mimic the pillars of the dock. The collected data was analyzed in MATLAB to determine if the intensity measurements were sufficient to recognize the dock.

9.4.3 Experimental set-up

Hardware

Hokuyo UTM-30LX can record information about the intensity of a reflection, where a greater amount of reflected laser beams provides a higher intensity measurement. The unit of the intensity measurement is device-specific and was, hence, only evaluated based on the measurements from the different objects.

Figure 9-3: An overview of the objects used for a small-scale experiment with the Hokuyo scanner. From left: Water glass, reflective band, bottle covered with aluminum foil, bear-shaped reflective figure, black thermos, mirror.

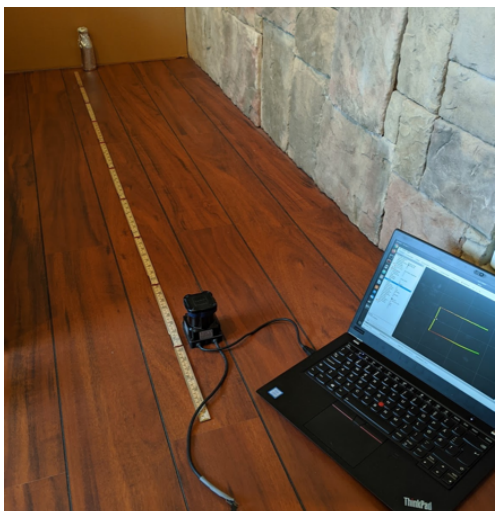


Figure 9-4: Experimental setup for laser intensity measurements from a bottle covered with aluminum foil.

Figure 9-3 shows the objects that were used in the laser experiment. Figure 9-4 shows the experimental set-up for an experiment where one of the objects was located 180 cm away from the laser. An additional experiment was conducted where the object was placed at a distance of 60cm.

Software

To analyze the intensity measurements, a bagfile was recorded and converted to tabular data for analysis in MATLAB. RViz was used for monitoring of the intensity measurements during the experiments.

9.4.4 Results

A chart is provided in Figure 9-5 as an overview of the intensity measurements obtained through two short scan tests for each of the objects in Figure 9-3. In the first test, the objects were placed 60cm away from the laser, whereas, in the second test, they were placed at 180cm.

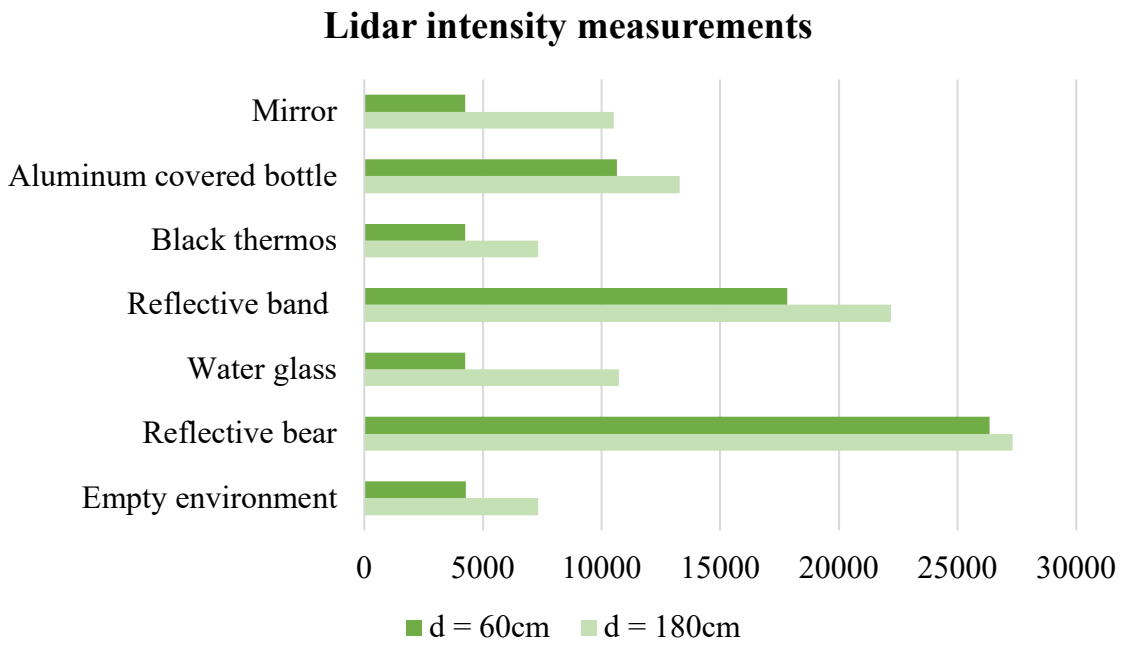


Figure 9-5: A chart showing laser scan intensity measurements from various objects placed at two different distances, 60 and 180cm

Figure 9-6 on the next page illustrates an intensity plot from a scan where the bear-shaped reflectors were placed at a distance of 80cm away from the laser, and with 40cm spacing. Measurements that show intensities above 25000 are marked with red dots.

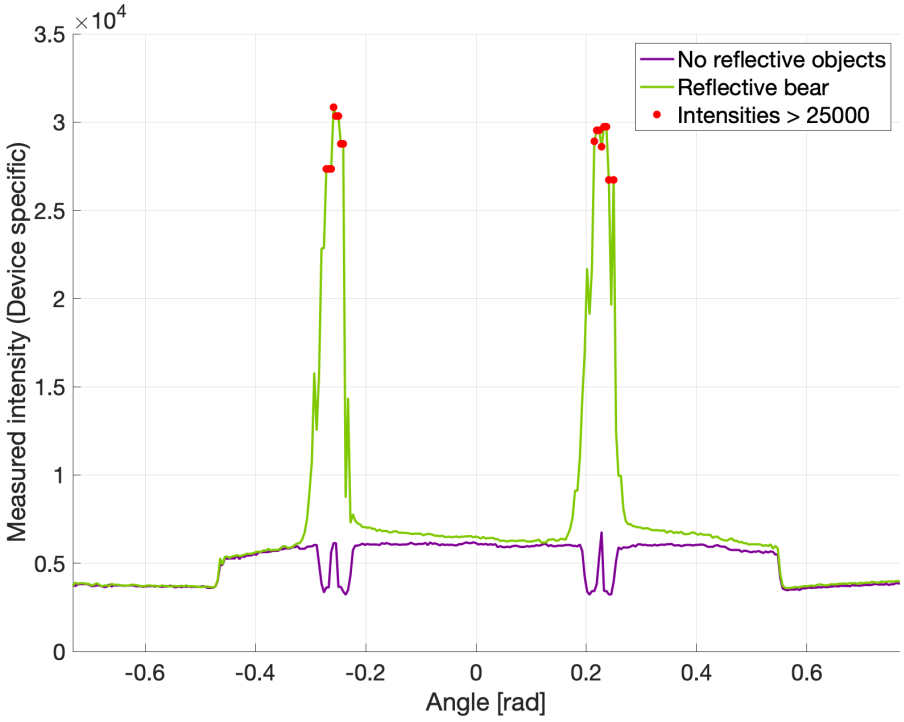


Figure 9-6: A plot that shows a comparison of two intensity measurements, one with reflective objects and one without.

Key takeaways from the laser experiment

- Reflective objects with prism structure provide significantly higher intensity readings compared to other objects tested. These results indicate that reflective objects with prisms are good alternatives for making distinguishable points for the lidar.
- For the distances considered in the tests, lighting conditions do not seem to have significant effects on the intensity measurements.
- The laser is rather simple to set up and the data is easy to handle.

9.4.5 Discussion of the results from the laser concept test

The results from the laser test show that reflective objects provide greater intensity readings and that this data can be used for landmark detection. However, possible false positives and errors may occur due to various conditions and reduce the reliability of the concept. Therefore, further considerations of the robustness, and the technology in general, need to be done before the concept gets deployed on the robot.

9.5 External comments on the two concepts

To get professional feedback on the two detection concepts, an external survey was constructed. The survey was sent to 20 people experienced either with relevant sensor technologies or with mobile robots in general. This section presents the main feedback from the 11 responses that were recorded. A complete copy of the survey is presented in Appendix IV.

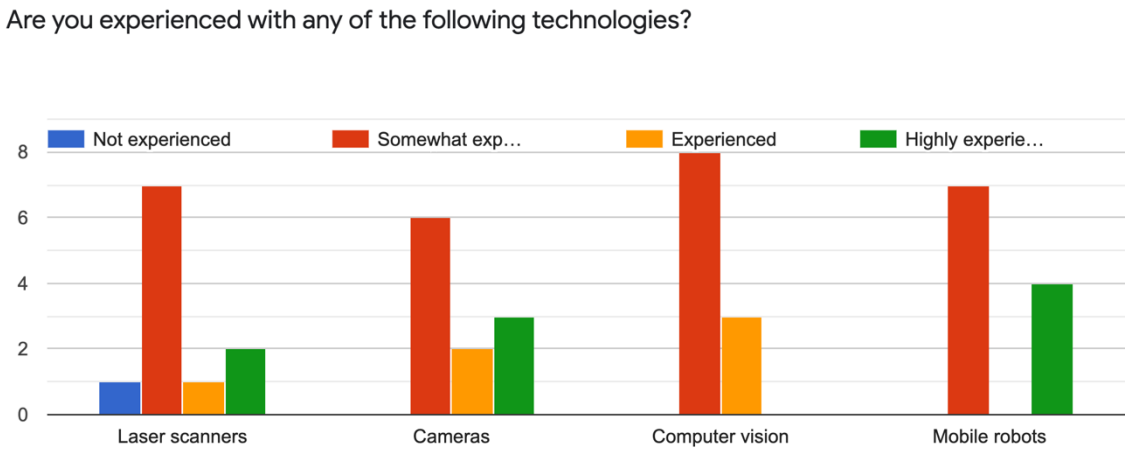


Figure 9-7: A chart showing answers recorded on experience with sensor technologies.

Figure 9-7 shows the distribution of the participants' experience. A follow-up question was also asked for the participants to indicate how many years of experience they have with their most familiar technology. Seven out of a total of eleven participants stated they have more than 2 years of experience, while the remaining four had at least one year.

The pie chart in Figure 9-8 shows the response on a question about the suitability of the laser concept along with helpful feedback that was provided through an optional written field.

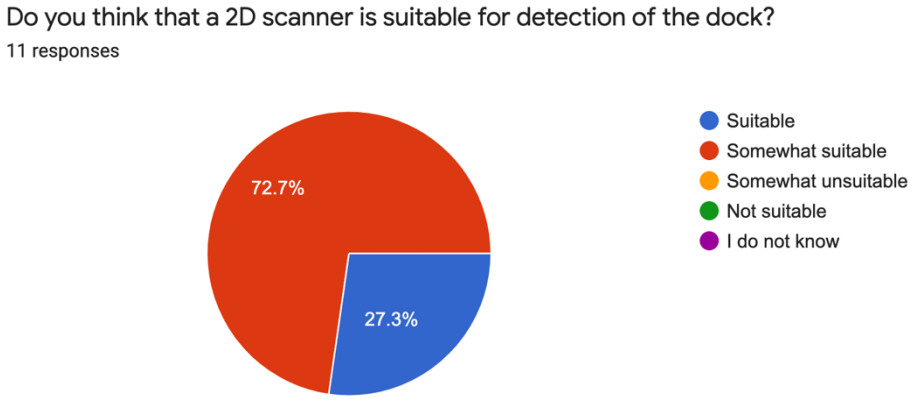


Figure 9-8: Answers recorded about the suitability of a laser scanner.

Participant, highly experienced with laser scanners:	"*… there are some concerns, for example false positive detections on materials that can reflect the laser as strong as the reflective object …*"
Participant, highly experienced with laser scanners:	"*I think your main limitation is the docking design for using laser only, also reflective landmarks are usually not complex enough to id them individually and lose precision over distance due to the radial nature of the sensor*"

Figure 9-9 provides a pie chart that shows the response to a question about the suitability of the camera concept. Below the figure is useful feedback received through an optional field for written responses.

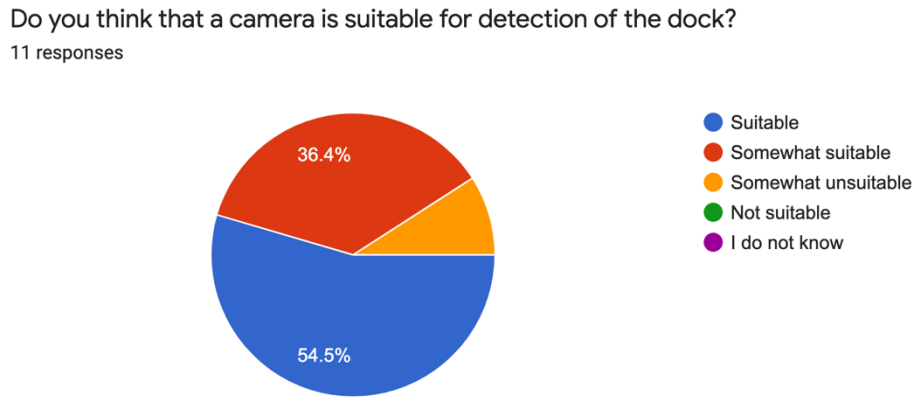


Figure 9-9: Answers recorded about the suitability of a camera.

Participant, highly experienced with cameras:	"*In your pictures you show lighted features, I would argue against that, there are plenty of well tested fiducial markers available such as whycon and AprilTags that will give you a lot more precision.* "
Participant, experienced with cameras:	"*Only using AprilTags will require you to be quite close to the tag to find the ground truth position, approximately 2-3 meters I think, but it depends on the camera and lighting conditions. An advantage to the AprilTags is that, when they are first detected, it is highly accurate and with few false positives*"

Finally, the participants were asked to answer whether they would choose a laser scanner or if they would a camera to perform the task. Responses are shown in Figure 9-10.

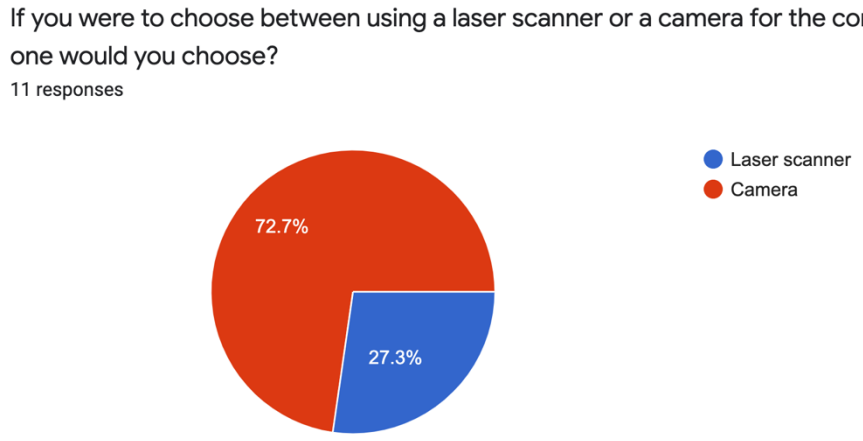


Figure 9-10: Answers recorded on the question where participants were asked to choose between a laser scanner or a camera.

As depicted in Figure 9-10, the majority of the participants said that they would prefer a camera rather than a laser scanner to perform the detection. Feedback also suggested using *AprilTags*, as fiducial detectors tend to provide more robust detections. The small-scale experiment in section 9.3 indicated that a more robust solution is necessary to obtain sufficient measurements. Because of this, a description of an experiment that has been conducted with an RGB-D camera and an AprilTag detector will now be presented.

9.6 Testing of externally recommended detection algorithm

This section presents a small-scale experiment on a fiducial detection algorithm named AprilTag [92]. The experiment was conducted as a result of the professional feedback from the survey presented in the previous section.

9.6.1 ROS AprilTag

AprilTag is a fiducial detector that can be used for multiple tasks, such as computer vision, camera calibration, or AR, which is provided by the ROS package named *apriltag_ros*. The algorithm uses distinguishable artificial landmarks called AprilTags, which are very similar to QR-codes in that they are 2D barcodes that can be programmed to contain desirable information. In contrast to other barcode systems, AprilTags contain minimal information and are, hence, easier to detect.

Existing libraries with pre-generated tag families and calibration algorithms can easily be used through the ROS package, and allow for precise determination of the 3D positions, orientations and identities of the landmarks. The AprilTag algorithm is robust and can detect the tags also under challenging conditions [93].

An example of a tag from a commonly used tag family, *36h11*, is shown in Figure 9-11. The name *36h11* comes from its 36-bit encoding and minimum Hamming distance, h11 [94].

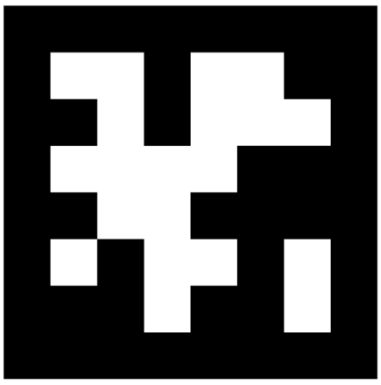


Figure 9-11: AprilTag from the tag family 36h11 with ID: 1.

9.6.2 Experiment goal

The goal for this experiment was to determine whether the AprilTag detector provided by ROS is appropriate for the detection of the charging station and the dock.

9.6.3 Steps

AprilTags were used to mimic the pillars of the charging station, and an RGB-D camera was used with AprilTag algorithm to determine if tags could be detected under various light conditions and from various distances. The results were also used to determine if the data could be used to calculate the position of the dock. The experiment was, hence, conducted outside, under sunny conditions, so that the algorithm could be tested in direct sunlight.

9.6.4 Experimental set-up

Hardware

AprilTags with ID 1 and 2 from the 36h11 family were printed on white paper and fixed to cardboard so that they could be mounted more rigidly. The tags were mounted approximately two meters apart to resemble characteristics of the charging dock. An Intel RealSense D415 depth camera [95] was used to gather image data from various distances. Figure 9-12 below provides an image of the experimental setup.

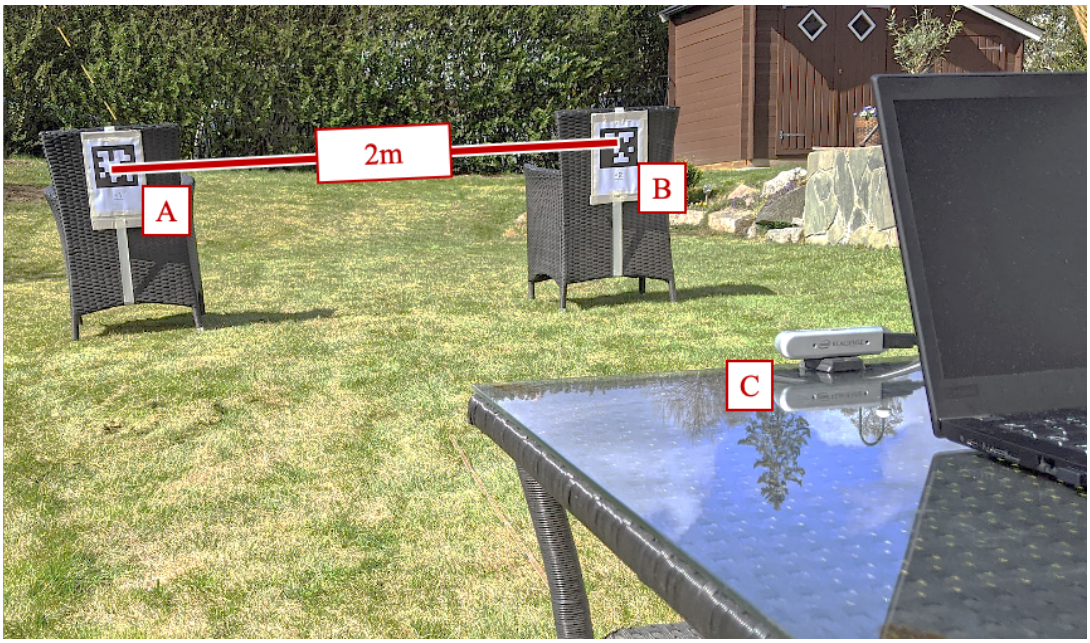


Figure 9-12: Experimental set-up for one of the scenarios for the AprilTag testing. A: Tag with ID:1, B: Tag with ID:2, C: Intel RealSense D415

Software

To use the camera with ROS, Intel RealSense has made a ROS package named *realsense2_camera*, which allows for simple control of the D415 RGB-D camera. Files from this package were used to run the camera, whereas to detect the AprilTags, the AprilTag detector for continuous detection from the package *apriltag_ros* was used. This detector enables AprilTag detection in video streams, which allows for detection also when moving.

RViz was used to monitor the experiment and to evaluate the performance of the algorithm. A bagfile was recorded so that data can be analyzed over again in case the experiment needs to be revisited.

The next section presents important results from the experiment.

9.6.5 Results

Results from the small-scale experiments are shown in the pictures below, where Figure 9-13 shows a typical tag detection image and the corresponding tag detection frames in RViz. Figure 9-14 shows a tag detection image where the estimate for the orientation of

the tag with ID 2 has errors. In this figure, the error is highlighted with a red circle. The last figure, Figure 9-15, depicts how the goal pose can be calculated from the data given by the tag detections. The leftmost illustration in this figure shows a broadcasted transformation that is constructed based on the obtained data. In all of the figures, the rightmost image shows the tag detection image, where the tags are highlighted and marked with numbers and the leftmost illustration shows the tag frames in RViz.

Figure 9-13: An illustration of an accurate and precise result from the test where, in the left illustration, the lower frame depicts the camera frame, and the two upper frames depict the frames of the detected tags. The detector successfully managed to estimate the pose of the two tags.

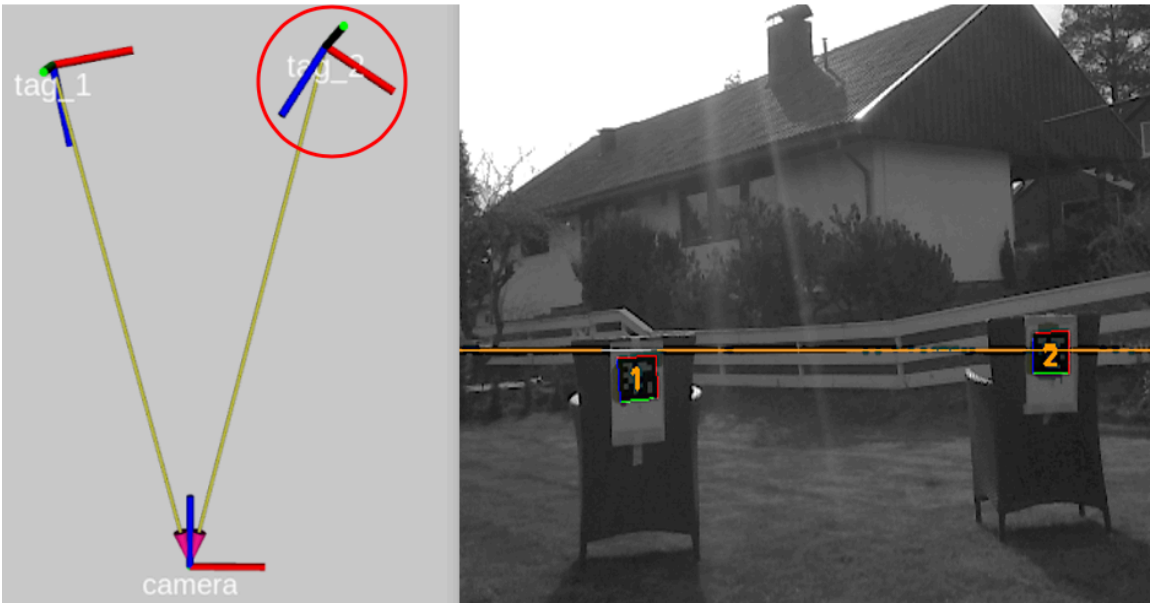


Figure 9-14: An illustration of a detection where the camera was directly exposed to sunlight and the pose estimation of the tag with ID: 2 had orientation errors. The red ring depicts the faulty detection for which the orientation of the tag is incorrect.

Figure 9-15: An image showing test results from one of the tests, where the AprilTags are detected and marked with their respective ID (1 and 2). To the left is a figure from RViz showing a transformation that was broadcasted to visualize how the charger pose can be determined from the AprilTag detection data.

9.6.6 Discussion of results from AprilTag experiment

The results from the small-scale experiment with the AprilTag detector indicate that the detection algorithm is robust and that it can provide sufficient location data both for the dock and for the charging station. The performance of the algorithm does, however, drop with increased range, and thus not considered appropriate for long-range detection.

Detection with the RealSense D415 and the AprilTag detector also seems to be affected by sunlight to a certain extent, in that some detections come with errors in the estimation of the tag pose. A pose estimate with orientation errors is detected by a red circle in Figure 9-14.

9.7 Final selection of detection concept

Based on the results from small-scale experiments, and feedback obtained through the expert survey, the following strategy is determined to be most appropriate for detection:

1) For localization of the charging dock from greater distances and for situations in which the desired detection lies outside a front-faced camera's field of view, a 2D lidar and the concept from section 8.2.1 will be used.
2) For the precise alignment and navigation from an initial pose in front of the charging station gate or the dock, the computer vision concept described in 8.2.2 will be used.

The next chapter describes the process of selecting a motion controller for Thorvald and the docking procedure.

10 Selection of motion controller

Two control designs have been considered appropriate for the docking system, a pure pursuit controller, and a pose regulator. Pugh's selection method will now be used to determine which controller is more optimal. Below is a table providing a list of criteria along with corresponding descriptions and weight coefficients.

Table 10-1: An overview of the criteria for selection of motion controller for the docking system.

Criterium	**Description**	**Weight coefficient**
Transferability	The controller does not rely on unique characteristics for the considered Thorvald configuration, and can easily be used on other configurations.	1
Integration	The controller can easily be integrated in Thorvald's system.	2
Feasibility	The controller is easy to implement, and the effort required is feasible within the project timeframe.	3

Based on the criteria in Table 2-1, the two controllers will receive a score between 1 and 3 based on how well they meet each criterium. For this scale, 3 is the best, and 1 is the worst. Similar to the process used to select the detection concept, scores will be multiplied with their corresponding weight coefficient and summarized to a final score. The controller that receives the highest total score will be used for the docking system.

Table 10-2: Screening matrix for selection of control strategy.

Criterium	**Weight**	**Controller**	
		Pure pursuit controller	Pose regulator
Transferability	**1**	1	3
Integration	**2**	2	3
Implementation	**3**	2	3
	Sum:	**12**	**18**

The pose regulator received the highest score and will hence be used to control the motion of the robot during the docking operation. In the next chapter, a complete solution that includes the system components that were selected in this chapter will be presented.

11 Proposed Solution

The previous chapters presented the selection of detection technology and control strategy that will be used for the docking system. This chapter presents a complete solution to the problem of making an autonomous docking system for Thorvald.

11.1 Function outlines

As a result of careful evaluation of various sensor technologies based on important criteria, the design of the autonomous docking system will allow for the use of either a laser scanner, a camera, or a combination of both. The laser system will use intensity measurements enhanced by reflective markers to distinguish two points. The camera system will use a depth camera along with an AprilTag detector to identify and localize the charger and the station gate.

The distances and angles to detected points will be extracted from the sensor data and provide enough information to determine a goal pose. If both sensors are available, the laser system will be used for rough localization and to generate a waypoint, whereas the camera system will be used to determine the goal pose. An algorithm for the generation of desired poses has been developed and is presented later in this chapter. Figure 11-1 illustrates how the various components of the system will be used to perform tasks for the secondary functions from the function analysis in Chapter 8.

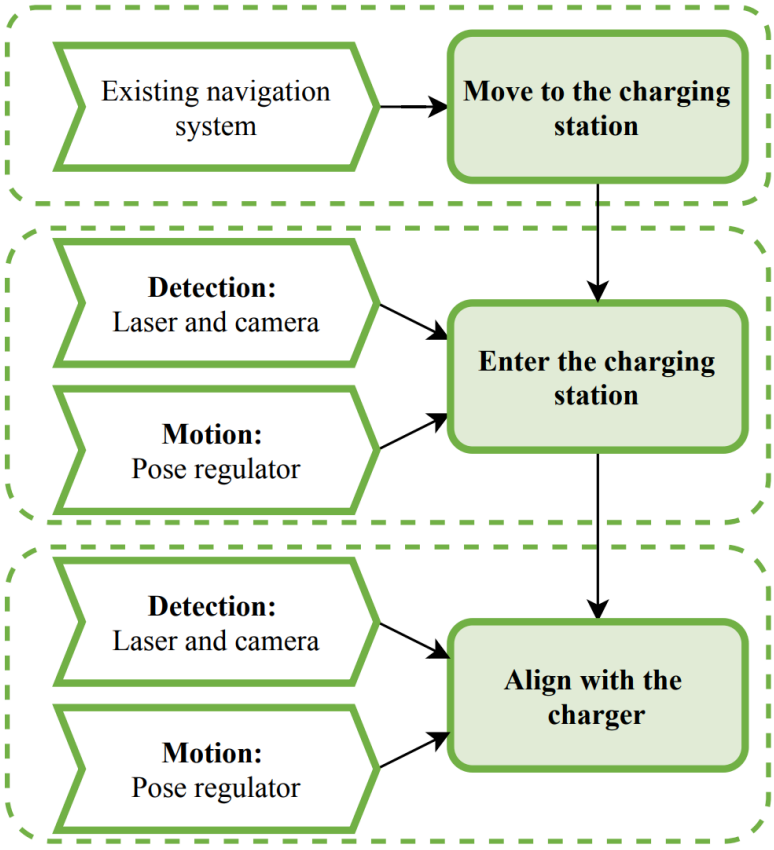


Figure 11-1: An illustration of how the secondary functions that were revealed in Chapter 8 will be covered.

11.2 Sensors

This section provides information about the proposed sensors for the docking system.

11.2.1 Laser scanner

The laser scanner that is proposed for the detection concept is the Hokuyo UTM-30LX-EW 2D laser scanner. The laser is chosen because it is already available, and because it has proven to provide sufficient data when detecting reflective objects based on intensity measurements through small-scale experiments.

Figure 11-2: An illustration of the Hokuyo UTM-30LX-EW laser rangefinder.

11.2.2 Laser scanner specifications

Table 11-1 on the next page shows the specifications for the Hokuyo UTM-30LX-EW laser given by its manufacturer that are relevant for the docking system. Scans from more than one laser can be merged and provide a greater scan angle [90].

Table 11-1: An overview of relevant specifications for the Hokuyo laser scanner.

Hokuyo UTM-30LX-EW Laser Scanner	
Scan angle (Horizontal)	270°
Angular resolution	0.25°
Range	Operating range: $0.1 - 30m$
Accuracy	At range 0.1 – 10m: $\pm 30mm$ At range 10 – 30m: $\pm 50mm$
Detection	Min. detectable width at 10m: $130mm$

A range experiment has been conducted for the laser to determine the minimum and maximum range that can be used for detection with intensity data. A description of the test, alongside results, is located in Appendix I. The range experiment was conducted outdoors with the laser being directly exposed to sunlight to mimic a worst-case scenario.

Table 11-2: An overview of detection specifications for the Hokuyo laser.

Specification	**Minimum**	**Maximum**
Detection of reflectors	$0.1\ m$	$8\ m$
Proposed intensity threshold	13.000	
Angular resolution	270°	

11.2.3 RGB-D camera

The Intel RealSense D415 depth camera has been proposed as a camera for the function. This camera is already available and has proven satisfactory performance with the AprilTag detector. However, other cameras may also be used as long as they provide adequate data for the AprilTag detector.

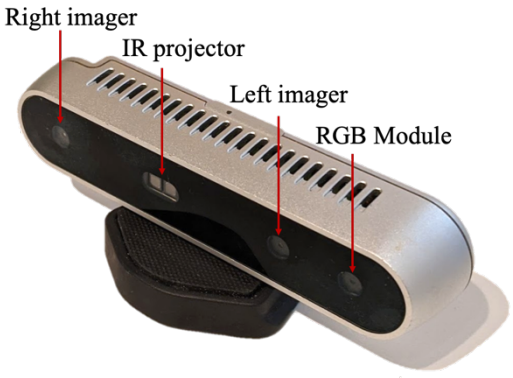


Figure 11-3: An image of the components in the Intel RealSense D415.

11.2.4 RGB-D camera specifications

Table 11-3 provides an overview of relevant specifications for the Intel RealSense D415 camera given by its producer [95].

Table 11-3: An overview of relevant specifications for the RealSense D415 RGB-D camera.

Intel RealSense D415 Depth Camera	
Depth FOV	$65° \pm 2° \times 40° \pm 1° \times 72° \pm 2°$
Depth Resolution	1280×720
Depth fps	90 fps
Depth range	$\sim 0.16 - 10\ m$

A simple range test has also been conducted for the depth camera to determine the minimum and maximum range for AprilTag detection. Quadratic tags with height and width set to 160mm were used for the experiment, and a complete overview of the results is provided in Appendix I. Based on the test, the following specifications have been set for AprilTag detection with the Intel RealSense D415 camera:

Table 11-4: An overview of range specifications for AprilTag detection with the RGB-D camera.

Specification	**Minimum**	**Maximum**
Detection of tags that are maximum 2.4 meters apart	$2.5\ m$	$4\ m$
Ranging based on detection of one tag	$0.5\ m$	$5\ m$

11.2.5 Sensor operation

As two sensor technologies will be used to complete the docking, Figure 11-4 is made to depict the areas for which the camera and laser each are expected to operate. The distance d is given by the maximum range values in Table 11-4, whereas μ is given by the camera specifications for the field of view in Table 11-3. The rotating laser's operational area is set by its field of view of 270° and a radius r equal to the maximum range in Table 11-2.

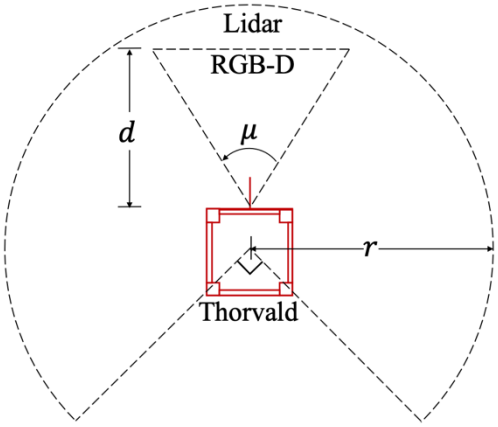


Figure 11-4: An illustration of the operating areas for the two sensor technologies.

11.2.6 Sensor detection prerequisites

The proposed detection system has a few requirements for the physical properties of the dock and the station. A mock-up design of a dock has been made to describe what is necessary to include for the docking system to work. The design is shown in Figure 11-5, where the center of the two poles defines the desired position for the robot's geometric center when successfully docked. In the design, the poles are depicted with reflective metal. In reality, these poles will need to be covered with reflective tape to provide satisfactory reflection. The pictured design is not necessarily how the dock will look, as the physical design goes beyond the scope of this thesis.

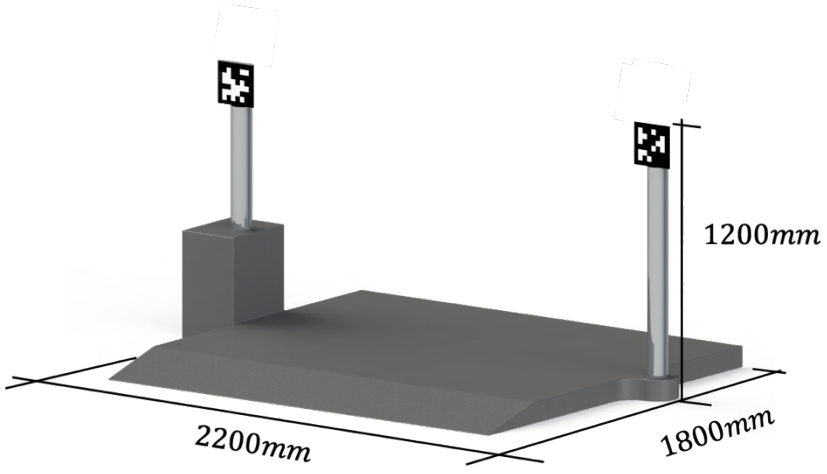


Figure 11-5: An image of a possible design of a charging dock which is made in SolidWorks for illustrative purposes. AprilTags with ID 1 and 2 from tag family 36h11 has been mounted on each of the poles.

11.2.6.1 Reflective markers

To distinguish objects of interest with the laser, it is proposed to cover the surface of the objects with reflective markers. Small-scale experiments have indicated that reflective markers with prism structure provide the best intensity measurements, thus making them more appropriate for the laser detection concept. Examples of reflective markers that are commonly used for laser surveying are shown in Figure 11-6.

Figure 11-6: An image of reflective markers used for surveying. (Leica Geosystems)

11.2.6.2 AprilTags

It is proposed to use two pairs of AprilTag IDs to mark both the charging dock and the entrance of the charging station. By doing so, the task of entering the docking station can easily be distinguished from the task of the actual docking. Furthermore, as a supplement to the function when only using cameras, a fifth tag ID can be used for range determination. A range may become necessary to ensure correct positioning during the docking. The following set of AprilTags from tag family 36h11 should be used for the functions, but IDs may be substituted if desired.

Tag ID: 0	This tag will be used to measure specific distances when a laser is unavailable.	
Tag ID: 1	This tag will be used to mark the left side of the charging dock.	
Tag ID: 2	This tag will be used to mark the right side of the charging dock.	
Tag ID: 3	This tag will be used to mark the left side of the charging station gate.	
Tag ID: 4	This tag will be used to mark the right side of the charging station gate.	

11.3 Waypoint and goal pose calculations

To plan a path to the goal, it is proposed to use a roadmap technique. The proposed technique is depicted in Figure 11-7 and is based on principles from trilateration and triangulation. The transformation between the base frame and the sensor frame is assumed available through ROS tf, which allows for the simplified illustration that only is based on the robot base frame constructed by x_b and y_b. The points can be extracted either from laser intensity measurements or AprilTag detections.

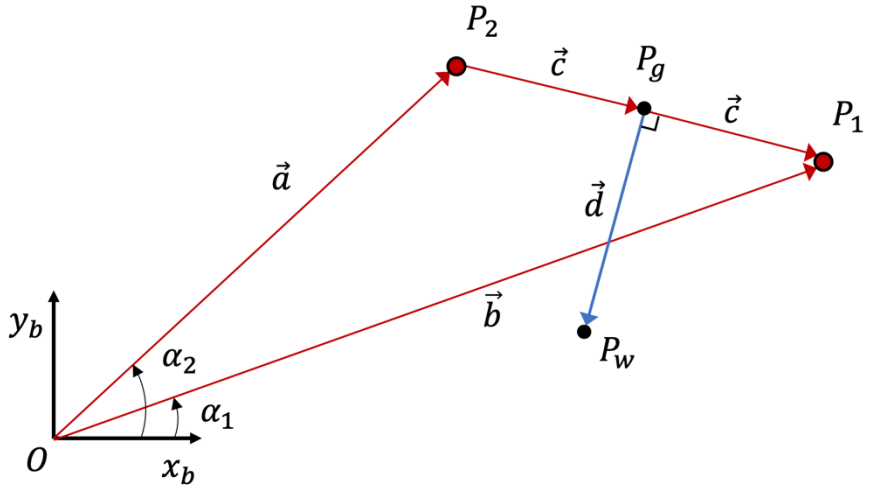


Figure 11-7: An illustration of the vector representation of a detection problem in an arbitrary situation.

The two points extracted from the sensor data are marked P_1 and P_2, with coordinates given with respect to the base frame. The points are detected at angles α_1 and α_2 from x_b, and their distances from the base link is the length of their vectors denoted by $\vec{a} = \overrightarrow{OP_2}$ and $\vec{b} = \overrightarrow{OP_1}$. Their coordinates are given by their x- and y-components in the base frame.

In every situation, P_1 will be defined as the point with the smallest angle with respect to the x-axis, $\alpha_1 < \alpha_2$, whereas P_2 will be the point with the greatest angle. P_g will denote the virtual point which defines the location of the goal, and P_w will define a virtual waypoint generated at a desired distance, $|\vec{d}|$, away from the goal. To ensure alignment with the charger, the waypoint will be defined on the normal of the baseline between P_1 and P_2. An orientation for each of the points is also necessary as the goal for this task is to determine a waypoint pose and a goal pose. The desired orientation at point P_w and P_g will, however, be determined from the opposite vector to vector $\vec{d}$. The distance to the detected points, $|\vec{a}|$ and $|\vec{b}|$, and the angle at which they are detected, α_1 and α_2, can easily be extracted

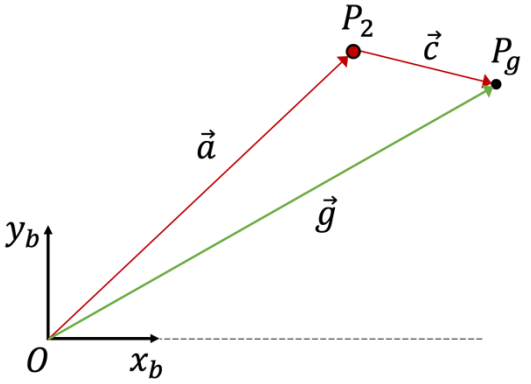


Figure 11-8: Illustration of the goal position vector.

from the sensor data. As it is assumed that the pillars are placed so that the middle point of their baseline defines the goal position, the position of P_g can be determined by vector $\vec{g}$ in Figure 11-8 as follows:

$$\begin{aligned}\overrightarrow{OP_g} &= \vec{g} \\ &= \overrightarrow{OP_2} + \overrightarrow{P_2P_g} \\ &= \vec{a} + \vec{c} \\ &= \vec{a} + \frac{\vec{b} - \vec{a}}{2}\end{aligned} \tag{10.1}$$

$$\Rightarrow \quad \vec{g} = \frac{1}{2} \cdot \begin{bmatrix} a_x + b_x \\ a_y + b_y \end{bmatrix} \tag{10.2}$$

where subscripts x and y denote component representation in the base frame.

To determine the coordinates of a waypoint a desired distance away from the goal, and that has the same orientation in the base frame as the baseline normal, basic vector properties will be used to condition the generation of vector $\vec{d}$ in Figure 11-7 and Figure 11-9. In Figure 11-9, $\vec{d}$ defines the vector from the desired goal point to the desired waypoint, which can also be expressed as $\overrightarrow{P_gP_w}$. This vector is desired to be perpendicular to the baseline normal between P_1 and P_2, thus $\vec{d}$ needs to be defined so that the dot product:

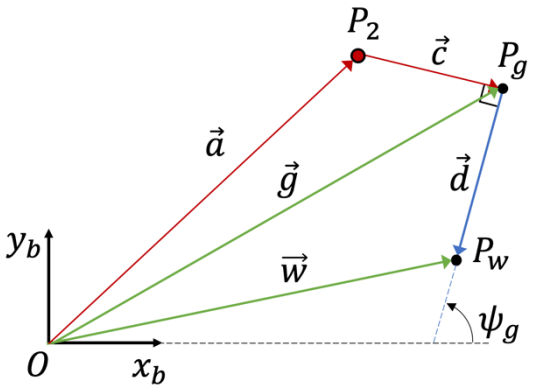


Figure 11-9: An illustration of the set of vectors needed to determine the coordinates of a waypoint with the desired properties.

$$\vec{d} \cdot \vec{c} = 0 \tag{10.3}$$

This property is given by the cosine rule (Formula 3.7), and when this condition holds for the dot product, the vector components d_x and d_y can be calculated as follows:

$$\begin{aligned}\vec{d} \cdot \vec{c} &= 0 \\ d_x c_x + d_y c_y &= 0 \\ \Rightarrow \quad d_x &= -\frac{d_y c_y}{c_x}\end{aligned} \tag{10.4}$$

The Euclidean distance can then be used to calculate d_y:

$$d_x^2 + d_y^2 = d^2 \tag{10.5}$$

$$\Rightarrow \quad d_y = \pm \sqrt{\frac{c_x^2 d^2}{c_x^2 + c_y^2}} \tag{10.6}$$

The expression for d_y provides two possible solutions for the vector component pair which will make $\vec{d}$ either point in the direction which is illustrated in Figure 11-9, or the opposite. However, it is reasonable to assume that vector $\vec{d}$ is always directed in the opposite direction of vector $\vec{a}$, towards the base of the robot. This assumption is reasonable because the charger, most likely, will be placed so that the robot does not move behind it. With this assumption made, a final condition can be set for $\vec{d}$ that allows for only one solution of the vector. Based on the unit circle, shown in Figure 11-10, any vector in the 2nd and 3rd quadrant will have a negative value for cosine. Any angle, λ, between $\vec{a}$ and $\vec{d}$ bigger than $\frac{\pi}{2}$ will, hence, result in a negative dot product and imply that the vector is directed towards Thorvald.

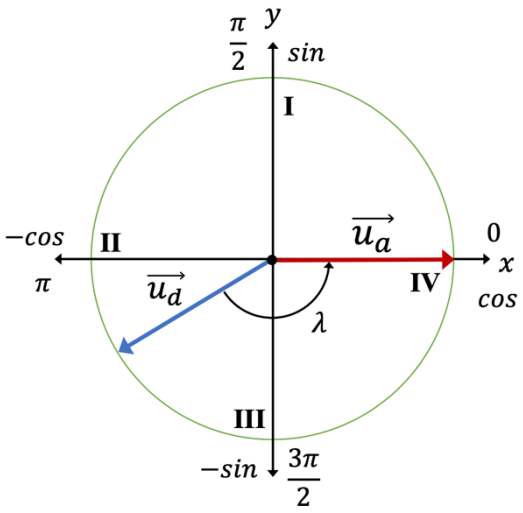


Figure 11-10: An illustration of the unit circle with the unit vectors of $\vec{d}$ and $\vec{a}$ to illustrate the direction condition.

$$\vec{d} \cdot \vec{a} < 0$$
$$d_x a_x + d_y a_y < 0 \tag{10.7}$$

By replacing d_x by the expression in Equation (10.4) we get,

$$-\frac{d_y c_y}{c_x} a_x + d_y a_y < 0 \tag{10.8}$$

from which the value of d_y that validates the inequality will be the desired value of d_y to define $\vec{d}$. With vector $\vec{d}$ fully defined, $\vec{w}$, the vector from the origin to the waypoint, P_w, can be defined as follows:

$$\vec{w} = \vec{g} + \vec{d} \tag{10.9}$$
$$= \vec{a} + \vec{c} + \vec{d}$$
$$\Rightarrow \quad \vec{w} = \begin{bmatrix} a_x + c_x + d_x \\ a_y + c_y + d_y \end{bmatrix} \tag{10.10}$$

To determine the orientation of the goal and the waypoint, ψ_g, the opposite vector to $\vec{d}$ will be used. This is because the orientation of this vector is the same as the desired heading for Thorvald at the locations. As $\vec{d}$ is fully defined, the angle can easily be obtained through the following relationship, where atan2 is used for correct consideration of quadrants:

$$\tan(\psi_g) = \frac{-d_y}{-d_x} \tag{10.11}$$
$$\Rightarrow \quad \psi_g = atan2(-d_y, -d_x) \tag{10.12}$$

Finally, the poses for the waypoint and the goal with respect to the base can be defined as:

$$X_w = \begin{bmatrix} a_x + c_x + d_x \\ a_y + c_y + d_y \\ \psi_g \end{bmatrix} \tag{10.13}$$

$$X_g = \frac{1}{2} \cdot \begin{bmatrix} a_x + b_x \\ a_y + b_y \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \psi_g \end{bmatrix} \tag{10.14}$$

However, singularities occur when P_1 and P_2 have the same Cartesian x-coordinate. In this case, c_x becomes zero and both d_x and ψ_g become undefined. This can be solved by restricting the calculations from potential zero divisions with the assumption that vector $\vec{d}$ always points towards the positive part of the longitudinal axis of the robot:

$$\vec{w} = \begin{cases} \vec{g} - \begin{bmatrix} 2 \\ 0 \end{bmatrix}, & if\ c_x = 0 \\ \vec{g} + \vec{d}, & if\ c_x > 0 \end{cases} \tag{10.15}$$

The distance between the waypoint and the goal will be set to $3\,m$ for the standard configuration of Thorvald. With both a laser scanner and a camera available, Thorvald will calculate the pose of the waypoint from laser scan measurements, before transitioning into using computer vision and *AprilTag* detection to finalize the alignment. Laser measurements will be extracted from *LaserScan* [96] messages which provide instant access to the distance at which a point is located. The corresponding angle needs to be calculated as follows:

$$\alpha_i = \angle OP_i = \alpha_0 + idx_i * \alpha_{inc} \tag{10.16}$$

in which idx is the index for the detected point P_i and α_{inc} is the angle increment specific for the Hokuyo laser, and α_0 represents the start angle of the scan.

AprilTag detections, on the other hand, do not provide distances and angles, but positions relative to the camera frame. Although the depth camera provides 3D data, it is assumed that the docking procedure takes place on a flat surface and that data can be simplified into two dimensions. The AprilTags are assumed to be located in the horizontal plane and the y-coordinates in the camera frame will be set to 0. The camera's z-axis defines the depth in the photo and should be mounted so that the axis aligns with the longitudinal axis of the robot, x_b in Figure 11-7. By doing so, the x-axis of the camera aligns with y_b and that the distance to a point, $d_{i,camera}$, can be calculated by using the Euclidean norm (Formula 3.6), and angle to the point, $\alpha_{i,camera}$ with trigonometry.

$$d_{i,camera} = \sqrt{z_i^2 + x_i^2} \tag{10.17}$$

$$\alpha_{i,camera} = \tan^{-1}\left(\frac{x_i}{z_i}\right) \tag{10.18}$$

11.4 Controller

This section describes the construction of a controller for the motion of Thorvald during the docking procedure. To ensure that the control design is feasible, integrable and transferable, a pose controller that is based on the kinematic model of a robot with differential drive is proposed.

11.4.1 Kinematic model

As described in Chapter 5 in section 7, the principles of the kinematic bicycle model can be used to create a simplified unicycle model for a Thorvald configuration with a differential drive. The following kinematic model is derived by using the center of rotation of the base frame as a reference point. By assuming low-speed and low acceleration planar motion and that the no-slip condition holds throughout the docking procedure, the following matrix representation can be used for Thorvald's motion:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos(\psi) \\ \sin(\psi) \\ 0 \end{bmatrix} \cdot v_x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdot \omega \tag{10.19}$$

11.4.2 Pose regulator

To enable the robot to navigate to a specific pose, a pose regulator based on polar coordinates will be used. As it is desired to control the robot based on Cartesian coordinates, Figure 11-11 illustrates the relationships between an initial pose and a desired goal pose in the Cartesian Space.

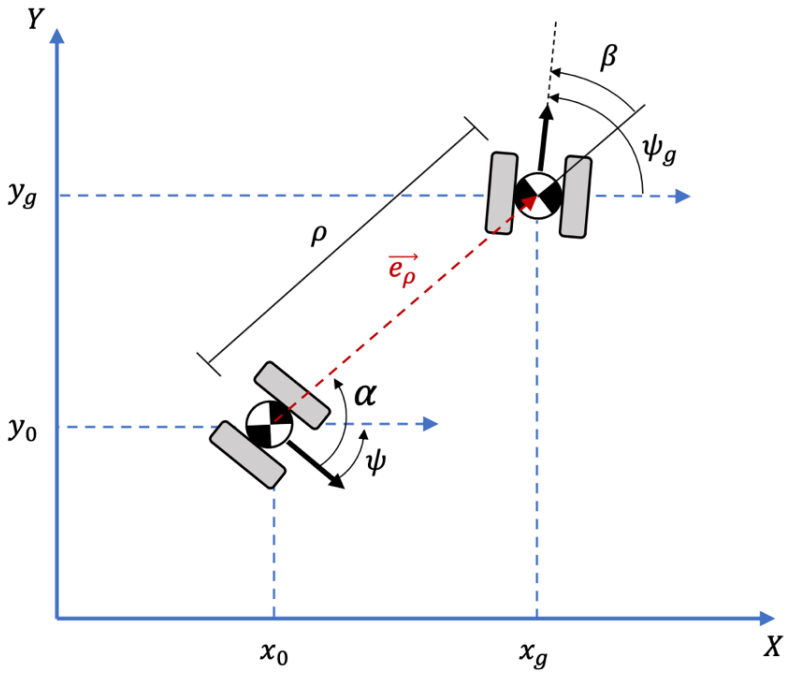


Figure 11-11: An illustration of the polar coordinate notation for the kinematic unicycle model in which (x_0, y_0, ψ) represents the initial pose, (x_g, y_g, ψ_g) the goal pose with respect to the global coordinate system. ρ represents the Euclidian distance from the robot to the goal, α represents the angle of the goal vector with respect to Thorvald's sagittal axis and ψ is the current heading angle. β represents the angle that the goal orientation ψ_g makes with the goal vector.

A few adjustments have been made to the polar coordinate relationships that were depicted and described in Chapter 5 in order to regulate to an arbitrary pose. ρ remains as the Euclidean distance from the initial to the final pose given by the goal vector $\overrightarrow{e_\rho}$ and α as the angle between the heading of the robot and the goal vector. β has been modified and now represents the angle that the goal pose orientation, ψ_g makes with the goal vector. These relationships are given by the following equations,

$$\rho = \sqrt{\Delta x^2 + \Delta y^2} \tag{10.20}$$

$$\alpha = atan2\left(\frac{\Delta y}{\Delta x}\right) - \psi \tag{10.21}$$

$$\begin{aligned}\beta &= -\psi - \alpha + \psi_g \\ &= \psi_g - atan2\left(\frac{\Delta y}{\Delta x}\right)\end{aligned} \tag{10.22}$$

where $\Delta x = x_g - x_0$ and $\Delta y = y_g - y_0$, which results in the following matrix representation of the robot's kinematic model expressed in polar coordinates:

$$\begin{bmatrix}\dot{\rho} \\ \dot{\alpha} \\ \dot{\beta}\end{bmatrix} = \begin{bmatrix}-\cos(\alpha) & 0 \\ \frac{\sin(\alpha)}{\rho} & -1 \\ -\frac{\sin(\alpha)}{\rho} & 0\end{bmatrix} \begin{bmatrix}v_x \\ \omega\end{bmatrix} \tag{10.23}$$

Recall that the motion command that will be sent to Thorvald is of type Twist, and that the control inputs will be the linear velocity, v_x, and the yaw rate, ω. In polar coordinates, the following non-linear control law will be used for the pose regulator:

$$v_x = k_\rho \cdot \rho \cdot \cos(\alpha) \tag{10.24}$$

$$\omega = k_\alpha \cdot \alpha + k_\rho \cdot \frac{\sin(\alpha)\cos(\alpha)}{\alpha} \cdot (\alpha + k_\beta \cdot \beta) \tag{10.25}$$

where k_ρ, k_α and k_β are constant gains. It can be shown that for small angles of α, the control law can be linearized to:

$$v_x = k_\rho \cdot \rho \tag{10.26}$$

$$\omega = k_\alpha \cdot \alpha + k_\beta \cdot \beta \tag{10.27}$$

This linear control is proposed used when $-10° < \alpha < 10°$ and is proven stable when $k_\rho > 0$, $k_\beta < 0$ and $k_\alpha - k_\rho > 0$. Outside this range, the non-linear design should be used, which is proven stable for $k_\rho, k_\alpha, k_\beta > 0$ [70].

The controller will be used both to navigate to the waypoint, and from the waypoint to the goal. However, the controller is designed so that the robot should obtain the desired heading at the waypoint, and the gains for the final alignment will, therefore, be adjusted so that the controller mainly regulates based on the range. When the robot reaches a point within the

tolerated range of the goal, the robot will stop, and the docking will be considered as completed.

A Simulink model is made for the pose regulator to analyze its response for the kinematic model. The unicycle was set to start from various initial conditions with the aim of ending up in $[x, y, \psi]^T = [0, 0, 0]^T$. The corresponding motion response is shown in Figure 11-12, where the initial positions are marked with circles. An illustration of the Simulink model can be found in Appendix II.

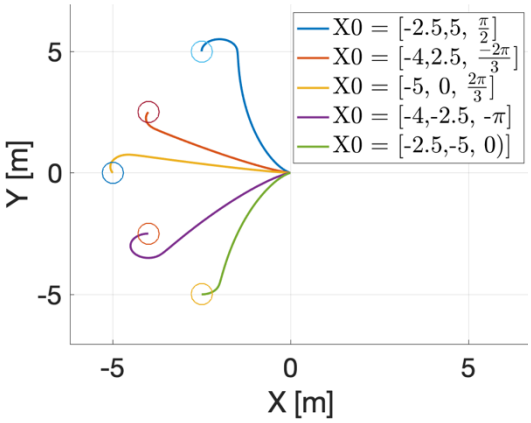


Figure 11-12: A graph that shows the motion response to commands for a unicycle kinematic model controlled with a pose regulator from five different initial conditions.

11.5 Localization

Because a laser scanner is available, *amcl* will be used for the localization of the robot, thus requiring a map and incoming laser data. The *gmapping* node will be used to perform SLAM and generate a laser-based map of the charging environment.

Figure 11-13 shows the gmapping procedure performed in Gazebo. The charging environment that is mapped is inspired by the environment that is used for testing at the

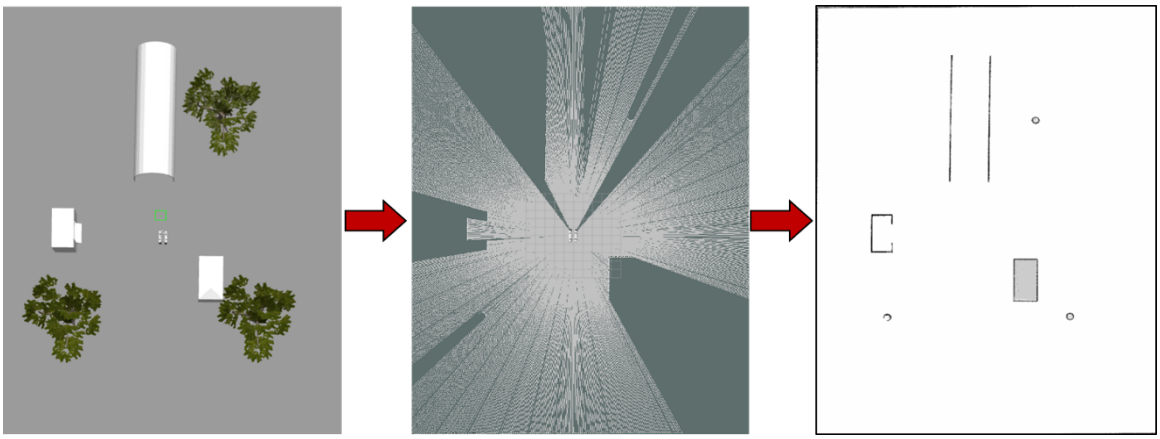


Figure 11-13: An illustration of Thorvald using SLAM to make a map of a simulation environment. The left image shows the simulation environment, the middle image shows RViz monitoring of the map generation and the right image shows the final map of the environment.

University. The top structure is a strawberry tunnel, and the leftmost structure is a mock-up design for a charging station. The rightmost structure represents an on-site office. However, a simplified charging environment will be made to test the function by using simulated laser scans because the virtual laser scan is incapable of measuring intensity. Therefore, the only structure in the environment will be the dock shown in Figure 11-5, thus making this structure the only object detected by the laser. A simplified algorithm will be used to distinguish the two poles from the laser data.

Finally, to obtain the initial configuration in front of the charging station, Thorvald will use its abilities to navigate on a topological map. An initial pose will be defined, from which the developed docking system will guide Thorvald to the charger.

This chapter has presented a complete solution to the docking problem. In the next chapter, the most important system components will be described in pseudocode.

12 System design

This chapter presents the most relevant algorithms used for the software. The algorithms are presented as pseudocode to provide overview of the functionality of the various functions.

12.1 Waypoint and goal pose generation

The following algorithm describes the function that generates poses for the waypoint and the goal. This function will be referred to in the next two sections when the techniques for localizing with the two sensor technologies are explained.

Algorithm 1 Calculate waypoints.

```
def calculate_waypoints(vector1, vector2):
    Desired distance from waypoint to goal = d_des
    Input vector with least angle = a
    Input vector with greatest angle = b
    Calculate vector c
    Calculate goal vector g
    if a and b have equal x-components:
        Waypoint vector w = g − [d_des, 0]^T
        Goal orientation = 0
    else
        Calculate vector d.
        Waypoint vector w = a + c + d
        Calculate goal orientation
    end if
    Define the goal pose and waypoint pose
return goal pose and waypoint pose
```

However, since the reference point on that will be used to control the standard configuration of Thorvald is not placed at its geometric center, a distance equal to the length from the center of rotation to the geometric center must be added to the goal position before executing the pose regulator.

12.2 Laser and reflective markers

Algorithm 2 on the next page describes how the laser data will be used to distinguish and localize points of interest. The points will first be sorted based on intensity measurements, and then sorted into clusters based on their mean distance. The mean distance and angle for each cluster will be used to define the two pillars, and moreover the vectors from which Algorithm 1 will generate waypoints.

Algorithm 2 Waypoint and goalpose from laser scan.

```
for received laser scan do
    Extract ranges, angle_min, angle_increment and intensities.
    Set intensity threshold
    for index, range in ranges do
        Extract current point intensity
        if intensity > intensity threshold then
            Calculate the angle to the point:
                angle = angle_min + index · angle_increment
            Store in intense_points[ranges, angles]
        end if
    end for
    for index, range in intense_points[ranges] do
        if range < mean(intense_points[ranges]) then
            Add to close cluster array
        else
            Add to far cluster array
        end if
    end for
    Closest point = close_cluster[mean_range, mean_angle]
    Furthest point = far_cluster[mean_range, mean_angle]
    Calculate waypoint and goal pose with the waypoint generator
end for
```

12.3 Camera and AprilTags

Algorithm 3 describes how the AprilTag detections will be used to localize the points of interest. The station gate and the charger dock will be identified with different pairs of tag IDs, the localization node with AprilTag detection will include multiple publishers so that the station, dock, and range have each their topic. The algorithm will first check if the charger or the station is detected and then use the distance and angle data to generate waypoints by using Algorithm 1.

Although the AprilTag detections provide sufficient information to determine complete poses for the waypoint and the goal, the function in Algorithm 1 will also be used for camera detection. The detections will not be used directly, because the small-scale experiments included some errors in orientation.

Algorithm 3 Waypoint and goalpose from AprilTag detections.

```
for received AprilTag data do
    if ID1 and ID2 are detected then
        The charger is detected
        Calculate distances and angles to ID1 and ID2
        Calculate waypoint and goal pose
        Publish waypoint and goal pose
    else if ID3 and ID4 are detected then
        The station gate is detected
        Calculate distances and angles to ID3 and ID4
        Calculate waypoint and goal pose
        Publish waypoint and station pose
    else
        No tag pair detected
    end if
    if ID0 is detected then
        The range marker is detected
        Calculate distance to ID0
        Publish distance to ID0
    end if
end for
```

12.4 Controller design

Algorithm 4 describes the motion controller for the function. For the pose regulator, a linear design will be used when the angle to the desired pose is less than 10 degrees. Below is also a Simulink model showing the design of the pose regulator.

Algorithm 4 Pose regulator.

```
for desired waypoint do
    Calculate ρ, α and β.
    while ρ > 0.2m & heading error ∈ ±5°  do
        if α ∈ ±5° then
            Use linearized control design
        else
            Use non-linear control design
        end if
        if linear velocity is outside limits then
            linear velocity equals upper/lower linear velocity limit
        else if angular velocity is outside limits then
            angular velocity equals upper/lower angular velocity limit
        end if
        Execute commands for linear velocity and angular velocity
    end while
end for
```

This chapter has proposed a solution to the problem of autonomously docking Thorvald in the charging station. Virtual and real-world experiments will now be conducted to evaluate the performance of the docking system.

13 Simulation of the concept

As parts of the modified version of the Waterfall method, the system design has been re-evaluated and improved several times to obtain the desired functionality. The software for the function has been constructed in modules, where the detection and controller have been developed separately.

This chapter describes experiments that have been conducted in Gazebo to validate the various system components before deploying them on the actual robot.

13.1 Goals

The main goal for the simulations was to collect data on the performance of the docking system and to obtain a set of results to compare with when running a real-world test.

13.2 Simulation set-up

The simulations were set up in Gazebo, as depicted in Figure 13-1 below. A smaller robot configuration than the standard configuration was the only one available for the real-world experiments, hence, a virtual model of the same configuration with a virtual laser scanner was used for the virtual experiments. The tolerance for the goal position was set to $0.2\ m$ and for the goal orientation to $\pm 10°$, as is described in the system specifications. The simulation of detection was only done with the virtual laser scanner because virtual camera technology was unavailable.

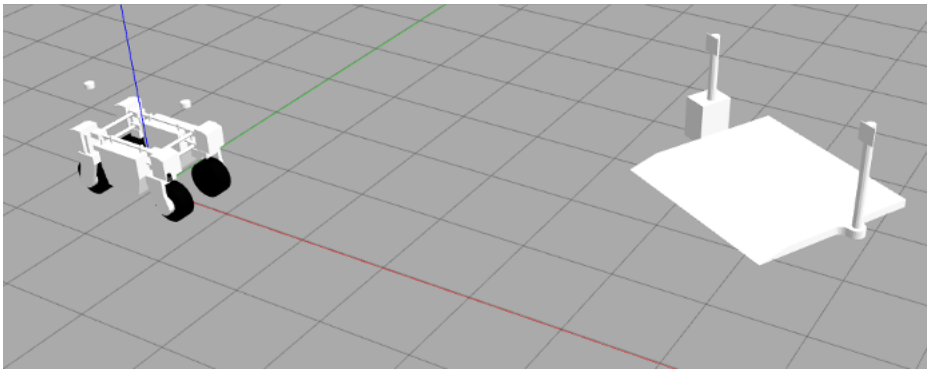


Figure 13-1: An illustration of the simulation environment used for the virtual tests of the docking system.

13.3 Steps

The robot had the same initial pose $[0, 0, 0]^T$, in every simulation, as in the illustration above. The dock was, on the other hand, moved in between every simulation to test the performance from various angles. Only a total of five experiments were conducted, as the performance of the function already has shown to be consistent through the final iterations of software development.

Results were gathered as position and heading errors between Thorvald's actual and desired pose, both at the waypoint and at the goal.

13.4 Results

The following two tables present the result from the simulations. In Table 13-1, generated desired poses are listed alongside the robot's actual position and orientation at these points. The poses are described by vectors as $X = [x, y, \psi]^T$.

Table 13-1: An overview of the data collected from the simulations in Gazebo.

Experiment	Goal pose		Waypoint	
	Desired	**Actual**	**Desired**	**Actual**
1	[9.99, 10.24, 1.57]	[9.91, 10.18, 1.569]	[9.99, 7.24, 1.57]	[9.99, 7.14, 1.571]
2	[14.18, -1.98, 0.657]	[14.00, -1.99, 0.651]	[11.79, -3.81, 0.657]	[11.72, -3.85, 0.651]
3	[7.47, 3.77, 0.63]	[7.33, 3.63, 0.63]	[5.04, 2.01, 0.63]	[4.97, 1.96, 0.625]
4	[7.16, -5.58, -0.93]	[7.03, -5.46, -0.93]	[5.36, -3.18, -0.93]	[5.30, -3.12, -0.94]
5	[7.97, -4.99, -0.173]	[7.84, -4.86, -0.168]	[5.01, -4.48, -0.173]	[4.92, -4.47, -0.173]

Table 13-2: An overview of statistical errors calculated from the relationship between the desired and actual poses presented in the previous table.

Error	Mean, μ	Standard deviation, σ
Goal position	$173.3\ mm$	$15.7\ mm$
Goal heading	$2.4 \cdot 10^{-3}\ rad$	$2.9 \cdot 10^{-3}\ rad$
Waypoint position	$88.4\ mm$	$7.4\ mm$
Waypoint heading	$4.4 \cdot 10^{-3}\ rad$	$4.0 \cdot 10^{-3}\ rad$

Table 13-2 provides the main results of the simulated experiments. These results will be used as a basis for comparison when conducting real-world experiments in the next chapter.

14 Early testing and validation

This chapter presents the last step in this development, a real-world test of the developed system.

14.1 Goals

Real-world experiments have been conducted to evaluate the performance of the proposed solution through validation of the various concepts used for the system.

Main test goal:

Based on the overall main goal for the project, the main objective of the experiment was to prove that the proposed docking system can be used as an autonomous navigation system for Thorvald to dock.

Sub-goals:

1) Validate detection of the charger with the Hokuyo lidar.

Determine if the laser detection concept can distinguish the charger and that the pose generator can determine a feasible waypoint and a goal pose from the data.

2) Validate detection of the charger with the RGB-D camera and AprilTags.

Determine if the camera and corresponding computer vision algorithm can be used to distinguish the charger and that it provides sufficient data for the pose generator.

3) Validate the functionality of the pose regulator.

Determine if the control design can navigate Thorvald to the desired poses.

14.2 Steps

The experiment was set up outside, next to the robotics lab at the University. An omnidirectional configuration of Thorvald was used, as a standard configuration was unavailable. Therefore, the experiments were also used to verify the transferability of the system. The following steps were planned for the experiments:

1. Perform necessary preliminary tasks before startup.
2. Create and store a map of the test environment.
3. Set up the dock with reflective markers and AprilTags.
4. Run tests to tune the control parameters.
5. Conduct experiments while using the laser for detection.
6. Conduct experiments while using the depth camera for detection.

For each experiment, the robot was set to start with a new initial position and orientation. Initial conditions were limited by the used sensors' field of view because the experiment was conducted on an uneven surface, thus not allowing continuous detection. The markers were, therefore, placed so that they were perceived by the sensors from the initial position.

14.3 Experimental set-up

Hardware

Two aluminum pipes were used to mimic the pillars of the dock and the entrance of the station. For experiments with the laser, a Hokuyo UTM-30LW-EX 2D lidar and two reflective triangles, usually used to mark trailers for cars, were used. For experiments with the AprilTag detector, the RealSense D415 camera and tags with ID: 0 and 5 from tag family 36h11 were used. Figure 14-1 depicts the material that was used for the experiments.

Software

The experiment was monitored through RViz, and the *gmapping* node was used to construct a map of the test environment so that the *amcl* package could be used for localization. The navigation system was set up so that the goal and waypoints that were generated in the base frame were transformed into map coordinates.

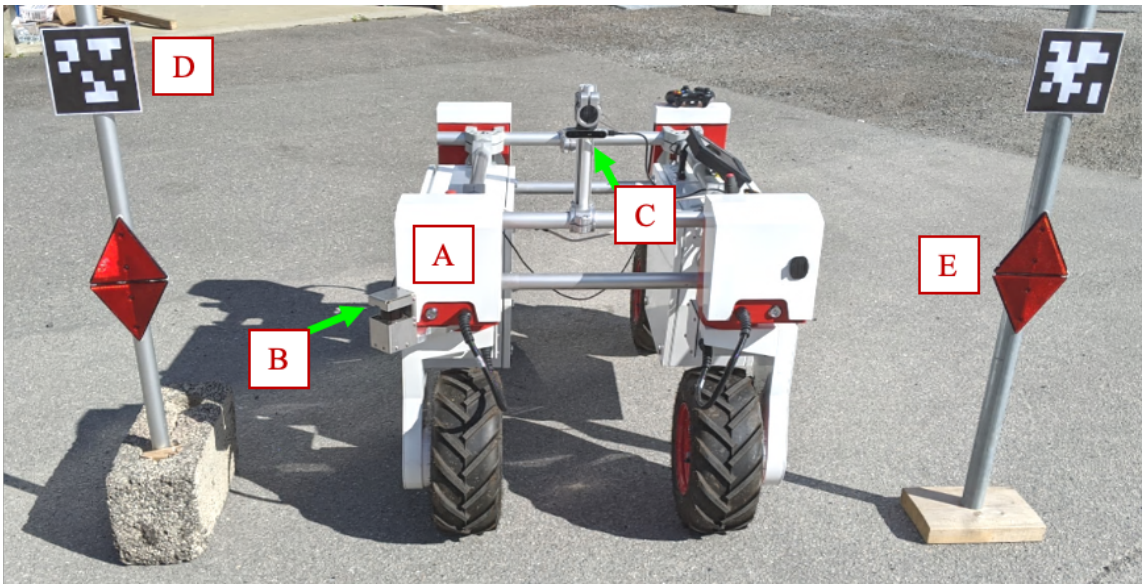


Figure 14-1: An image of the various components that will be used for the real-world experiments.

14.4 Results

The next two sections present results from the various experiments conducted with the laser and the camera. Results are presented in terms of position error and heading error at the desired poses, and error in detections.

Figure 14-2 and Figure 14-4 consist of plots that show errors from each of the experiments, where black crosses mark the error for the respective experiments, the red line marks the mean error and the green dashed lines depicts the error plus and minus one standard deviation. The tables in this section, Table 14-1, Table 14-2, Table 14-3 and Table 14-4, provide results from the experiments as numerical data.

14.4.1 Results from experiments with laser

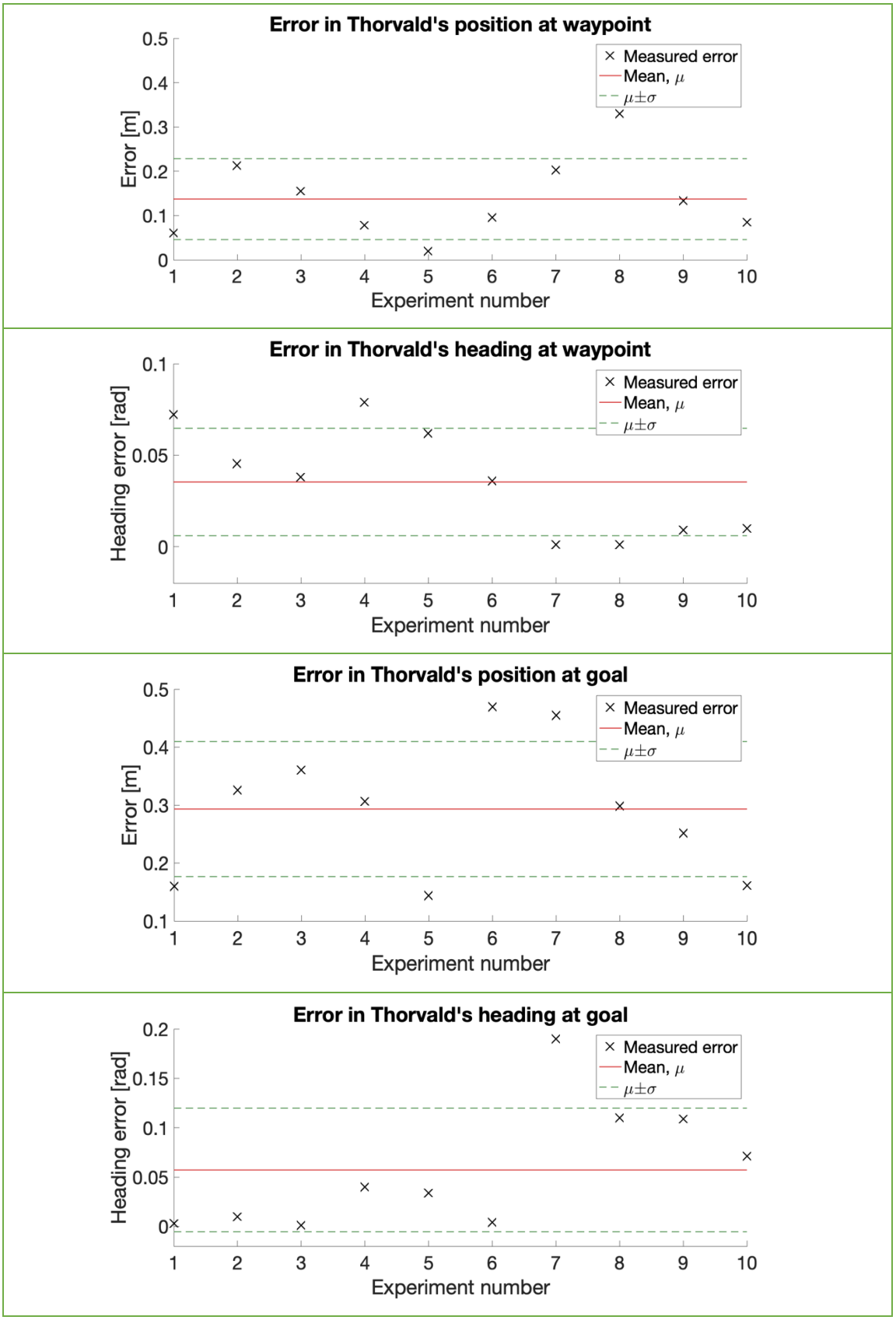


Figure 14-2: Four plots that show the results from real-world experiments with the Hokuyo lidar. From the top: Error in Thorvald's position at the waypoint, error in Thorvald's heading at the waypoint, error in Thorvald's position at the goal, and error in Thorvald's heading at the goal.

Table 14-1: An overview of the main results from the real-world experiment with the Hokuyo lidar.

Error	**Mean, μ**	**Standard deviation, σ**
Goal position	293 mm	117 mm
Goal heading	0.057 rad	0.063 rad
Waypoint position	137 mm	91 mm
Waypoint heading	0.035 rad	0.029 rad

Table 14-2: An overview of results for detection accuracy and precision with the Hokuyo laser.

Detection error	**Mean, μ**	**Standard deviation, σ**
Goal offset	123 mm	85 mm
Waypoint offset	103 mm	72 mm

Figure 14-3 shows images from the experiments with the laser, where the desired position is depicted by a check mark and the actual position by a yellow arrow.

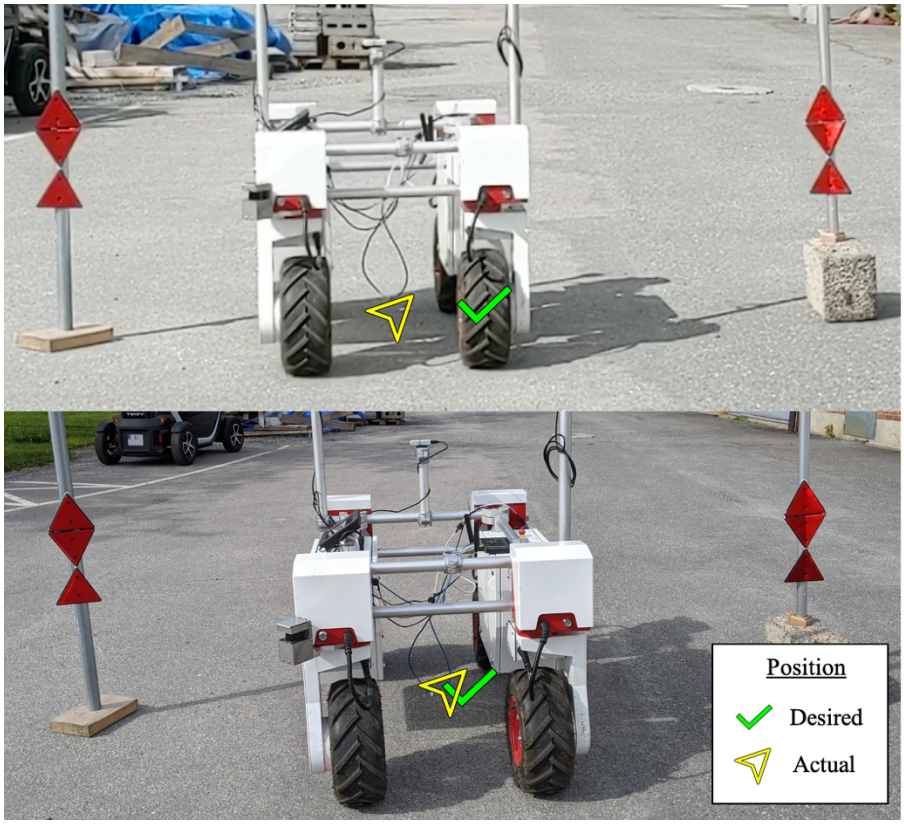


Figure 14-3: Two images showing different test results for docking with the Hokuyo lidar. The arrow and the check mark respectively represent the actual and the desired position at when finished docking. In the upper image, Thorvald has ended up outside the tolerances set for the docking.

14.4.2 Results from tests with the RGB-D camera and AprilTags

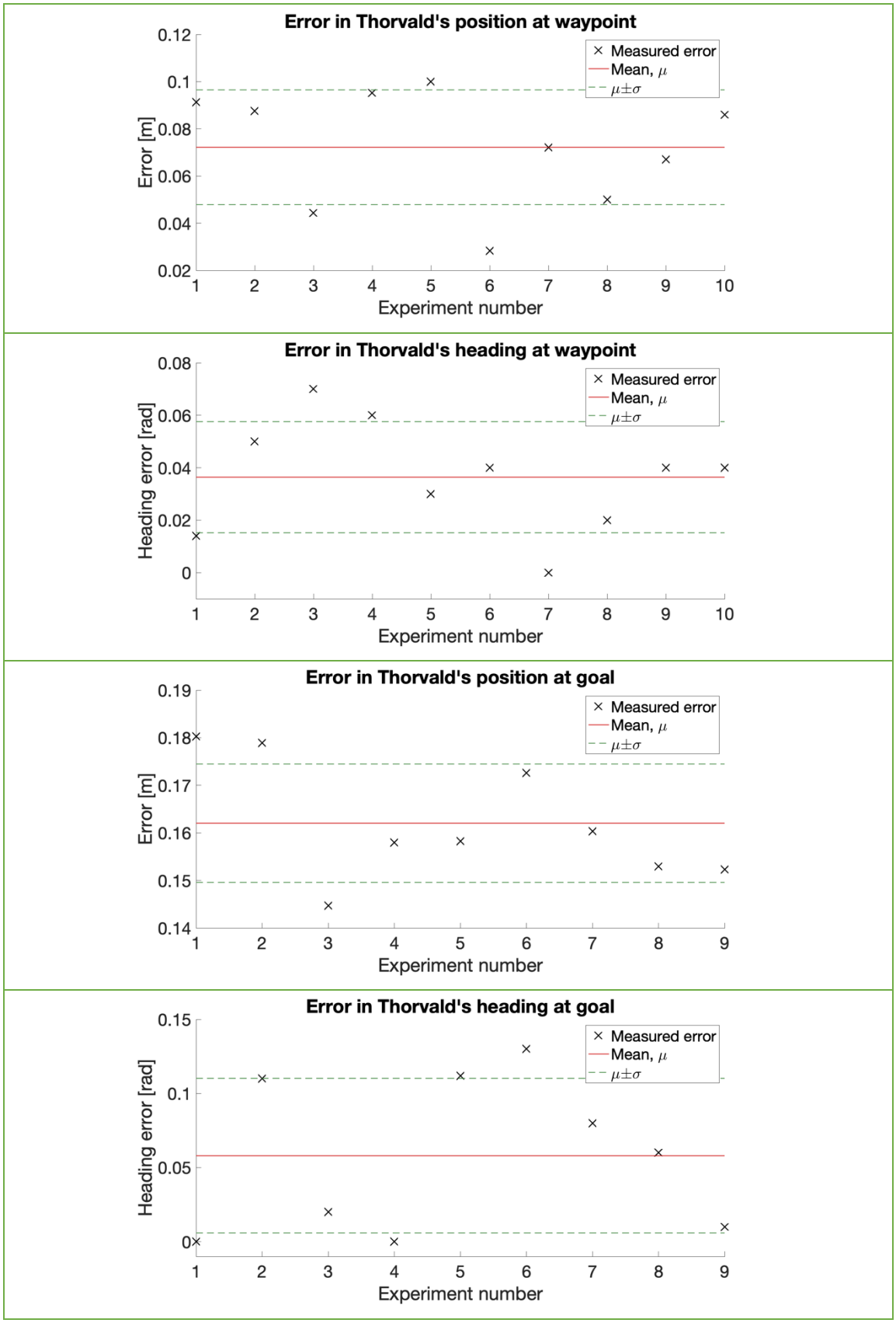


Figure 14-4: Four plots showing the results from real-world experiments with the RGB-D camera. From the top: Error in Thorvald's position at the waypoint, error in Thorvald's heading at the waypoint, error in Thorvald's position at the goal, and error in Thorvald's heading at the goal.

Table 14-3: An overview of the main results from the real-world experiment with the RGB-D camera.

Error	**Mean, μ**	**Standard deviation, σ**
Goal position	$0.162\ m$	$0.0124\ m$
Goal heading	$0.058\ rad$	$0.052\ rad$
Waypoint position	$0.072\ m$	$0.023\ m$
Waypoint heading	$0.036\ rad$	$0.021\ rad$

Table 14-4: An overview of results for detection accuracy and precision with the AprilTag detector.

Detection error	**Mean, μ**	**Standard deviation, σ**
Goal offset	$0.114\ m$	$0.069\ m$
Waypoint offset	$0.099\ m$	$0.047\ m$

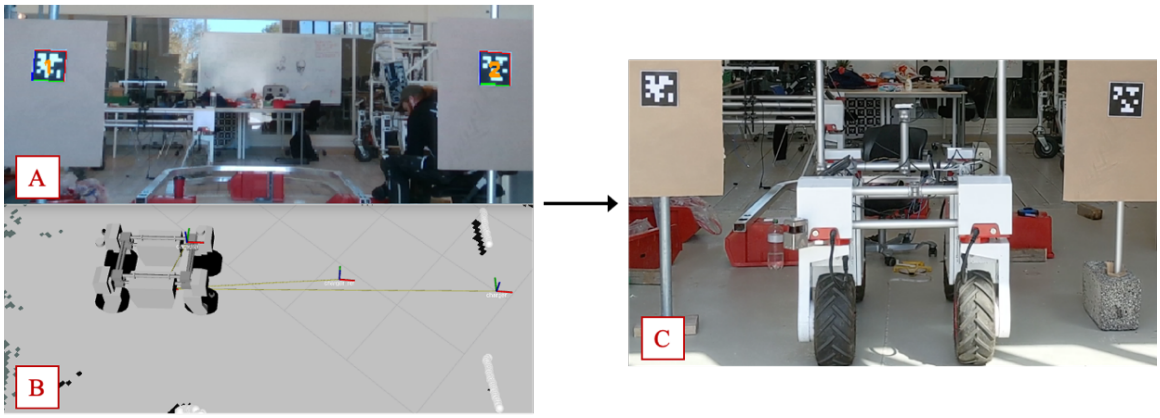


Figure 14-5: Images that show the process of docking when using AprilTags. A: A tag detection image. B: An image of transforms used to visualize the waypoint and goal pose in RViz. C: An image of Thorvald, successfully docked in between the pillars.

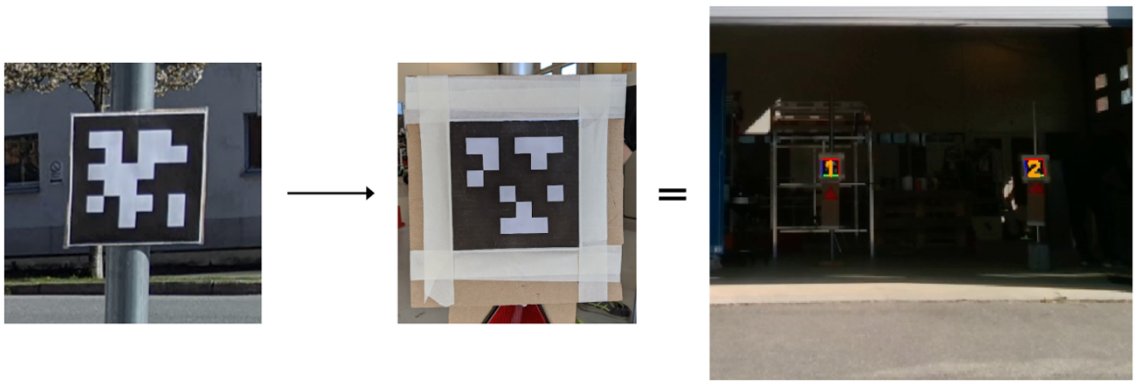


Figure 14-6: An illustration that shows the modifications that were done with white tape to obtain sufficient contrast for the AprilTags. These modifications significantly improved the tag detections, as is shown in the rightmost image, where the tags are detected almost without illumination

On the previous page, Figure 14-5 shows images from experiments conducted with the AprilTag detector algorithm. The pictures show a detection image, broadcasting of frames from the obtained data and an image of Thorvald successfully docked. Figure 14-6 shows modifications that were done on the AprilTag to improve the detection performance.

This chapter has presented real-world experiments that have been conducted for the autonomous docking system that has been developed. The next chapter will discuss the related results in more detail and reflect upon essential parts of the development process.

15 Process evaluation and discussion

This section presents an evaluation of the development process based on the process chart in Figure 3-5 and Figure 3-6. Results from experiments conducted throughout the process are discussed, and the goals for the project are revisited one last time for verification.

15.1 Process evaluation

Investigation and planning

An early discovery was that robot navigation is complex and requires attention to many aspects to become operational in various environments. Due to the strict timeframe, the construction of a framework and a thought-out plan for the thesis was essential. In phase one, the main goal, sub-goals, and corresponding activities were formulated to ensure motivation and purpose for the process. A milestone plan was made for the project, and many limitations were set to ensure feasibility. The limitations helped narrow down the scope of the thesis but may have had consequences as, for instance, energy consumption for Thorvald was not considered.

Research and development

A large amount of time was spent on information gathering due to the immensity of the robotics field. Insight in research conducted on mobile robots, navigation and controls provided both a theoretical and practical understanding of the problem, as well as inspiration for the system specifications. However, the timeframe resulted in limitations for the span of the investigation, and better approaches than the ones considered may exist.

The function analysis conducted in Chapter 8 revealed the need for three secondary functions. Navigation to the charging station was assumed completed by existing topological navigation. As both navigation into the station and the docking itself were considered very similar, it was decided to look for solutions that could complete both tasks with the same technology to reduce the need for additional functionality and sensors.

Analysis and improvements

The circumstances around COVID-19 peaked during the process and made parts of the project rather challenging. The University was locked down, and it was not possible to get access to the robot until it opened up again in late April 2020. As a result, the majority of the development relied on simulation and results from small-scale experiments that were set up at home.

The overall impression of the process is satisfactory and corresponds to the personal expectations of the author in consideration of the basic initial knowledge about robotics and software. On reflection, certain parts of the process could have been done differently to possibly obtain better results:

- Extensive testing of the chosen sensor technologies should have been conducted to determine the probability of false positives and other errors.

- A more comprehensive patent search should have been conducted to ensure that the technology that has been used to develop the function is not protected.
- It would have been convenient to specify the system more during the initial phase so that the scope of the investigation could be narrowed down to only what was relevant for the docking system.
- The survey that was sent out to gather feedback from experienced people could have been sent out to more people to get more opinions on the concepts.
- More sensors could have been evaluated, as the selection of sensors in this thesis may have been biased because of their availability.
- Advanced control strategies could have been investigated to search for controllers with better performance and greater robustness to uncertainties and external disturbances.

15.2 System design evaluation

Important assumptions

The system was designed for 2D data and motion in a 2D space. However, the operational environment of the agricultural robot is far from flat and is very likely to contain, for instance, rocks and irregularities. The algorithm used to calculate the distance to AprilTag detections measures the distance to the tags based on only the z- and x-axis of the camera. As the y-axis is not considered, the distance to a tag will be incorrectly calculated when the robot has non-zero pitch and roll angles. The consequence will be incorrect localization of the goal and waypoint and errors in the alignment. These challenges occurred during the final experiment, as Thorvald was pitched when detecting the tags. As a result, the goal pose was generated away from the center of the dock. Therefore, the calculation should be upgraded to three dimensions to ensure correct measurements.

Sensor technology

Multiple trade-offs can be discussed for the choice of sensors. Based on the conducted experiments, the 2D lidar will provide long-range detection with high accuracy but can only distinguish objects by using intensity measurements. The generation of waypoints may, therefore, fail when the station gate and the dock are both inside the laser's field of view. Feedback received through the expert survey proposed replacing the 2D lidar by a 3D lidar to allow for differentiation through feature extraction.

On the contrary, an RGB-D camera and the AprilTag detector will enable Thorvald to distinguish the charging station gate from the dock. However, experiments indicated a limited range for the camera. Hence, the proposed system allows for a combination of both sensors to account for both long-range detection and the ability to distinguish landmarks.

In the early stages, challenges due to environmental conditions such as lighting and weather were discussed for the sensors. Small-scale experiments with the depth camera and the AprilTag detector indicated negative influences from sunlight, though only for the

orientation of the tags. The waypoint generator is designed to only rely on their position, which eliminates the error effects on the system. Furthermore, a discovery during the real-world experiments was that the AprilTags need a homogeneous background with sufficient contrast to be detected. For the experiments, cardboard and white tape was used to enhance contrast for the AprilTags, which resulted in significantly improved detections as was shown in Figure 14-6. The modifications enabled the AprilTag detector to detect the tags, even with minimal illumination.

To determine the intensity threshold for the laser, triangular reflectors that are commonly used to mark the back end of cars and trailers were used. Similar objects will likely appear in Thorvald's operational environment and may result in false-positive detections. Due to the construction of the laser detection algorithm, false positives may lead to an incorrect calculation of waypoints.

Table 14-2 and Table 14-4 show results for the detection in real-world experiments. The mean offset for the localization of the waypoint and goal with the laser and the AprilTags were both over 100 mm, which was the maximum offset that was set in the specifications. However, the RViz monitoring indicated very stable and accurate detections and localization of the waypoint and goal on the map. Hence, these errors may have occurred due to poor robot pose estimates during the generation of the waypoint and the goal, as this was not done continuously.

For Thorvald to be able to work all 24 hours, the docking system also needs to be appropriate for nighttime operation and all types of weather. These conditions have not been considered in this report but will require attention before the system is deployed on the robot. Assumedly, the laser and camera will encounter challenges with detection when it snows, or during heavy fog. Immediate solutions can be to trust the map and use topological navigation to navigate even closer to the desired position. However, doing so will require another module for robot localization, as amcl also relies on laser data.

Control strategy

The pose regulator that was chosen to control Thorvald's motion during the docking procedure has proven to work well for its application. Although the kinematic unicycle model does not directly depend on any physical properties of the robot, it uses the center of rotation as a reference, which differs among the various configurations of Thorvald. Other references may be used but will require minor modifications to parameters in the controller or for the generation of waypoints. Despite this, the controller has proven to be very transferable in simulation, and easy to integrate with Thorvald's existing system.

The proposed controller was tuned for satisfactory performance during testing with the omnidirectional configuration. However, results from the experiment show that Thorvald often ended up outside the tolerances that were specified. What caused the errors is not clear, but a possible reason is a poorly tuned controller. Results from Table 14-1 where the pose regulator was tested with laser detection show large errors for the offset in position at

the goal, where the mean was calculated to approximately 30 cm. The mean offset at the waypoint was lower, approximately 140 mm, which indicates that the tuning of the simplified control strategy for final alignment was insufficient.

The controllers were, however, tuned in between experiments with the laser and the camera, and showed significant improvements where the mean offset at goal for the AprilTag experiments was calculated to 162 mm, which is inside the tolerances that were set in the specifications. The mean offset at the waypoint was only 72 mm, which indicates a very good performance from the pose regulator, as the tolerance was set to 200 mm. Though, to determine the optimal controller gains for the pose regulator, extensive testing will need to be conducted.

Localization

Amcl was chosen to perform the robot localization. However, amcl requires a detailed map and robust laser measurements to provide accurate pose estimates. For the real-world experiments, some cars that were present during the mapping procedure were moved during the experiments. This change in the environment may have led to a loss of reference for the localization, thus poor pose estimates. RViz monitoring indicated that the inaccurate localization of Thorvald during the experiments, which may have been one of the causes of the errors in the results.

15.3 Personal growth

The development of an autonomous docking system for Thorvald has also led to personal growth. The project allowed me to use both my theoretical knowledge and practical understanding to take on real-world challenges. The learning curve has been steep, but the development has given me unique knowledge that I am sure will become useful in the future. I have also become more familiar with methods that can be used in any development process. Although it is hard to follow every step of a method, I have discovered that it is possible to modify methods to make them more appropriate for personal use.

Besides learning about theories and methods, I have also been lucky to have talented people sharing their knowledge to help me with the project. I have realized that I still have much to learn, but that people and literature can give access to the knowledge I need.

The most important personal takeaway from this master's thesis is, however, my improved understanding of robotics, controls, and autonomy. By working hands-on with sensors and data tools, I have improved my understanding of the intervention between hardware and software and become more experienced with robots.

16 Conclusion

An autonomous navigation system has been planned, realized, and developed according to the main goal for this master's thesis. This report describes the process of developing a navigation system that is integrable with the Thorvald concept and can be used by various configurations with minor adjustments.

The system has been developed with IPD methodology, where various design methods have been used to continuously specify the system while converging into a solution that corresponds to the system goals. Software for the system has been developed and verified through virtual and real-world experiments.

This navigation system function will take the Thorvald concept one step closer to a fully autonomous profile and contribute to a reduction in yield reduction caused by humans and heavy machinery.

The main results of the project are presented in the following list:

16.1 Main results

- A robust docking system that can be used with either an RGB-D camera, a lidar or a combination of both has been developed.
- The AprilTag detector which is used with the Intel RealSense D415 RGB-D camera is robust up to 4 meters, given that the tags are visible and placed on a background with contrasts.
- Laser detection with Hokuyo UTM-30LX-EW is proven to be sufficient up to at least 10 meters with a clear line of sight, also when directly exposed to sunlight.
- Experiments have indicated that reflective markers with prism structure can provide intensity measurements approximately four times greater than their surroundings for the Hokuyo laser.
- A systematic detection algorithm has been derived to generate a goal pose and a corresponding waypoint based on the position of two landmarks.
- A pose regulator has been designed based on the kinematic model of a robot with a differential drive. The pose regulator enables navigation to specified poses.
- Final experiments gave the following results for laser detection:
    - Error in goal detection:
      $\mu = 0.123\ m,\ \ \sigma = \ 0.085\ m$
    - Error in waypoint detection:
      $\mu = 0.103\ m,\ \ \sigma = \ 0.072\ m$
- Final experiments gave the following results for AprilTag detection:
    - Error in goal detection:
      $\mu = 0.114\ m,\ \ \sigma = \ 0.047\ m$
    - Error in waypoint detection:
      $\mu = 0.099\ m,\ \ \sigma = \ 0.047\ m$

- Final experiments gave the following results for the pose regulator:
  - Error in position at goal:
    $\mu = 0.162\,m,\ \ \sigma = \ 0.0124\,m$
  - Error in heading at goal:
    $\mu = 0.058\,rad,\ \ \sigma = \ 0.052\,rad$
  - Error in position at waypoint:
    $\mu = 0.072\,m,\ \ \sigma = \ 0.023\,m$
  - Error in heading at waypoint:
    $\mu = 0.036\,rad,\ \ \sigma = \ 0.021\,rad$
- Amcl can be used for localization of the robot, given that maps remain unchanged.
- The docking system can navigate Thorvald from an initial point outside a charging station to a final configuration inside that aligns with the charger.
- The software is written in Python using the *rospy* API, which makes it fully compatible with the ROS framework.

16.2 Recommendations

Immediate recommendations for the docking system are:

- Tune the parameters of the pose regulator.
- Invest in better reflective markers.
- Ensure sufficient contrast for the AprilTags.
- Conduct extensive real-world tests.
- Illuminate the AprilTags to enable operation at night.
- Determine the need for maintenance and calibration of the sensors.

16.3 Future work

The following list can be used as inspiration for future work:

- Investigate algorithms for the removal of false positives in laser detections.
- Improve the system by including 3D data in calculations so that uneven ground does not affect the developed system.
- Implement functionality for dynamic navigation considerations such as obstacle detection and avoidance.
- Fuse the data from laser and AprilTag detections to enhance the robustness of the system.
- Conduct comprehensive testing of the sensor in various environments and with the presence of reflective objects that may appear in Thorvald's operational environment to propose a final intensity threshold that provides a minimal number of false positives.
- Experiment with other solutions for robot localization.
- Investigate advanced control strategies for pose regulation.
- Invest in a 3D lidar for landmark classification in laser measurements.

17 Bibliography

This chapter provides an overview of the sources that have been used for information gathering. The written sources are separated from the web sources.

17.1 Literature and written sources

[1] U. Nations, *World population prospects 2019*, no. 141. 2019.

[2] D. Epron, C. Plain, F.-K. Ndiaye, P. Bonnaud, C. Pasquier, and J. Ranger, *Effects of compaction by heavy machine traffic on soil fluxes of methane and carbon dioxide in a temperate broadleaved forest*, vol. 382. 2016.

[3] A. J. Thomasson, *World map of the status of human-induced soil degradation*, vol. 52, no. 3–4. 1992.

[6] M. Bergerman, J. Billingsley, J. Reid, and E. Van Henten, "Robotics in agriculture and forestry," in *Springer Handbook of Robotics*, Springer International Publishing, 2016, pp. 1463–1492.

[7] L. Grimstad, *Modular , Mobile Robots for Applications in the Agricultural Domain Modulære , mobile roboter for jordbruket*. Ås, 2018.

[8] M. M. Andreasen and L. Hein, *Integrated Product Development*. Springer, 1997.

[9] E. B. Magrab, *Integrated Product and Process Design and Development: The Product Realization Process, Second Edition (Environmental & Energy Engineering)*, 2nd ed. New York: CRC Press, 2009.

[11] S. Pugh, *Total Design: Integrated Methods for Successful Product Engineering*. Pearson Education, 1991.

[12] A. B. Badiru, *Systems Engineering Models*, 1st Editio. Boca Raton : Taylor & Francis, a CRC title, part of the Taylor &: CRC Press, 2019.

[14] B. Eberle, *Scamper: Games for Imagination Development*. Prufrock Press, 1996.

[15] A. F. Osborn, *Applied Imagination: Principles and Procedures of Creative Thinking*, 1st ed. New York: Scribner, 1953.

[16] Mind Tools, "SMART Goals - Time Management Training," *Mindtools.Com*. 2017.

[30] P. A. Tipler and G. Mosca, *Physics for Scientists and Engineers*, 6th ed. W.H. Freeman, 2003.

[31] D. C. Lay, S. R. Lay, and J. J. McDonald, *Linear Algebra and Its Applications*, 5th ed. Pearson Education, 2016.

[32] K. Gieck and R. Gieck, *Engineering formulas*, 8th ed. McGraw-Hill, 2006.

[36] G. Dudek and M. Jenkin, "Inertial sensing, GPS and odometry," in *Springer Handbook of Robotics*, Springer International Publishing, 2016, pp. 737–751.

[37] P. Robertson, *Introduction to SLAM Simultaneous Localization And Mapping*. Cognitive Robotics, 2005.

[38] C. Stachniss, J. J. Leonard, and S. Thrun, "Simultaneous localization and mapping," in *Springer Handbook of Robotics*, Springer International Publishing, 2016, pp. 1153–1175.

[41] L. E. Kavraki and S. M. LaValle, "Motion planning," in *Springer Handbook of Robotics*, Springer International Publishing, 2016, pp. 139–161.

[42] G. Carbone and F. Gomez-Bravo, "Motion and Operation Planning of Robotic Systems," vol. 29, Springer International Publishing, 2015, pp. 3–27.

[43] J. Minguez, F. Lamiraux, and J. P. Laumond, "Motion planning and obstacle avoidance," in *Springer Handbook of Robotics*, Springer International Publishing, 2016, pp. 1177–1201.

[44] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe, *Mobile Robot Positioning: Sensors and Techniques*, vol. 14, no. 4. 1997.

[45] R. Siegwart and I. R. Nourbakhsh, *An Introduction to Autonomous Mobile Robots*, vol. 1, no. 1. Massachusetts Institute of Technology, 2004.

[46] H. R. Everett, *Sensors for Mobile Robots: Theory and Application*. 1995.

[48] W. Dargie and C. Poellabauer, *Localization*. John Wiley & Sons Ltd., 2010.

[49] A. N. Khojasteh, M. Jamshidi, E. Vahedu, and S. Telikani, *Introduction to Global Navigation Satellite Systems and Its Errors*, vol. 3, no. May. 2016.

[51] S. Liu, M. M. Atia, T. B. Karamat, and A. Noureldin, *A LiDAR-aided indoor navigation system for UGVs*, vol. 68, no. 2. 2015.

[54] J. Wojtanowski, M. Zygmunt, M. Kaszczuk, Z. Mierczyk, and M. Muzal, *Comparison of 905 nm and 1550 nm semiconductor laser rangefinders' performance deterioration due to adverse environmental conditions*, vol. 22, no. 3. Versita, 2014.

[55] R. Szeliski, *Computer Vision*. London: Springer London, 2011.

[56] W. K. Pratt, *Digital Image Processing*, vol. 19, no. 3. 1994.

[57] S. J. Russell and P. Norvig, *Artificial Intelligence A Modern Approach; PearsonEducation*. 2003.

[58] N. O'Mahony *et al.*, *Deep Learning vs. Traditional Computer Vision*, vol. 943, no. April. 2020.

[59] R. Girshick, J. Donahue, T. Darrell, and J. Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014.

[60] J. Redmon and A. Farhadi, *YOLOv3: An Incremental Improvement*. 2018.

[61] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, vol. 2016-Decem. .

[62] S. Aukstakalnis, *Practical Augmented Reality: A Guide to the Technologies, Applications, and Human Factors for AR and VR*. Addison-Wesley, 2016.

[63] P. D. Howe and M. S. Livingstone, *Binocular vision and the correspondence problem*, vol. 5, no. 8. Association for Research in Vision and Ophthalmology (ARVO), 2005.

[64] S. Statler, *Beacon Technologies*. Apress, 2016.

[66] A. Packard, K. Polla, R. Horowitz, and F. Borelli, *ME 132, Dynamic Systems and Feedback Class Notes*. Berkeley, US, 2005.

[67] R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*, vol. 83, no. 4. 2008.

[68] R. Rajamani, *Vehicle Dynamics and Control*, 2nd ed. Boston, MA: Springer US, 2012.

[69] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, vol. 53, no. 9. London: Springer London, 2009.

[70] P. Corke, *Robotics, Vision and Control*, vol. 118, no. 6. Cham: Springer International Publishing, 2017.

[71] Y. Kuwata, J. Teo, G. Fiore, E. Frazzoli, S. Karaman, and J. P. How, "Real-time Motion Planning with Applications to Autonomous Urban Driving," vol. 17, no. 5, IEEE Transactions on Control Systems Technology, 2009, pp. 1105–1118.

[77] M. Wermelinger, M. Bjelonic, P. Frankenhauser, D. Jud, and M. Hutter, *Programming for Robotics Introduction to ROS*. Zurich, 2020.

[81] T. Foote, *Tf: The transform library*. 2013.

[89] H. Yuen, J. Princen, J. Illingworth, and J. Kittler, "Comparative study of Hough Transform methods for circle finding," in *Image and Vision Computing*, vol. 8, no. 1, Elsevier, 1990, pp. 71–77.

[92] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, pp. 3400–3407.

[93] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, 2016, pp. 4193–4198.

17.2 Web sources

[4] Jäti weed-killing robot: "This robot kills weeds with lasers – RealAgriculture." [Online]. Available: https://www.realagriculture.com/2017/11/this-robot-kills-weeds-with-lasers/. [Accessed: 24-Jan-2020].

[5] Avo weeding robot: Ecorobotix, "The autonomous robot weeder from Ecorobotix," 2017. [Online]. Available: https://www.ecorobotix.com/en/autonomous-robot-weeder/. [Accessed: 24-Jan-2020].

[10] NMBU strategy: NMBU, "Strategi for verdiskaping og innovasjon 2019-2022 | Norges miljø- og biovitenskapelige universitet," 2019. [Online]. Available: https://www.nmbu.no/om/strategi/innovasjon. [Accessed: 13-Feb-2020].

[13] Waterfall method: TutorialPoint, "SDLC - Waterfall Model - Tutorialspoint," *Tutorials Poiints*, 2020. [Online]. Available: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm. [Accessed: 03-Feb-2020].

[17] About Ubuntu: Canonical, "Ubuntu Project: About," *ubuntu.com*, 2016. [Online]. Available: https://ubuntu.com/about. [Accessed: 24-Jan-2020].

[18] About Python: Python Foundation, "About Python™ | Python.org," *About Python*, 2016. [Online]. Available: https://www.python.org/about/. [Accessed: 24-Jan-2020].

[19] About Atom: GitHub Inc, "Why Atom?" [Online]. Available: https://flight-manual.atom.io/getting-started/sections/why-atom/. [Accessed: 15-May-2020].

[20] About ROS: Open Source Robotics Foundation, "ROS.org | About ROS," *ROS.org*, 2014. [Online]. Available: https://www.ros.org/about-ros/. [Accessed: 24-Jan-2020].

[21] ROS Melodic: Open Source Robotics Foundation, "melodic - ROS Wiki." [Online]. Available: http://wiki.ros.org/melodic. [Accessed: 13-Mar-2020].

[22] About Gazebo: I. Saito, "Gazebo." [Online]. Available: http://gazebosim.org/. [Accessed: 15-May-2020].

[23] About Draw.io: JGraph, "jgraph/drawio: Source to app.diagrams.net." [Online]. Available: https://github.com/jgraph/drawio. [Accessed: 15-May-2020].

[24] About SolidWorks: Solid Solutions, "Solidworks 3D CAD Packages," *www.solidsolutions.co.uk*, 2016. [Online]. Available: https://www.solidworks.com/product/solidworks-3d-cad. [Accessed: 15-May-2020].

[25] *urdf* documentation: D. Coleman, "urdf - ROS Wiki," *last edited*, 2014. [Online]. Available: http://wiki.ros.org/urdf. [Accessed: 16-Mar-2020].

[26] About MATLAB: Mathworks, "MATLAB - Mathworks - MATLAB & Simulink," *Www.Mathworks.Com*, 2016. [Online]. Available: https://se.mathworks.com/products/matlab.html?s_tid=hp_ff_p_matlab. [Accessed: 15-May-2020].

[27] About Simulink: Mathworks, "Simulink - Simulation and Model-Based Design - MATLAB & Simulink," 2015. [Online]. Available: https://se.mathworks.com/products/simulink.html?s_tid=hp_ff_p_simulink. [Accessed: 15-May-2020].

[28] ISO 9000: International Organization for Standardization, "ISO 9000:2015(en), Quality management systems - Fundamentals and vocabulary," *International Organization for Standardization Online Browsing Platform*, 2015. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:9000:ed-4:v1:en. [Accessed: 13-Feb-2020].

[29] ISO 9001: International Organization for Standardization, "ISO 9001:2015(en), Quality management systems - Requirements," 2020. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:9001:ed-5:v1:en. [Accessed: 13-Feb-2020].

[33] About the Norwegian Patent Board: Regjeringen, "Patentstyret." [Online]. Available: https://www.regjeringen.no/no/dep/nfd/organisation/etater-og-virksomheter-under-narings--og-fiskeridepartementet/Subordinate-agencies-and-institutions/patentstyret-/id435115/. [Accessed: 21-Feb-2020].

[34] About the European Patent Office: E. P. Office, "The EPO at a glance." [Online]. Available: https://www.epo.org/about-us/at-a-glance.html. [Accessed: 21-Feb-2020].

[35] About the World Intellectial Property Organization: World Intellectual Property Organization, "Inside WIPO," *wipo.int*, 2014. [Online]. Available: https://www.wipo.int/about-wipo/en/. [Accessed: 21-Feb-2020].

[40] About the Roomba iSeries: IRobot, "Roomba iSeries | iRobot." [Online]. Available: https://www.irobot.com/roomba/i-series?_ga=2.54449709.403465439.1581865484-2102031925.1581865484. [Accessed: 17-Feb-2020].

[47] XSENS IMUS: XSENS, "Inertial Sensor Modules." [Online]. Available: https://www.xsens.com/inertial-sensor-modules. [Accessed: 11-Feb-2020].

[50] Sensor basics: M. Rouse, "What is Active Directory? - Definition from WhatIs.com," *TechTarget*, 2011. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/active-sensor. [Accessed: 28-Jan-2020].

[52] Velodyne Lidar on wavelengths: I. Velodyne Lidar, "Guide to LiDAR Wavelengths - Velodyne LiDAR." [Online]. Available: https://velodynelidar.com/newsroom/guide-to-lidar-wavelengths/. [Accessed: 02-Feb-2020].

[53] Lidar technology: PicoQuant, "LIDAR/Ranging/SLR | PicoQuant." [Online]. Available: https://www.picoquant.com/applications/category/metrology/lidar-ranging-slr#description. [Accessed: 02-Feb-2020].

[65] Magnetic guidance: N. M. Corporation, "Magnetic Guide Sensors | Roboteq." [Online]. Available: https://www.roboteq.com/all-products/magnetic-guide-sensors. [Accessed: 04-May-2020].

[72] Course on pure pursuit controllers: J. K. Steven Waslander, "Lesson 2: Geometric Lateral Control - Pure Pursuit - Module 6: Vehicle Lateral Control." [Online]. Available: https://www.coursera.org/lecture/intro-self-driving-cars/lesson-2-geometric-lateral-control-pure-pursuit-44N7x. [Accessed: 25-May-2020].

[73] About the Open Robotics Foundation: Open Source Robotics Foundation, "About Open Robotics," *openrobotics.org*. [Online]. Available: https://www.openrobotics.org/. [Accessed: 04-Feb-2020].

[74] roscpp documentation: T. Quigley, Morgan; Faust, Josh; Gerkey, Brian; Straszheim, "roscpp - ROS Wiki," 2015. [Online]. Available: http://wiki.ros.org/roscpp. [Accessed: 23-Mar-2020].

[75] rospy documentation: Open Source Robotics Foundation, "rospy - ROS Wiki," 2017. [Online]. Available: http://wiki.ros.org/rospy. [Accessed: 23-Mar-2020].

[76] About ROS: Open Source Robotics Foundation, "ROS.org | About ROS," *ROS.org*, 2014. [Online]. Available: https://www.ros.org/about-ros/. [Accessed: 03-Feb-2020].

[78] actionlib documentation: V. P. Eitan Marder-Eppstein, "actionlib - ROS Wiki," *ROS.org*, 2011. [Online]. Available: http://wiki.ros.org/actionlib. [Accessed: 16-May-2020].

[79] Navigation stack documentation: E. Marder-Eppstein and M. Ferguson, "navigation - ROS Wiki," *ROS.org*, 2016. [Online]. Available: http://wiki.ros.org/navigation. [Accessed: 15-Apr-2020].

[80] gmapping documentation: Brian Gerkey, "gmapping - ROS Wiki," *ROS wiki*, 2018. [Online]. Available: http://wiki.ros.org/gmapping. [Accessed: 25-May-2020].

[82] rosbag documentation: Open Source Robotics Foundation, "ROS Wiki rosbag." [Online]. Available: http://wiki.ros.org/rosbag?distro=melodic. [Accessed: 23-Apr-2020].

[83] amcl documentation: E. Kang, "amcl - ROS Wiki," *ROS.org*, 2020. [Online]. Available: http://wiki.ros.org/amcl. [Accessed: 02-Apr-2020].

[84] Twist message documentation: Open Source Robotics Foundation, "geometry_msgs/Twist Documentation," 2018. [Online]. Available: https://docs.ros.org/api/geometry_msgs/html/msg/Twist.html. [Accessed: 26-May-2020].

[85] About Gazebo: I. Saito, "gazebo - ROS Wiki," *ROS.org*. [Online]. Available: http://wiki.ros.org/gazebo. [Accessed: 16-Apr-2020].

[86] RViz documentation: J. F. Dave Hershberger, David Gossow, "rviz - ROS Wiki," *ROS.org*, 2009. [Online]. Available: http://wiki.ros.org/rviz. [Accessed: 16-Apr-2020].

[87] About OpenCV: Itseez, "About - OpenCV library," *OpenCV*, 2000. [Online]. Available: https://opencv.org/about/. [Accessed: 16-Apr-2020].

[88] Canny Edge Detection: A. Mordvintsev and A. K, "Canny Edge Detection — OpenCV-Python Tutorials 1 documentation," *Sphinx*, 2013. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html . [Accessed: 04-May-2020].

[90] Hokuyo UTM-30LX-EW product details: Hokuyo, "Scanning Rangefinder Distance Data Output/UTM-30LX-EW Product Details | HOKUYO AUTOMATIC CO., LTD." [Online]. Available: https://www.hokuyo-aut.jp/search/single.php?serial=170. [Accessed: 23-Mar-2020].

[91] Google Pixel 3: GSM Arena, "Google Pixel XL - Full phone specifications," *GSM Arena*, 2016. [Online]. Available: https://www.gsmarena.com/google_pixel_3-9256.php. [Accessed: 22-Apr-2020].

[94] Hamming distance: N. Raut, "What is Hamming Distance?," *TutorialsPoint*, 2018. [Online]. Available: https://www.tutorialspoint.com/what-is-hamming-distance. [Accessed: 23-Apr-2020].

[95] Intel RealSense D415 product details: Intel, "Depth Camera D435 – Intel® RealSense™ Depth and Tracking Cameras," *Https://Www.Intelrealsense.Com/Depth-Camera-D435/*, 2019. [Online]. Available: https://www.intelrealsense.com/depth-camera-d415/. [Accessed: 28-Apr-2020].

[96] LaserScan message documentation: Open Source Robotics Foundation, "sensor_msgs/LaserScan Documentation," 2016. [Online]. Available: http://docs.ros.org/melodic/api/sensor_msgs/html/msg/LaserScan.html. [Accessed: 22-Mar-2020].

Appendix I: Range experiments

Laser range test

The following experiment was conducted to determine the range for the laser.

Experimental set-up

Figure 17-1 shows the experimental set-up for the range test. Reflective markers have been placed at various distances to determine the intensity measurements that can be expected by the Hokuyo Laser.

Figure 17-1: An image of the experimental set-up for a range experiment with the Hokuyo laser. The laser is directly exposed to sunlight.

Results

Table 0-1: Results from range determination test for the laser scanner. The results from scans at 8m are set in parentheses due to the amount of reflected points being very few compared to the other measurements.

Set distance ($\pm$ 0.5m)	Avgerage Intensity	Avgerage measured distance	False positives
2 m	14033	2.039 m	No
3 m	15748	3.011 m	No
4 m	15590	4.022 m	No
5 m	14824	5.015 m	No
6 m	15265	6.002 m	No
7 m	14103	6.997 m	No
(8 m)	(12854)	(8.98 m)	(No)
9 m	14568	8.98 m	No
10 m	14174	9.98 m	No

Camera range test

The following experiment was conducted to determine the range for the RGB-D camera.

Experimental setup

Apriltags with IDs 0-2 will be used to determine range. ID 1 and 2 will be used to determine from how far away the charger can be detected. ID 0 will be used as a range tag to determine the maximum and minimum range for detection of one tag. The experiment will be both be conducted with the camera directly exposed to sunlight and with the sun shining from behind.

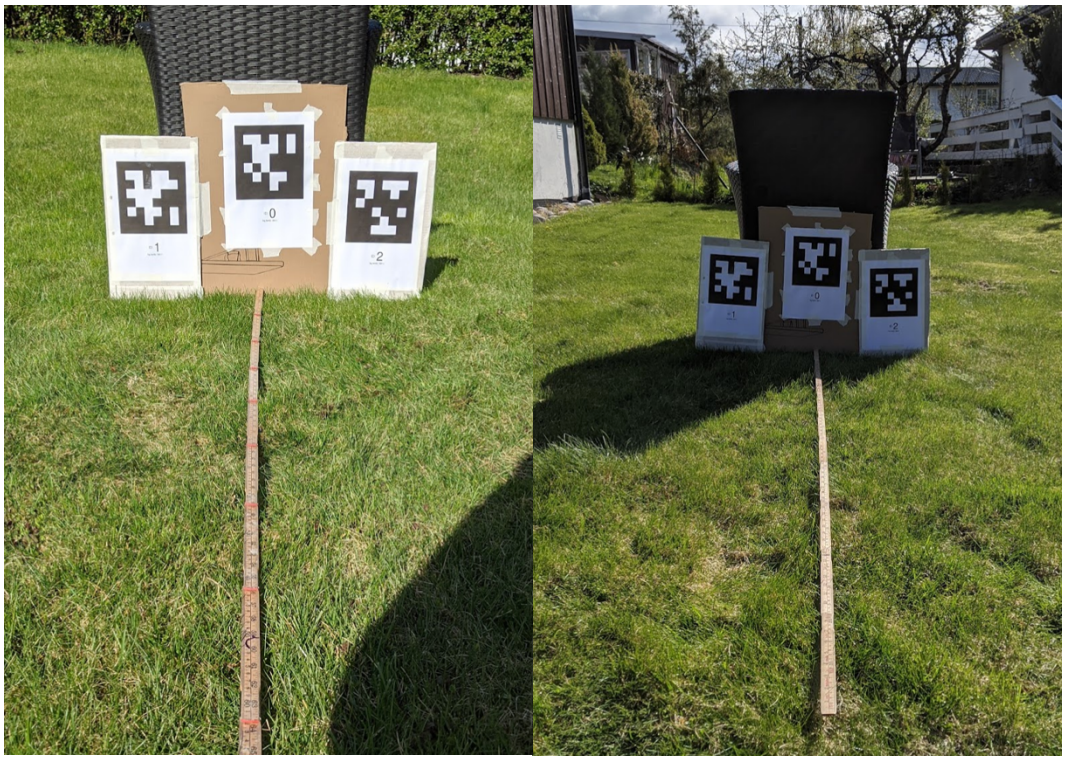


Figure 0-2: Images of the experimental set-ups to determine the range specifications for AprilTag detection. In the left image, the sun is located behind the camera, and in the right, the sun is in front.

Results

Table 0-2: An overview of detection data obtained from a range test with the camera. Ranges marked with parentheses are non-continuous range measurements.

Set distance ($\pm\ 0.5m$)	Sun exposure	Charger detected	Range tag distance
$min.$	No	Not detected	$0.32\ m$
	Yes	Not detected	$0.32\ m$
$2\ m$	No	Continuously	$2.03\ m$
	Yes	Continuously	$2.01\ m$

Table 0-3: Table 0-2 continues.

Set distance ($\pm$ 0.5m)	Sun exposure	Charger detected	Range tag distance
3 m	No	Continuously	3.08 m
	Yes	Continuously	3.01 m
4 m	No	Continuously	4.01 m
	Yes	Continuously	3.99 m
5 m	No	Continuously	5.10 m
	Yes	Continuously	4.99 m
6 m	No	Often	6.03 m
	Yes	Often	5.98 m
7 m	No	Sometimes	7.08 m
	Yes	Sometimes	(6.95 m)
8 m	No	Very rare	(8.08 m)
	Yes	Very rare	(8.01 m)
9 m	No	Not detected	Not detected
	Yes	Not detected	Not detected
10 m	No	Not detected	Not detected
	Yes	Not detected	Not detected

- Minimum distance of detection with tags 1 and 2 placed 2.4 meters apart: 2.5 m

Appendix II: Simulink mode

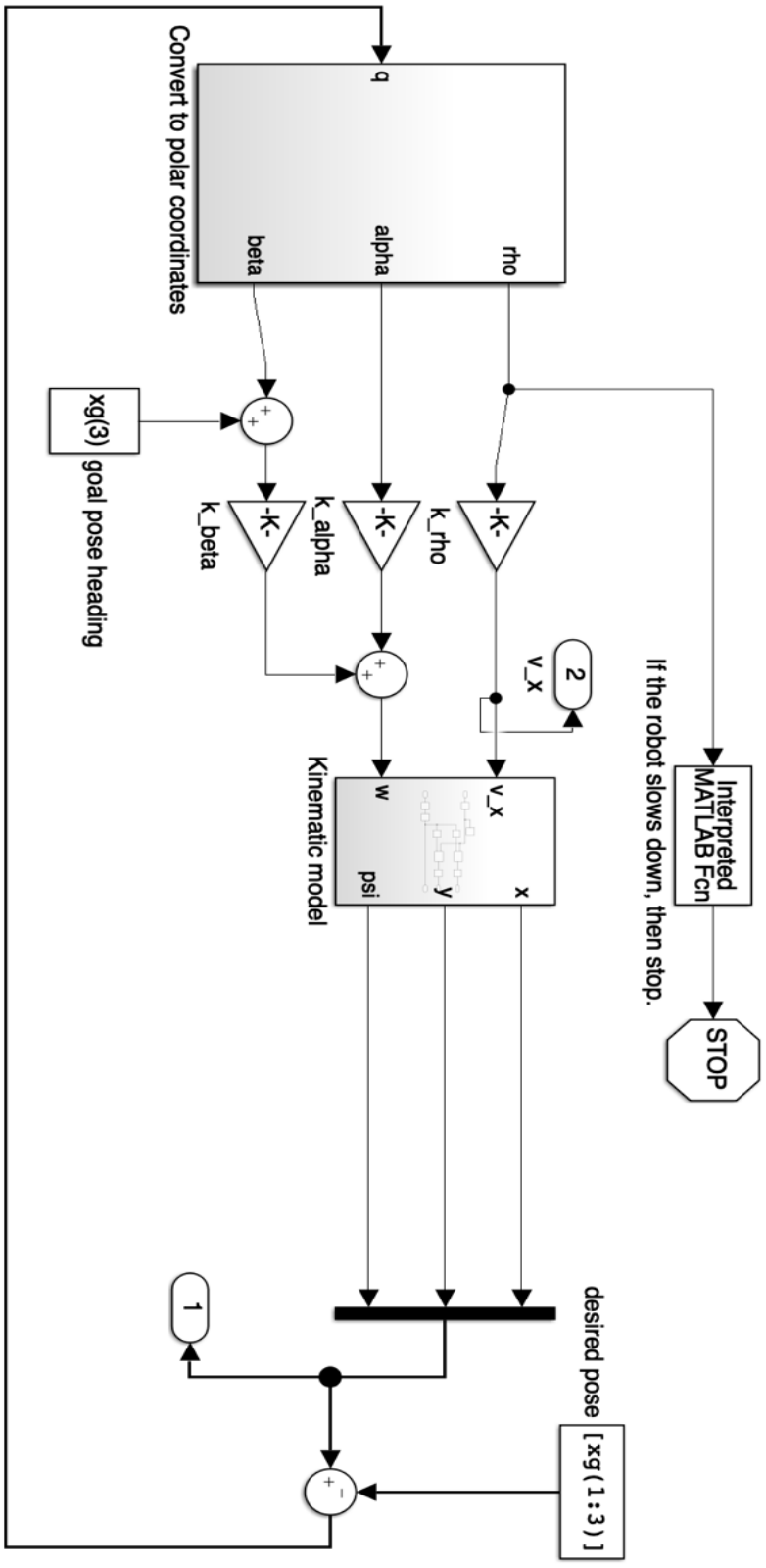


Figure 0-1: An illustration of the Simulink model that is used to analyze the system response of the motion controller.

Appendix III: Patents

A brief patent search has been conducted to map the technologies that cannot be utilized to make the autonomous docking function for Thorvald. A more comprehensive search should be done if the developed function is intended to be commercialized.

Key takeaways

- iRobot has multiple patents on their technologies, including the docking of their commercial robot, Roomba [40].
- Locus Robotics Corp. has international patents for localization of a charger, and a controller for docking of a wheeled robot.
- No other patents on autonomous docking were found during this search.

Search words

Autonomous docking, Robot docking, Autonomous charging, Robot charging station, Robot charging, Charging mobile robot, Automatic charging, recharge robot,

Appendix IV: External survey

A survey on detection concepts for a mobile robot

In the development of an autonomous docking function for the agricultural robot platform, Thorvald, I am investigating sensor technology that can enable Thorvald to recognize a charging dock. The charging dock will be located inside an assumed static environment.
As an intuitive approach to the problem, Thorvald can be considered as a car that is supposed to park within the boundaries of a given parking space. Thorvald needs to recognize the space, navigate to it and align inside the marked area. Imagine that the parking space width is marked by two poles. Thorvald can use these two poles to generate a path that it can follow to safely park. The question is; How can Thorvald distinguish these poles from the surrounding environment so that a path can be generated?
For now, the scope is narrowed down to two concepts that consider two different technologies for recognizing the poles, one that considers the use of a 2D laser scanner and one that considers a camera. Brief descriptions of the concepts are provided below alongside some questions.
The main purpose of this survey is to obtain feedback from people who have knowledge and expertise with robots and sensor technologies. The feedback will be used to help select a final concept for the autonomous docking function.
* Required

1. Are you experienced with any of the following technologies? *

Mark only one oval per row.

	Not experienced	Somewhat experienced	Experienced	Highly experienced
Laser scanners	◯	◯	◯	◯
Cameras	◯	◯	◯	◯
Computer vision	◯	◯	◯	◯
Mobile robots	◯	◯	◯	◯

2. How many years of experience do you have with the technology that you consider yourself most experienced with?

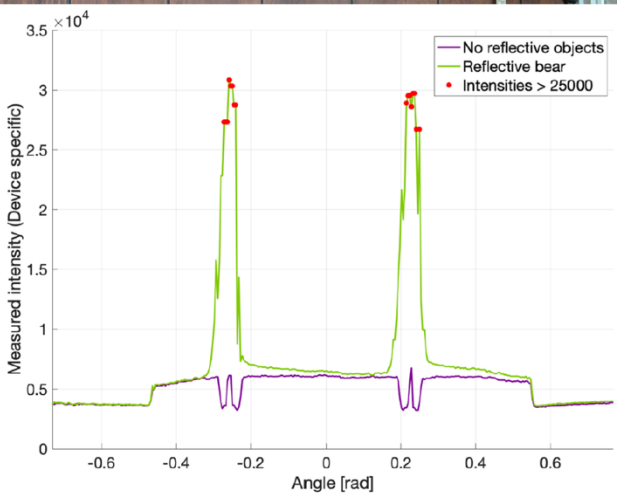
Mark only one oval.

- ◯ < 1 year
- ◯ 1 - 2 years
- ◯ 2 - 4 years
- ◯ > 4 years

Concept 1: Lidar and reflective objects

The first concept considers using a 2D laser scanner to distinguish two points based on intensity measurements. Reflective objects with prisms have proven, through small-scale tests, to give significantly higher intensity measurements compared to other objects. In addition to intensity, laser scans provide both ranges and angles, which makes it possible to map detected objects with respect to the laser frame.

The experimental setup for one of the small-scale tests that have been conducted is shown below alongside a graph that shows the intensity measurements from a scan.
Laser scanner: Hokuyo UTM-30LX-EW
Reflective objects: Reflective figures with prisms



3. Do you think that a 2D scanner is suitable for detection of the dock? *

Mark only one oval.

- ◯ Suitable
- ◯ Somewhat suitable
- ◯ Somewhat unsuitable
- ◯ Not suitable
- ◯ I do not know

4. Do you think that using a 2D scanner to detect reflective objects will provide robust measurements in dynamic environments? *

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Maybe

5. Maintenance of laser scanner:

Mark only one oval per row.

	Very often	Often	Now and then	Rare	Very rare
Calibration	◯	◯	◯	◯	◯
Cleaning	◯	◯	◯	◯	◯
Service / Inspection	◯	◯	◯	◯	◯

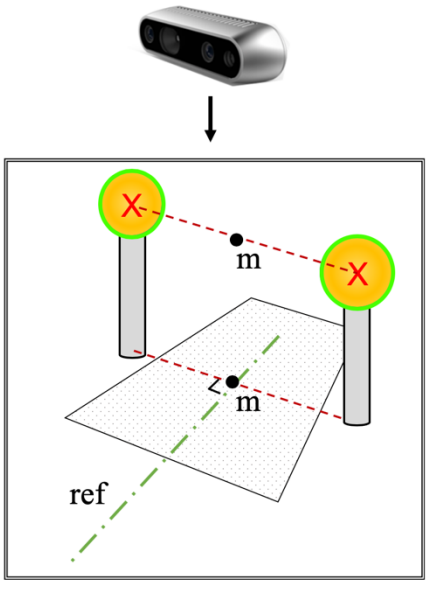
6. Do you have any major concerns regarding the laser concept?

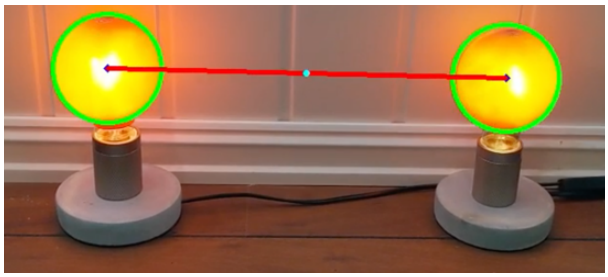
7. Other comments:

Concept 2: Camera and visible landmarks

The second concept considers using an RGB-D camera to detect two landmarks, either shapes or other distinguishable features. RGB-D cameras differ from standard RGB cameras in that they, in addition to the pixel color values, provide depth measurements. This allows for the creation of a 3D point cloud, and hence opportunities to obtain information about the position of objects with respect to the camera frame.

An illustration of the concept is shown below where two pillars mark the boundaries and the middle position (m) of the charging station. A simple reference path is generated as the orthogonal from the baseline between the pillars at m.
The idea is to use the camera to detect patterns or shapes with distinguishable features (geometries, QR-codes or April-tags), and extract the position of the two pillars to be able to define a final pose for the robot.
Underneath the illustration is an image showing results obtained from a small-scale test conducted with the camera of a cellphone.





8. Do you think that a camera is suitable for detection of the dock? *

Mark only one oval.

- ◯ Suitable
- ◯ Somewhat suitable
- ◯ Somewhat unsuitable
- ◯ Not suitable
- ◯ I do not know

9. Do you think that using a camera can provide sufficient data to enable Thorvald to obtain information about the position of the dock? *

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Maybe

10. Maintenance of camera:

Mark only one oval per row.

	Very often	Often	Now and then	Rare	Very rare
Calibration	◯	◯	◯	◯	◯
Cleaning	◯	◯	◯	◯	◯
Service / Inspection	◯	◯	◯	◯	◯

11. Do you have any major concerns regarding the camera concept?

12. Other comments about the camera concept:

13. If you were to choose between using a laser scanner or a camera for the considered task, which one would you choose? *

Mark only one oval.

◯ Laser scanner
◯ Camera

14. Any final comments or advice for the development of the docking function?

Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway