NTNU
Norwegian University of
Science and Technology

Autonomous Path-Planning and -Following for a Marine Surface Robot

Mats Håkon Follestad

Marine Technology
Submission date: June 2017
Supervisor: Roger Skjetne, IMT

Norwegian University of Science and Technology
Department of Marine Technology

MSC THESIS DESCRIPTION SHEET

Name of the candidate: Follestad, Mats Håkon

Field of study: Marine control engineering

Thesis title (Norwegian): Autonom baneplanlegging og -følging for ein marin overflaterobot

Thesis title (English): Autonomous path-planning and -following for a marine surface robot

Background
A highly maneuverable and robust multi-purpose marine model platform, called the "C/S Saucer", has been developed for laboratory experiments in the NTNU Marine Cybernetics Laboratory (MC-Lab). The intended use of this multi-purpose robotic vehicle is for students to design, implement, and test a variety of nonlinear guidance, control, and estimation algorithms for specified experimental case studies.

The objective of this project is to study autonomous path-planning with obstacle avoidance, using a global map combined with local terrain mapping and obstacle avoidance, based on relevant sensors. The problem scenario is for the robot to move from present location to a destination in a global map. A global path to follow is then constructed. While moving along the path, simultaneous localization and mapping (SLAM) is used by the robot to map and understand the local terrain, to localize itself within it, and to make corrections to the path if necessary (Ueland, 2016). Lastly, reactive anti-collision maneuvers should be considered upon detection of suddenly appearing static or dynamic obstacles, like other vessels. The purpose is to understand how information from a global map can be fused with a SLAM technique and reactive obstacle detection to ensure autonomous path-planning and path-following.

Work description
1. Perform a background and literature review to provide information and relevant references on:
   - Path-generation in regional and local maps.
   - SLAM based on Lidar or other range-based sensors, for terrain mapping and relative positioning.
   - Proximity sensors for local obstacle detection.
   - Methods for reactive obstacle avoidance.
   - Autonomy layers of an autonomous system.

   Write a list with abbreviations and definitions of terms, explaining relevant concepts related to the literature study and project assignment.

2. Create a system for an unmanned surface vessel that combines a global and a local path planning method: This shall:
   - enable the vessel to plan a route between points, taking static obstacles into account, and
   - react to dynamic obstacles and replan route locally.

   The global path planner shall plan the path around static obstacles, based on the information of an a-priori global map.

3. Using relevant sensors, the system should be able to perform simultaneous localization and mapping (SLAM), which in combination with proximity sensors enables the system to map and better perceive the local terrain. Explain how this increases the autonomy of the vessel.
   - The local path planner will use this information to deploy reactive anti-collision maneuvers to the path if necessary.

4. Use the highly maneuverable and robust multi-purpose marine model platform, called the C/S Saucer, to test the developed systems in the NTNU Marine Cybernetics Laboratory (MC-Lab).

5. Modify the already existing simulator for the C/S Saucer developed in (Ueland, 2016) such that it fits the problem in this thesis and can be used for verification of your algorithms and methods.

6. Include the MC-Lab Qualisys system in the experiments, such that the vessel position provided by the qualisys system can be fused with the localization estimates from the SLAM algorithm. This will make the position estimates of the C/S Saucer more robust when operating in a dynamic environment.

Specifications
The scope of work may prove to be larger than initially anticipated. By the approval from the supervisor, described topics may be deleted or reduced in extent without consequences with regard to grading.

The candidate shall present personal contribution to the resolution of problems within the scope of work. Theories and conclusions should be based on mathematical derivations and logic reasoning identifying the various steps in the deduction.

The report shall be organized in a logical structure to give a clear exposition of background, results, assessments, and conclusions. The text should be brief and to the point, with a clear language. Rigorous mathematical deductions and illustrating figures are preferred over lengthy textual descriptions. The report shall have font size 11 pts. It shall be written in English (preferably US) and contain the following elements: Title page, abstract, acknowledgements, thesis specification, list of symbols and acronyms, table of contents, introduction with objective, background, and scope and delimitations, main body with problem formulations, derivations/developments and results, conclusions with recommendations for further work, references, and optional appendices. All figures, tables, and equations shall be numerated. The original contribution of the candidate and material taken from other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a Harvard citation style (e.g. *natbib* Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct. Such practice is taken very seriously by the university and will have consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed upon.

The thesis shall be submitted with a printed and electronic copy to the main supervisor, with the printed copy signed by the candidate. The final revised version of this thesis description must be included. The report must be submitted according to NTNU procedures. Computer code, pictures, videos, data series, and a PDF version of the report shall be included electronically with all submitted versions.

Start date: 15 January, 2017 **Due date:** As specified by the administration.

Supervisor: Roger Skjetne
Co-advisor(s): Dong T. Nguyen (thesis report), Einar S. Ueland (C/S Saucer and SLAM), Petter Norgren (SLAM)

Trondheim,

_____
Roger Skjetne
Supervisor

Abstract

Autonomous navigation for unmanned surface vessels is a modern and rapidly growing field of research. In recent years, many concepts and solutions have been proposed in order to achieve autonomous control of marine vessels. This thesis reviews the development of an autonomous path-planning and -following system with obstacle avoidance to avoid collisions, for a model-scale surface vessel. The system was developed to be able to navigate in a global or locally known map of static obstacles, where there were uncertainties regrading unknown static and dynamic obstacle present in the area of operation.

A LIDAR was used as a proximity sensor, in order for the system to sense the environment and to map the dynamic obstacles in the vicinity of the vessel, and in addition perform simultaneously localization and mapping. This information was used by the collision avoidance system in order to avoid obstacles.

The autonomous path planner used in this thesis is a slightly modified version of the A* algorithm, that generates a path towards a set goal. In order to follow the path, a Line of Sight steering law has been implemented, that has been modified for use on an omnidirectional vessel. A modified version of the Dynamic Window algorithm has been developed and implemented, to enable the vessel to avoid dynamic obstacles that is met along the path. The system was designed for use on the omnidirectional vessel named the CS Saucer, which was used as the platform to test the resulting system in the NTNU Marine Cybernetics Laboratory.

The experiments and computer simulations showed that the collision avoidance system could handle static obstacles very well. The collision avoidance system was also shown to handle dynamic obstacles, but there were some limitations to the system in this regards. This was mainly due to the dynamic obstacles' velocity not being estimated and compensated for. This should be the main focus of further work, as well as using the obstacle velocity information to make the collision avoidance system abide the International Regulations for Avoiding Collision at Sea (COLREGS).

Sammendrag

Autonom navigering for ubemannede overflate fartøy er ett moderne forskningsfelt i rask utvikling. I senere år har det stadig kommet nye forslag til løsninger og konsepter for å oppnå autonom kontroll av marine fartøy. Denne avhandlingen tar for seg utviklingen av et autonomt navigeringsystem som kombinerer autonom baneplanelegging og autonom banefølging, med ett system for å unngå kollisjoner. Dette systemet skulle implementeres og testes for ett modell-skala fartøy. Systemet ble utviklet for å navigere i et globalt eller lokalt kjent kart av statiske objekter, hvor det var knyttet stor usikkerhet anngående ukjente statiske og dynamiske objekter innenfor operasjons området.

For at fartøyet skal klare å oppdage statiske hindringer og dynamiske objekter brukes en LIDAR som en nærhetssensor for å lage ett kart av dynamiske objekter og for å utføre "samtidig lokalisering og kartlegging" (SLAM). Denne informasjonen ble brukt for å unngå kollisjoner.

Den autonome baneplaneleggeren som er brukt i denne avhandlingen er en A* algoritme, som generer en bane mot ett satt mål. For å følge banen, er det blitt implementert en siktlinje (LOS) styringslov som er blitt designet for bruk på et rundt fartøy. En modifisert version av algoritmen "Dynamic Window" er blitt implementert. Dette ble gjort for å gjøre fartøyet i stand til å unngå kollisjoner med dynamiske objekter den møter langs den generete banen. Systemet ble designet for bruk på det runde modell-skala fartøyet CS Saucer, som ble brukt til å teste det utviklede navigasjons systemet i NTNUs Marin Kybernetiske Laboratorium.

Modellforsøk og datasimuleringer viste at kollisjon unnvikelses systemet fungerte godt for statiske objekter. Systemet viste også at det var i stand til å takle dynamiske objekter, men at det også hadde noen begrensninger i disse tilfellene. Dette var hovedsakelig på grunn av at farten på de dynamiske hindringene ikke blir estimert og tatt høyde for. Dette bør være hovedfokuset for videre arbeid, samt å bruke denne informasjonen til å gjøre at kollisjonsunngåelsen følger sjøveisreglene (COLREGS).

Acknowledgement

I would like to thank my supervisor Professor Roger Skjetne for his guidance during the work of this thesis.

I would like to thank my co-advisors, Einar S. Ueland for helping me when there was trouble with the CS-Saucer, and Dong T. Nguyen for proof reading the thesis report.

I would also like to give a special thanks to my brother, Sindre G. Follestad for helping me proof reading and finalizing my thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

2D Two Dimensions

3D Three Dimensions

AIS Automatic Identification System

AUV Autonomous Underwater Vehicle

COLREGS International Regulations for Avoiding Collisions at Sea

CS Cyber Ship

DARPA Defense Advanced Research Projects Agency

DOF Degrees of freedom

DW Dynamic Window

IMO International Maritime Organization

IMU Inertial Measurement Unit

LIDAR Light Imaging Detection And Ranging

LiPo Lithium polymer battery

LOS Line of sight

LQG Linear Quadratic Gaussian

LQR Linear Quadratic Regulator

MC-lab The NTNU Marine Cybernetics Laboratory

NaN Not a Number

ONR Office of Naval Research

POA Projected Obstacle Area

PWM Pulse Width Modulated

QTM Qualisys Track Manager

Radar Radio Detection And Ranging

RGB Red-Green-Blue

RHIB Rigid-hulled inflatable boat

ROS Robot Operating System

RPi2 Raspberry Pi 2

RRT Rapidly-exploring Random Tree

SLAM Simultaneous Localization and Mapping

Sonar Sound Navigation And Ranging

USB Universal Serial Bus

USV Unmanned Surface Vessel

VFF Virtual Force Field

VFH Virual Field Histogram

VHF Very High Frequency

VO Velocity Obstacle

Nomenclature

α_k The tangential angle of the path

a_x^i The x-position of the vessel on a circular trajectory at interval i

a_y^i The y-position of the vessel on a circular trajectory at interval i

C_A Hydrodynamic Coriolis matrix

$\chi_d(e)$ The desired course angle

χ_p the angle of the path in the Basin-relative reference frame

$\overline{\Delta\eta}_o$ The resulting mean offset between two reference frames after rotation

$\chi_r(e)$ The velocity-path relative angle

$C_{RB}(v)$ Coriolis Rigid body matrix

D Linear damping matrix

$\overline{\Delta\psi}$ The mean heading difference between the two reference frames

$D(v_r)$ Nonlinear damping matrix

$e(t)$ Cross-track error

f Individual thrust force vector

$f(n)$ Estimate on total cost to reach goal via node n

$g(n)$ Cost of path from start node to node n

$g(\eta)$ Vector of gravitational/buoyancy forces and moments

$goal_x$ x-coordinate of the goal node

$goal_y$ y-coordinate of the goal node

$h(n)$ Heuristic value. An estimate of the cheapest path to the goal from node n

Iz Moment of inertia about the z-axis

K_d Derivative controller gains

K_p Proportional control gains

m Total mass of the system

$M(S_i(\xi))$ Gives the map value at position $S_i(\xi)$

M_{RB} Rigid body mass and inertia matrix

n Node in graph

n_x x-coordinate of a given node in the graph

n_y y-coordinate of a given node in the graph

N_r Linear damping in yaw

$N_{r|r|}$ Nonlinear damping yaw

$N_{\dot{r}}$ Added mass in yaw

ω_c Cut-off Frequency

$\hat{\psi}$ Estimated Heading

$R(\psi)$ Rotation matrix

$\hat{r}$ Estimated rate of turn in yaw

$S_i(\xi)$ The world coordinates of the laser scan endpoints

$s(t)$ along-track distance

T Thrust allocation matrix

$\tau_{external}$ External forces acting on the vessel

τ Generalized force vector

V_a The set of admissible velocities

V_d The velocities obtainable within the Dynamic Window

v_r Linear velocity relative to local current

V_r The resulting search space in the Dynamic Widow Algorithm

V_s The set of possible velocities obtainable for a vessel

$\hat{v}_{xy}$ Estimated velocity along the x- and y-axis

V_{xyd} Desired velocity along the x- and y-axis

ξ Vessel position vector

$x(t)$ x position of vessel at time t

X_u Linear damping in surge

$X_{u|u|}$ Nonlinear damping in surge

$X_{\dot{u}}$ Added mass in surge

$y(t)$ y position of the vessel at time t

Y_v Linear damping in sway

$Y_{v|v|}$ Nonlinear damping sway

$Y_{\dot{v}}$ Added mass in sway

Chapter 1

Introduction

1.1 Motivation

The world is becoming increasingly reliant on automated systems to preform complex tasks. As these tasks grow more complex, the systems' ability to make intelligent decisions must also increase. The increase in the ability to perform intelligent decisions shifts a system from an automated system to an autonomous system. An automated system is a system that is specialized to perform a well defined task, where the process to reach the goal is clearly defined. In comparison an autonomous system is a self governing system capable of performing tasks with loosely based goals, by using intelligent reasoning. The study of autonomous vehicles is a research field that is rapidly growing. The environment an autonomous vehicle operates in is strongly dependent on the vehicle type, ranging from the depths of the ocean to the vastness of space. One common problem for autonomous vehicle operation is that the area of operation often has a high degree of uncertainty regarding the dynamic and static obstacle present in the environment.

This is also the case for the environment an Unmanned Surface Vehicles USV is expected to operate in. A USV must be able to operate in such a way that it avoids collision, while it simultaneously works towards reaching the set goal. In order to ensure the safety of the USV, and of the environment surrounding it, the USV must be able avoid the obstacles present in the environment. The robot must be able to have an understanding of its situation, and use this situational awareness to deduce feasible and safe decisions regarding the mission objective. The USV considered in this thesis is the CS Saucer, which is an omnidirectional model-scale vessel actuated by three rotating azimuth thrusters.

Previous work done on the CS Saucer can be found in, (Sharoni, 2016), (Ueland, 2016) and (Spange, 2016). In the work of (Ueland, 2016) and (Spange, 2016) the focus of the work is shifted towards autonomy. In (Ueland, 2016) a LIDAR was installed on the CS Saucer, and a system for autonomous exploration of a static small-scale marine environment was developed. The resulting system was tested and verified by experiments preformed in a controlled basin facility. (Spange, 2016) reviewed the development of an autonomous docking feature for the CS Saucer in a sheltered lab environment. The next step in the sense of the autonomy of the vessel would be to make the system able to handle dynamic changes in the environment by applying local collision avoidance maneuvers. Collision avoidance is an integral part

of making a vehicle autonomous, and a local scheme for this will help the vehicle when encountering dynamic changes in the environment.

1.2 Background

1.2.1 Path planning in global and regional maps

Path planning in global and regional maps, where one plans the path around a map of presumed known static obstacles, is a problem with many solutions. The global approach can define a path, from start to goal using the information granted by the regional and global maps that defines static obstacles in the area of operation. One could also utilize a map that is continuously built based on the current sensor measurements. The downside of the global approach is that the global methods often must process much information, which affects the time it takes to generate a path. The time this requires might vary between less than a second, or even up to several minutes. From this one can say that the "reaction time" of a global method might potentially be slow, and may not be suited for a rapidly changing environment. However if one introduces dynamic obstacle avoidance as a local correction to the path, algorithms with much faster "reaction time" can be utilized to handle a local world model where the inaccuracies and dynamics which are not represented in the global map are detected and included in the local planning.

1.2.2 Reactive obstacle avoidance

In (Tan et al., 2004a) it is shown that the architecture of collision avoidance systems can be divided into three major categories, namely as listed below.

- The Deliberative Architecture
- The Reactive Architecture
- The Hybrid Architecture

The deliberative architecture is the same as a global method described in chapter 1.2.1, and the reactive architecture is what was mentioned as a local path planning method method in chapter 1.2.1. The reactive architecture is also referred to as a sense-act method. In this architecture the sensor measurements might be used directly, instead of needing a map of the environment. This leads to a lower computational cost, which leads to a better response time. The downside of the reactive architecture, is that it might lead to non-optimal paths, and even trap the vessel in a local minima if used alone (Borenstein and Koren, 1999). To create a more optimal architecture, the deliberative and reactive parts are combined to a hybrid architecture. By doing this one can get the best of both worlds, where the deliberative layer guides the vessel towards the goal while the reactive layer compensates for most of the unforeseen and dynamic changes to the local environment.

1.2.3 Proximity sensors

A proximity sensor is a sensor that is capable of detecting the presence of objects in its vicinity without the need of any physical contact with the objects. There is a wide variety of different types of sensors capable of accomplishing this. In this thesis

the Light Detection And Ranging device (LIDAR) that was implemented in (Ueland, 2016) was used as a proximity sensor. A LIDAR is a remote sensing device that measures distance to nearby objects by illuminating its environment with a laser and analyzing the reflected light. The reflected laser signal is sampled by vision acquisition, and the time it took for the signal to return reveals the distance to the target. Figure 1.1 illustrates how the 2 Dimensional (2D) LIDAR installed on the CS saucer emits a laser pulse that is reflected by a wall and then sampled by vision acquisition. The LIDAR in figure 1.1 rotates around its own axis, and will in that way sense the environment in a 2D plane.

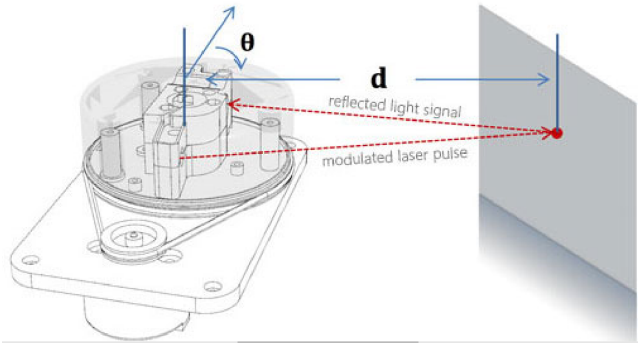


Figure 1.1: 2D LIDAR, illustrated with the RPLIDAR, courtesy Roboshop

1.2.4 SLAM for terrain mapping and relative position

The Simultaneous Localization and Mapping (SLAM) problem is a process where a mobile robot generates a map using measured distance data while simultaneously localizing itself in it, without the need of any a previous knowledge of the location. This is an immensely powerful tool, SLAM makes it possible for a robot to use terrain aided navigation in an unknown environment. From this one can also add that robots that utilizes SLAM could also be used for terrain mapping, for example for seabed mapping, underground mine exploration, mapping of dangerous areas and even space exploration.

The robot needs sensors in order to sense the environment, and in that way be able to gather data that could be used for mapping the terrain, and determining its own position. Different environments require different sensors that are suitable for the environment of the area of operation. Here the most significant sensors currently in use for performing SLAM will be presented. From (Chong et al., 2015) it said that the most common sensors for preforming SLAM are of the type:

- Laser Range Finders
- Acoustic Sensors
- Stereo Vision Sensors
- RGB-D Sensors

SLAM based on LIDAR A LIDAR enables fast data acquisition with a high level of accuracy. The data cloud representing the distances to objects in the vicinity

of the vehicle is used to both map the surrounding area, and simultaneously estimate the position of the vehicle. An example of a 2D LIDAR being used in a multi-SLAM framework can be found in (Koch et al., 2016). In (Koch et al., 2016), two robots were equipped with a 2D LIDAR, and a map is built in parallel by data gathered from both of the robots. The system was deployed in the Robocup Rescue 2015 environment, and shows that it has an advantage over open source algorithms such as Hector-SLAM when building maps of a larger scale.
In (Amzajerdian et al., 2011) the possibility of using a 3D LIDAR for mapping the elevation on planetary bodies during the landing phase of a space craft is presented. This information would then be used for navigation and to detect the most suitable and safe landing site, which would make the landing less hazardous.

Acoustic sensors For underwater vehicles sonars are most often used since laser and visual sensors struggle under water. The sonar emits sound waves, in order to then analyze the reflected sound waves, and in that way determine the distances to surrounding obstacles. For mobile robots, ultrasonic sensors are generally the cheapest available source of spatial sensing. However ultrasonic sensors have low spatial resolution and sensing range and in addition they are easily disturbed by the environment, and is dependent on the acoustic reflectivity of the measuered surface. In (Fallon et al., 2013) a system for reacquiring features of interest in a shallow water ocean environment for AUVs is described. Using a sonar that generates forward looking sonar images, a SLAM algorithm that detects and tracks features in the images was implemented, so that one could re-navigate to a previously mapped target. In (Jung et al., 2009) a wheel based autonomous vehicle was used for SLAM mapping an indoor environment utilizing a ultrasonic sensor.

SLAM based Stereo Vision Sensors Stereo vision sensors can be used for 3D mapping and finding the robot pose in the environment by utilizing stereo cameras and monocular cameras as can be seen in (Lemaire et al., 2007). Information such as features or distances can be obtained by analyzing multiple images captured by cameras (Chong et al., 2015). Stereo vision systems use two or more 2D images, taken from different positions, to construct 3D information (Mustafah et al., 2012). A stereo camera gains information about the distance from the disparity in textured areas of the image. Monocular cameras are used to gain depth information, by repeatedly observing features to get the feature's parallax. In (Gil et al., 2010) a system that utilized feature based SLAM, as the robot was required to observe visual landmarks as hold-points in the map was presented. If the landmark could not not be detected from some viewpoints, it would lead to failure, as an area that was already explored with such a landmark could end up not being verified as explored.

RGB-D sensors A RGB-D depth sensor projects a structured infrared spectrum light, which is then perceived by an infrared camera, and analyzed to gain depth information about the environment(Chong et al., 2015). This gives an RGB image with per pixel depth information (Henry et al., 2014). In general, structured light sensors are not usable under direct sunlight, because they are sensitive to external illumination. In (Endres et al., 2014) a 3-D slam system for RGB-D sensors, such as the Microsoft Kinect is presented. The presented approach utilizes visual key-points from the color images, and uses the depth images to localize them in 3D. A method

for utilizing the RGB-D method for full 3D mapping is also presented in (Henry et al., 2014) .

1.2.5 Autonomy and Unmanned Surface Vessels

Autonomy in engineering can be defined as an engineering system's ability to make decisions in order to complete loosely defined tasks of varying complexity, without the need of a human in the loop. The level of autonomy achieved by a robot is based on the amount of human interaction needed in order for the robot to be functional to be operational, and how much human intervention or guidance is required in the robot's decision making. Taxonomies and layers of an autonomous system will be reviewed in chapter 2.3.

A concept of an autonomous USV named ReVolt is presented in (Adams, 2014). The vessel is envisioned to be electrically powered with a cruising speed of 6 knots. Due to ReVolt not needing a crew, the vessel will be very cost effective as well as increasing the load capacity since there is no need for crew facilities. As of now, a 1:20 scale model concept ship is being tested, and is intended serve as inspiration for equipment makers and ship yards. The concept ship can be seen in figure 1.2a.

(a) A 1:20 scale model of the envisioned ReVolt, cortesy DNV GL

(b) A figure illustrating the prototype ASV the Sea Hunter, courtesy of DARPA

Figure 1.2: Pictures of the 1:20 scale model ReVlot, and the Sea Hunter

In the article (Vincent, 2016) a prototype of an autonomous warship is presented. The ship is called "Sea Hunter" and is intended to robustly track quiet diesel electric submarines (Littlefield). The vessel is intended to be unmanned, and capable of avoiding collision on its own accord. To do this a radar in combination with Automatic Identification System (AIS) is used to build up the situational awareness of the vessel. In addition in (Vincent, 2016) it is said that the collision avoidance maneuvers must be executed in a way that is recognized as human like. This implies that the collision avoidance scheme will be COLREGS compliant. In (Prigg, December 2016) it is stated that the Sea Hunter has completed some initial trials, and testing of the Sea Hunter's autonomy system is scheduled to continue through fall 2017 as a two-year test program jointly funded by DARPA and the Office of Naval Research (ONR). The Sea Hunter can be seen in figure 1.2b.

1.3 Previous Work Preformed on the CS Saucer

1.3.1 Marine Cybernetics Vessel CS Saucer: Design, construction and control

In (Idland, 2015) the CS Saucer was designed and constructed, and a dynamic positioning and a tentative path following system was developed for the CS Saucer and tested in the NTNU Marine Cybernetic Laboratory (MC-lab). The embedded controller myRIO from National Instruments is used to control the vessel.

1.3.2 Marine Inverted Pendulum

In (Sharoni, 2016) the CS Saucer was used to investigate if the CS Saucer was able to balance an inverted pendulum. The dynamic coupled equations for the marine inverted pendulum. Using this mathematical model, a simplified linearized version was used to design a Linear Quadratic Regulator (LQR) feedback controller. A Lundenberg state estimator was designed, and in combination with the LQR solves the linear quadratic Gaussian (LQG) problem. The resulting observer was able to reconstruct the unmeasured states that were necessary for feedback control. The resulting system was tested in the NTNU Marine Cybernetics Laboratory (MC-lab). The experiments preformed in the MC-lab showed that the objective was not completely achieved, but it is believed that it is possible to achieve it given some more work. One of the main concerns in this regard was the thruster mapping. The thrusters were unreliable, especially in the zero thrust region, and when thrust direction was changed.

1.3.3 Marine Autonomous Exploration using a LIDAR

In (Ueland, 2016), a complete design of a control system for a model-scale marine surface vessel capable of autonomous exploration in small-scale marine environments is created. The system implemented is an autonomous system that merges exploration strategies, path planner, Simultaneous Localization and Mapping (SLAM) algorithms, motion controller and a strategy for generating controller set points. The CS saucer was extensively upgraded with new hardware and the existing software control system was replaced with a Robot Operating System (ROS) platform. The vessel was equipped with a 2D Laser Range Scanner (LIDAR). The SLAM problem was solved by implementing an existing open source package applies the data gathered from the LIDAR to preform SLAM. In (Ueland, 2015) the CS Saucer was modeled to find an adequate equation of motion. The model was used to create the vessel simulator, developed in (Ueland, 2016) as well as the observer that was implemented in the control system. The resulting system was demonstrated through both simulations and experiments preformed in the MC-lab.

1.3.4 Autonomous Docking for Marine Vessels Using a LIDAR and Proximity Sensors.

In (Spange, 2016) a system for autonomous docking for a marine vessel was developed. The CS saucer was modified with physical modules that increased the yaw resistance of the CS Saucer, which altered the behaviour of the vessel in sway and yaw. The thrust allocation was also given new configurations and orientations. All this was done so that the CS saucer would emulate a leisure boat. Proximity sensors

were also added, to aid the autonomous exploration, by detecting obstacles outside the LIDAR scan plane. Simulations and experiments preformed in the MC-Lab were done in order to verify the system. Through the experimental results it was concluded that the trials that went without any incidents could be viewed as a proof of concept.

1.4 Objectives

The overall objective of this thesis was to develop a system capable of autonomous path-planning and following in a dynamic environment, for the omnidirectional marine surface robot CS Saucer. To successfully achieve this main objective, several partial goals has been set. These are defined as follows:

- Background literature review on Path-generation in regional and local maps, methods for reactive obstacle avoidance, autonomy layers for an autonomous system, SLAM based on range-based sensors and proximity sensors for local obstacle detection.
- Create a system for an unmanned surface vessel that combines a global and a local path planning method.
- Make use of the open source Hector-SLAM algorithm implemented for the system in (Ueland, 2016), and use this in combination with information gathered by relevant proximity sensors to map and better perceive the local terrain.
- Include the MC-lab Qualisys system to the experiments, such that the position measurement provided by the Qualisys system can be fused by the localization estimate from the SLAM algorithm.
- Use the CS Saucer to test the developed system in the MC-Lab
- Modify the simulator developed in (Ueland, 2016) so that it fits the problem in this thesis, and so that it can be used for verification of the algorithms and methods used in this thesis.

The objective and sub-objectives have been formulated in cooperation with my supervisor, and can also be seen in the thesis description sheet included at the very start of the thesis.

1.5 Thesis Contributions

The main contributions of this thesis are:

- Replacing the set-point generated control law developed in (Ueland, 2016) with a LOS-steering law. The implemented steering law steers the course of the CS Saucer by controlling the velocity of the vessel, rather than using the heading which is the traditional approach.
- Updating the map processing to include the raw laser scans from the 2D LIDAR that was implemented in (Ueland, 2016), in order map dynamic obstacles in the vicinity of the CS Saucer.

- Developing a scheme to convert the Qualisys coordinate system, so that it can be used in a pre-generated map.
- Creating a scheme that lets the Hector-SLAM be redeployed, and continue building on a previously built map. This also allows the position estimate from the redeployed Hector-SLAM algorithm to be used in the already known map.
- Developing a method to include simple simulated obstacle into the map, that simulates the obstacles used in this thesis. The simulated obstacles can be used both in simulations, and in experiments done in the MC-lab
- Developing a sensor fusion scheme to combine the position measured by the MC-lab Qualisys system with the position estimated by the Hector-SLAM algorithm.
- Developing a collision avoidance system that is able to handle static obstacles, and dynamic obstacles up to a certain degree.

1.6 Scope and Delimitations

1.6.1 Thesis Structure

The outline of the thesis is organized as follows:

Chapter 2: Presents a literature review that provides relevant references on path generation in regional and local maps, methods for reactive obstacle avoidance and autonomy layers of an autonomous system.

Chapter 3: Presents the theory behind the A* algorithm that was utilized in this thesis as a global path planner, as well as the theory behind the dynamic window algorithm. The theory behind the Hector-SLAM algorithm is also presented in this chapter.

Chapter 4: Presents the CS Saucer along with the experimental setup regarding the Software Architecture and Hardware Architecture.

Chapter 5: Covers the mathematical model, as well as the various transformations and conversions among the different reference systems, in order to convert information from one reference system to another.

Chapter 6: Covers the guidance, navigation and control of the vessel in detail. The modified Dynamic Window algorithm is presented, as well as the implemented LOS steering law. The chapter also covers the modifications done to the map processing, and the sensor fusion scheme.

Chapter 7: Presents the results from the experiments done in the MC-lab and the simulations done in this thesis.

Chapter 8: The results and the performance of the resulting is discussed in this chapter. Measures that can be taken in order to increase the performance of the

developed guidance system is also discussed.

Chapter 9: The final chapter covers the concluding remarks regarding the performance of the developed system, and suggestions for improvements in the developed system. Ideas for further work is also presented.

1.6.2 Limitations

The following addresses some of the limitations of the system developed in this thesis:

- The experiments are preformed in a controlled environment with the following properties:
    1. There are no environmental forces present during the MC-lab experiments.
    2. The obstacles used for the experiments need to be vertical, and not transparent in order to be detected by the LIDAR.
- An external computer connected through the MC-lab wifi network is needed in order to run the "*Exploration_pathplanner*" node. This is due to the limited computational power of the single-board computer installed on the CS-Saucer.
- If the velocity of the dynamic obstacles is to high, the collision avoidance system will have very little time to react to the obstacle due to the time lag present in the system.

Chapter 2

Literature Review

2.1 Path Generation in global and regional maps

There are many different algorithms one can find that could preform well as a global path finder. In this text it is mainly A*, RRT and D* that will be discussed. They are algorithms that are widely used to solve the path planning problem. Since it is the A* algorithm that is used as a global path planner in this thesis, the focus will be on comparing the A* algorithm with RRT and D*.

2.1.1 A*

The A* algorithm as presented in (Hart et al., 1968), is a best search algorithm that uses a heuristic approach to find the cheapest path from a defined start to a goal in a graph. A definition of graphs can be seen in (Hart et al., 1968). The A* algorithm is an algorithm that is easy to implement, and is widely used in path planning. The algorithm is recognized as very efficient, and is a complete algorithm, meaning that it will find a solution if there exists one. The theory behind the A* algorithm will be presented more thoroughly in chapter 3.1.2. Examples where the A* algorithms is being used as a path planner for marine environments can be found in (Shah and Gupta, 2016) and (Larson et al., 2006). In (Shah and Gupta, 2016) a modified version of the A* was developed, and was intended to be used for computing paths on large marine domains. A real world scenario was also presented, and it was concluded that the paths generated by their modified A* algorithm finds the path much faster than the original A* or the Theta* algorithm in very large maps. In (Larson et al., 2006), an autonomous navigation system was developed for a SEADOO challenger 2000 sport boat. In this system, the A* algorithm was used to generate the path around stationary objects, in combination with a dynamic obstacle avoidance scheme.

2.1.2 D*

The D* algorithm behaves very similar to the A* algorithm, except that it is dynamic in the sense that the arc cost parameters can change during the problem solving process(Stentz, 1994). In comparison to the A* algorithm, D* keeps the already calculated information when the path is determined, and modifies it so that it can reuse the information if the path must be re-planned. This allows the D* algorithm to be more dynamic compared to the A* algorithm, in the way that it can use new information which becomes available and re-plan the path much faster than A*. One

of the downsides of the original D* is that it is quite complex and hard to understand. There are several algorithms that are denoted as D*, such as Field D*, and D*lite. D*lite is an algorithm that is built based on the Lifelong Planning A* (Koenig and Likhachev, 2005). In (Koenig and Likhachev, 2005), D*lite is compared to the D*, and the conclusion is that D*lite determines the same paths as Focused D*, and thus moves the robot in the same way. However, D*lite is algorithmically different from D*, and is a algorithm that in comparison with D* is simpler to understand and implement.

2.1.3 Rapidly-Exploring Random Trees algorithm

Rapidly-Exploring Random Trees is a randomized path planning technique that was introduced in (LaValle, 1998). The RRT algorithm is relatively simple, and it can take the dynamics of the vessel into account (LaValle, 1998). Since RRT is a randomized method, it is not a complete method. The RRT starts by generating a tree structure that rapidly grows from the initial state. With each iteration, a connection is attempted between the drawn sample and the tree if the connection is feasible with regards to constraints and obstacles. This way the tree structure will explore larger and larger portions of the configuration space of the vehicle. In (LaValle, 1998) it is said that the RRT is biased towards places not yet visited, and one can also bias the RRT growing towards specific areas by increasing the probability of sampling states from that specific area (Wikipedia, g). This could be used to make the RRT focus the path more towards the goal.

In (Øivind Aleksander G. Loe, 2008) a version of the RRT algorithm was implemented as a global planner for the marine vessel "Viknes 830". The RRT algorithm was modified to be assisted by the A* algorithm, in a way such that the A* algorithm guides the RRT algorithm towards the shortest path. The performance of the RRT algorithm was deemed to give good performance for both the full scale experiments, and in simulations. Another example where the RRT algorithm has been used as a global path planner can be found in (Arab et al., 2016). Here a modified version of the RRT (a combination of sparse-RRT and RRT*) implemented for a 1/7 scale racing vehicle. The scale-model racing vehicle is expected to preform autonomous aggressive maneuvers. The experimental results demonstrated the high agility maneuvering performance under the autonomous driving control with the motion planner.

2.1.4 A Short Comparison of the Presented Algorithms

In (Loe, 2007) the RRT was compared to other popular path finding algorithms, as well as dynamic obstacle avoidance in combination with these global methods were also compared and discussed. Summing up his conclusion on A* vs RRT: A* generates the shortest possible path, while RRT will always generate sub optimal paths. However RRT includes the dynamics of the vehicle, which guarantees that the path is possible to follow for the vehicle (given that one uses an accurate model). RRT generally doesn't plan the route as risky as the A* algorithm. The A* has an edge over RRT in regards to the computational time it takes to find the path. In other words, they both have their own strengths and weaknesses. When comparing A* and D* the differences lies in re-planning and the computational power needed to generate the path, as well as the complexity of the algorithms. The D* algorithm

or one of the algorithms denoted as D* would suit the objective in this thesis quite well. However, as a part of this thesis a local path planner is implemented to compensate for the dynamic changes in the environment. This will reduce the need for re-planning the global path. The A* has the edge over D* in simplicity and is already shown to work well for the CS Saucer, as can be seen in (Spange, 2016) and (Ueland, 2016).

2.2 Reactive obstacle avoidance

2.2.1 Obstacle Detection

Reliably detecting and recognizing obstacles, in particular dynamic obstacles, is in general difficult. Common sensors or installations used to identify dynamic obstacles for a full scale ship includes:

- Radar
- Sonar
- AIS
- Stereo-camera(s)
- LIDAR

A short and comprehensive description of LIDARs and Stereo-cameras is given in chapter 1.2.4.

Radar

A radar system has a transmitter that emits radio waves in predetermined directions. The radio waves that are reflected back towards the transmitter are then processed and used to estimate the distance to the obstacle as well as its bearing and velocity. By measuring the time it takes for the signal to return and with the knowledge that the radio waves travels at the speed of light, one can deduce the distance to the target obstacle. A moving obstacle causes a Doppler shift in the signal, meaning that the frequency of the reflected signal is dependent on the relative speed between the obstacle and the observer (Wikipedia, c). Only the relative speed can be estimated from this information. However, by tracking the azimuth movement of the object over time one can deduce a speed estimate perpendicular to the relative speed, thereby getting the full scope of the velocity of the object (Wikipedia, e).

Sonar

A short description of how a sonar/acoustic sensor works is given in chapter 1.2.4. Sonars are widely used for ships in order to evaluate the elevation of the seabed, and to detect obstacles that may be present beneath the ocean surface.

AIS

AIS is short for Automatic Identification System. AIS is a system which enables a vessel to publish information about itself, and receive information about other vessels in the area. The information that is shared over the AIS is in general identification and ship characteristics, dynamic information about the vessel and voyage related information. The dynamic information consists of:

- Ship position
- Course over the ground
- Speed over the ground
- Heading
- Navigational status
- Rate of turn (where available)
- Angle of heel (optional)
- Pitch and roll (optional)

The voyage related information consists of the ship's draught and the destination of the vessel, as well as hazardous cargo. Optionally the route plan (way-points) can be shared. The information is transmitted over dedicated VHF frequencies. The update rate of this information over the AIS is dependant on the classification of the vessel as well as its speed and navigational status. For example a class A ship which is moving at a speed of above 23 knots is required to update the dynamic data at an interval of 2 seconds. A class A ship that is anchored or moored has a required update rate of 3 minutes. A class B vessel that has a speed above 2 knots is required to have a update rate of 30 seconds (Bole et al., 2014).

IMO requires that AIS must be installed on board voyaging ships with a gross tonnage of 300 or more, as well as all passenger ships regardless of size. Another point to note is that the quality of the dynamic information published by vessels are dependant on the precision and status of their sensors. The fact that not all vessels can be expected to be equipped with AIS, and that the update rate on the dynamic information from other vessels leads to the conclusion that AIS alone will not grant sufficient situational awareness in many real world scenarios.

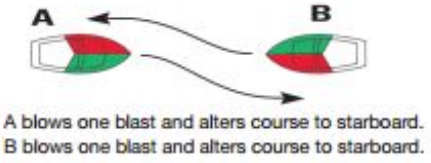
Digital Nautical Charts

Nautical charts are valuable tools in marine navigation. A nautical chart contains information about static objects in an environment, such as coastlines, piers, buoys, the elevation of the seabed and so on. It may also contain information about environmental forces such as currents and tides (Wikipedia, a). In digital form, such charts can be used as an a priori source of information for an autonomous navigation system.

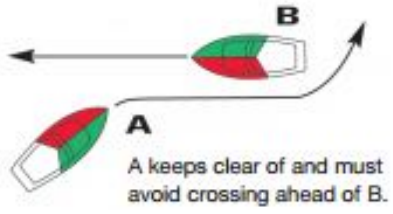
2.2.2 Avoidance Maneuvers

COLREGS – Following the Rules of the Road

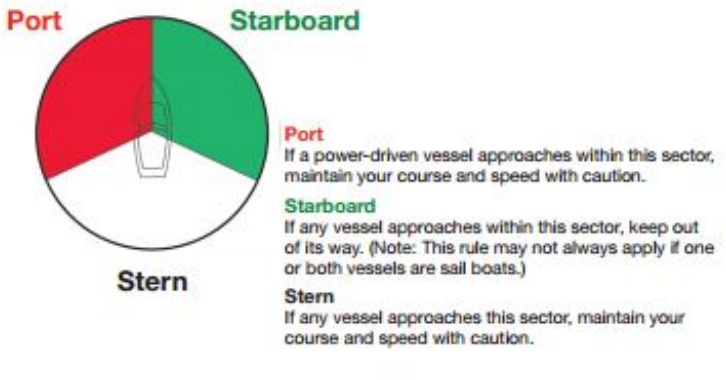
The International Regulations for Preventing Collisions at Sea (COLREGS) are a set of rules defining measures to be taken in order to reduce the risk of collision (Wikipedia, b). Among many other regulations, the rules define which course of action a vessel should take when encountering other vessels at sea. For example in a head on situation, a vessel should pass the other vessel on the port side. In figure 2.1 the head on situation and crossing situation is illustrated. For a crossing situation the give-way vessel should pass behind the other vessel. The give away vessel is defined as the vessel that has the other vessel at the starboard side, as stated by (IMO).



(a) A head on situation



(b) A crossing situation



(c) A figure illustrating the port, starboard and stern

Figure 2.1: A figure illustrating a head-on and a crossing situation for two vessels, courtesy of Transport Cannada

Projected Obstacle Areas

In (Larson et al., 2006) Projected Obstacle Areas (POA) is used to aid the collision avoidance, in conjunction with the Velocity Obstacle (VO) method as a local planner and the A* algorithm as the global path planner. A projected obstacle area is an estimate of the area that a moving obstacle could potentially occupy in the future. Since the possible areas a moving obstacle could occupy is rapidly increasing with time, it is necessary to identify a particular moment in time where the obstacle would pose a threat to the operating vessel. Otherwise the map could potentially be filled with obstacles, and hinder movement. To identify this moment, one must find the closest point of approach, which is the minimum distance between the vessel and

the object in time, along their paths. This can then be done for multiple segments of the path, to create a representation of the areas the vessel should avoid. Since the POA method handles probability and uncertainty values to project the area, one even has the opportunity to modify these so that that the POAs mimic some simple COLREGSs maneuvers.

2.2.3 Dynamic Window

The dynamic window algorithm is a reactive collision avoidance algorithm that was introduced in (Fox et al., 1997), and there are many modifications that has been suggested to this algorithm. In the DW approach the search for commands controlling the vehicle is carried out directly in the space of velocities. The vehicle dynamics is incorporated into the method by taking the acceleration limits into consideration. The only velocities in the velocity space that are considered are those that are reachable within the dynamic constraints. This way the DW prohibits infeasible speed commands, and in that way does not ask for an impossible control action. In addition to this restriction, only velocities that are safe with respect to obstacles are considered. It is a modified version of the dynamic window that is used for collision avoidance in this thesis. The theory behind the original Dynamic Window Algorithm will be presented in a more complete manner in chapter 3.2.

As mentioned earlier, in (Øivind Aleksander G. Loe, 2008) an autonomous navigation system was implemented for the full-scale marine vessel "Viknes 830". In this navigation system, the collision avoidance scheme was a modified version of the DW algorithm. In this case the DW algorithm was modified to be COLREGS compliant, as well including lateral velocities and vehicle accelerations in the predictions. To detect obstacles, the AIS system installed on the vessel was used. The performance of the collision avoidance system behaved well in the full-scale tests. It is concluded that the limiting factors of the collision avoidance are the sensors, as well as the processing power and maneuverability of the vessel. In (Eriksen, 2015) horizontal collision avoidance for autonomous underwater vehicles was studied. The Dynamic Window algorithm that was applied to a HUGIN 1000 AUV was assessed, and a simulator for the AUV with sonar sensor and an integral line of sight guidance system were developed. Since the DW algorithm is not intended for use on vehicles with second order non-holonomic constraints, modifications were made to make the algorithm suited for the HUGIN 1000 AUV. Simulations showed that the modified DW algorithm succeeded in avoiding collision when it was not trapped in a local minimum, even when influenced by ocean currents.

2.2.4 Velocity Obstacle

The Velocity Obstacle method (VO) is a method for avoiding obstacle introduced in (Fiorini and Shiller, 1993). The velocity obstacle is the set of all relative velocities between the vessel and the obstacles that will lead to a collision. The VO approach transforms each moving object into a fixed obstacle, by looking at the relative velocity of the vessel and the obstacles. Then mapping the the obstacle into the vessel's configuration space and using the relative velocity to add feasible sets of velocities and build the VO for the vessel. From this one can see that the tip of the vessel's velocity vector must be outside of the VO in order to avoid collision. Vector operations will then be able to determine the set of one-step maneuvers that

will avoid the obstacle collide with the vessel before a specified time (as seen from the current position of the vessel). In this way a trajectory correction is computed that avoids predicted collisions. One thing to also note, is that in the computation of the avoidance maneuvers the dynamic constraints of the vessel are replaced by constant bounds on the velocity and the turning angle. This way the dynamics of the vessel is not directly considered, but still taken some what into account.

An example of VO being used for in a full-scale experiment can be seen in (Kuwata et al., 2011). Here the VO was modified to be COLREGS compliant. The collision avoidance scheme was implemented for a USV named "PowerVent". Their system utilized two pairs of stereo cameras to estimate the position and velocity of the dynamic obstacles. The dynamic obstacle was a 11m RHIB, used as a traffic vessel. The results showed that the modified VO method behaved well in the full-scale experiments, and satisfied the COLREGS constraints. The system was also verified through simulations. In (Stenersen, 2015) the Velocity obstacle is used to avoid obstacles, in compliance with COLREGS in a simulated environment. Through simulations in diverse scenarios in combination with discussions of situations proving extra challenging, a thorough analysis of the VO method as a basis for a collision avoidance system was given.

2.2.5 Potential Field Method

The Potential field method is a method that was introduced in (Khatib, 1986). Here the general idea is that obstacles exert a repulsive forces on the vehicle, while the target or the goal asserts an attractive force. The resultant force acting on the vessel, determines the direction and the speed of the resulting travel (Koren and Borenstein, 1991). The potential field method depends on knowledge about obstacles in the area to be able to navigate.

A slightly different version of the method is the Virtual Force Field (VFF method. The VFF is able to generate a map represented by a two-dimensional Cartesian histogram grid by using sensor readings. The histogram grid contains the probabilities of obstacles being located different places in the area. The repulsive forces are then also modified by the probability values that tells the chance of there being a obstacle located there. In (Borenstein and Koren, 1991) some of the limitations of the VFF method are mentioned. One limitation is that it may not grant passage between closely spaced object. Another problem that may occur, is that when the vessel is traveling in narrow corridors, oscillations may start to develop if it does not travel along the center-line of the corridor.

2.2.6 Vector Field Histogram

The Vector Field Histogram (VFH) method was first presented in (Borenstein and Koren, 1991). The goal of VFH is to eliminate the shortcomings of VFF while retaining the advantages of its predecessors. While the VFF generates control commands directly from the two-dimensional histogram grid, the VFH introduces a: intermediate-level one-dimensional polar histogram. This polar histogram is constructed around the vessels momentary location, and use this map to help making good control choices. In (Borenstein and Koren, 1991) the VFH method is imple-

mented for a mobile robot named CARMEL. The proximity sensors used for the robot was 24 ultrasonic sensors that were ringed around CARMEL. The obstacles in the presented results were unknown static obstacles. In the conducted experiments the VFH method was concluded to behave well. In (Loe, 2007) DW and VFH was used and compared in a simulation environment with dynamic obstacles. The resulting discussion deemed DW superior to VFH for obstacle avoidance.

2.3 Autonomous Systems, Layers and Taxonomy

2.3.1 Layers Of Autonomy

An illustration of proposed layers for an autonomous marine system can be seen in figure 2.2, which presents a "bottom-up" approach towards autonomy .
From this figure, one can see the proposed control architecture for an autonomous system, which consists three layers defined as:

- Mission planner layer: Here the mission objective is defined and the mission is planned. Subject to contingency handling, any input from payload sensor data analysis and any other input from the autonomy layer, the mission may be re-planned.
- Guidance and optimization layer: This layer handles way-points and references commands to the controller.
- Control execution layer: Consists of actuator control and plant control. On board data processing is also handled at this level.

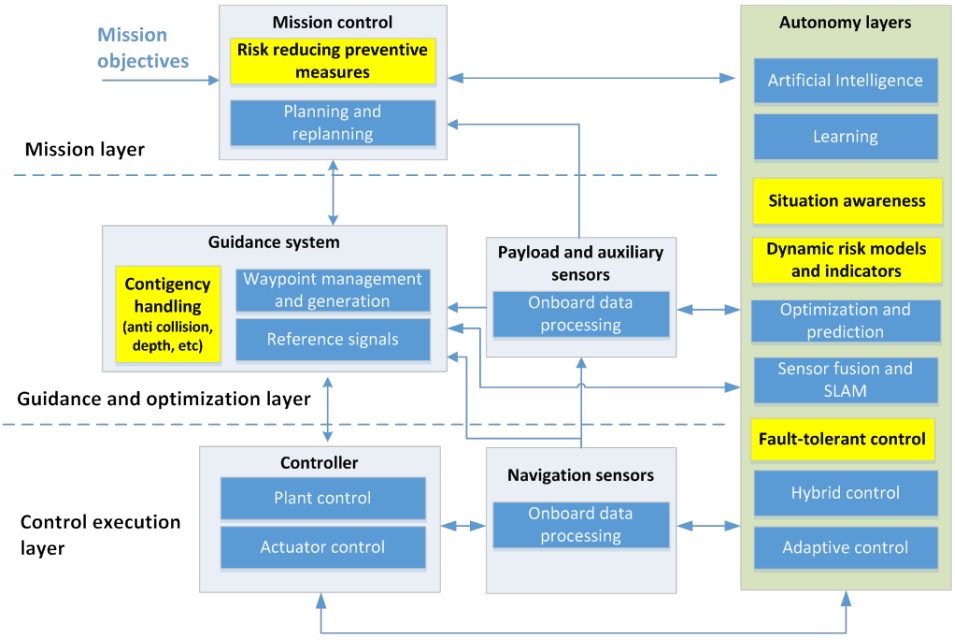


Figure 2.2: Proposed layers of an autonomous marine system. Courtesy: Asgeir J. Sørensen, 2015.

This architecture is taken from (Sørensen and Ludvigsen, 2015), and was inspired by teh work done in (Hagen et al., 2009).

2.3.2 Autonomous System Taxonomy

A autonomous system has different levels of autonomy, and can range from low-level assistance in decision making for a human operator to high-level autonomous decision making without human intervention (Fjellheim et al.). (Fjellheim et al.) presented a table for the different levels of autonomy, which is shown in the list below.

- level 1, Human Operated: The activity is a result of human initiated control inputs. The system has no autonomous control, although it may have information only responses to sensed data
- level 2, Human Assisted: The system can preform activity in parallel with human input, to make it easier for the human to preform the desired activity.
- level 3, Human Delegated: The system can preform a limited control activity, if it has been delegated the task.
- level 4, Human Supervised: The system can preform a variety of activities if given permission or direction by a human. The system provides sufficient information about the the process and the internal operation and behaviors, so that the supervising human can use this to redirect the system.
- level 5, Mixed Initiative: Both the human and the system can initiate behaviors on sensed data, where there is a variety of means provided to regulate the authority of the system, with regards to human operations.
- level 6, Fully Autonomous: The system requires no human operations to preform any of its designated activities across all planned ranges of environmental conditions. The system is able to make critical decisions on its own, and act based on its situational awareness.

This list was presented to illustrate that even though the list is not wrong in itself, there are many ways to present the levels of autonomy of a system. The literature are littered with different definitions and numbers with regards to the levels of autonomy. In (Vagia et al., 2015) a literature review on the definitions of the levels of autonomy over the years has been done. They reviewed and compared the different taxonomies different authors has used to classify the level of autonomy of a system. The authors used all this information and proposed a taxonomy of their own. An overview of this can be seen in figure 1 in (Vagia et al., 2015).The levels of autonomy as that was proposed by the authors in (Vagia et al., 2015) is as listed below. It is this taxonomy that will be used for this thesis.

- Level 1, Manual control: Computer offers no assistance
- Level 2, Decision proposal stage: The computer offers some decisions to the operator. The operator is responsible to decide and execute.
- Level 3, Human decision select stage: The human selects one decision and the computer executes
- Level 4, Computer decision select stage: The computer selects one decision and executes with human approval

- Level 5, Computer execution and human information stage: The computer executes the selected decision and informs the human
- Level 6, Computer execution and on call human information stage: The computer executes the selected decision and informs the human only if asked
- Level 7, Computer execution and voluntarily information stage: The computer executes the selected decision and informs the human only if it decides to
- Level 8, Autonomous control stage: The computer does everything without human notification, except if an error that is not into the specifications arrives. In that case the computer needs to inform the operator

Chapter 3

Theory

3.1 The Path Planning Problem

3.1.1 Search Graph

A search graph can be represented in many ways, and popular examples of metric or grid-based techniques include Meadow Maps, Voroni Diagrams, Regular Occupancy Grid and Quadtree Mapping (Campbell et al., 2012). The most common of these is the occupancy grid, which is the method that was utilized in (Ueland, 2016) and still used in this thesis. An occupancy grid represents the map as discretized into a grid where each cell in the grid denotes a possibility of being occupied. In a regular occupancy grid the entire cell is deemed as occupied if any point in the cell is occupied by an object.

To be able to solve the path planning problem with graph theory, one needs to generate nodes and place them in the map. There are several ways to do this, which can be seen in (Geraerts and Overmars, 2004). A simple way to do this for an occupancy grid, is to let each cell in the grid be a node. This can lead to a large amount of nodes, and increases the computer processing demands on the system. The nodes will represent a discrete position in the map, and paths to neighboring nodes will be easy to asses.

3.1.2 The A* Algorithm

To find an optimal path, the A* uses a heuristic approach, as stated above. The evaluation function $f(n)$ often referred to the f-score, is the function that is minimized and is determined by the equation:

$$f(n) = g(n) + h(n) \tag{3.1}$$

Where $g(n)$ is the cost that is accumulated from the start node by traversing other nodes in the graph, in order to reach n. $h(n)$ is the heuristic estimate of the cheapest path from n to the goal, ignoring the obstacles. For this case, the equation for the heuristic cost becomes:

$$h(n) = \sqrt{(goal_x - n_x)^2 + (goal_y - n_y)^2} \tag{3.2}$$

The algorithm operates with two lists to keep track of the progress, the open list and the closed list. The open list is a list of all the "discovered" nodes that has yet to be

evaluated, where the one with the lowest f-score is chosen to be evaluated for each iteration. This means that the algorithm solves the problem by always evaluating the most promising nodes, which classifies it as a best-first algorithm. After a node has been evaluated it is then put in the closed list, so that one can keep track of which nodes that has been evaluated, and use this to avoid having to re-evaluate nodes. To be able to recreate the path after the search is finished, one must also make sure that each node remembers its predecessor/parent with the lowest f-score, so that one may determine where it stemmed from when the search is finished.By using this information, it can be determined which set of nodes that generates the opitmal path. If the Heuristics does not over predict the minimum cost to reach the goal, the A* algorithm will find the shortest to the goal (Hart et al., 1968).

3.2 The Dynamic Window Algorithm

The dynamic window algorithm was first introduced in (Fox et al., 1997). The method assumes that the velocity and rate of turn for the vessel can be set as a constant within a given time interval. In addition, by neglecting the sway velocity of the vehicle, a trajectory of the vessel during the time interval may be estimated by using the velocity vector $[u_i, r_i]^T$. This allows the trajectories to be calculated as circular trajectories described by:

$$a_x^i = x - \frac{u_i}{r_i} sin(\theta) \tag{3.3}$$

$$a_y^i = x + \frac{u_i}{r_i} cos(\theta) \tag{3.4}$$

Where (x,y) describes the position of the vessel, u_i is the vessel's surge speed during interval i, r_i is the rate of turn during interval i and θ is the vehicle heading. The radius of the arc is defined by:

$$a_r^i = \left| \frac{u_i}{r_i} \right| \tag{3.5}$$

In order to find the optimal trajectory for a given time interval, a whole set of trajectories generated by different velocity vectors needs to be evaluated. In order to reduce the search space, dynamic and safety constraints are set, and must be satisfied in order to deem a given velocity vector feasible.

Admissible Velocities

One of the constraints that are set is that a velocity vector, $[u_i, r_i]^T$ must be admissible, which means that it must lead the vessel along a safe trajectory. The pair (u_i, r_i) is only considered admissible if the robot is able to stop before it reaches the closest obstacle on the corresponding trajectory. The admissible velocities can be defined as the velocities that satisfies the following equation:

$$V_a = \{(u,r)|\ |u| \leq \sqrt{2 \cdot dist(u,r) \cdot \dot{u}_b}\ ,\ \wedge\ |r| \leq \sqrt{2 \cdot dist(u,r) \cdot \dot{r}_b}\} \tag{3.6}$$

Where $dist(u,r)$ is the distance the vessel can travel along the arc without hitting a obstacle. $\dot{u}_b$ and $\dot{r}_b$ is the acceleration for breakage. Thus V_a is the set of velocities which allows the vessel robot to stop without colliding with a obstacle.

The Dynamic Window

The second constraint is that the velocity search space must be reduced to the velocities that are reachable within the given interval. This restriction takes the limited accelerations obtainable by the vessel in a given interval. These velocities are denoted as the Dynamic Window and denoted as V_d. The velocities must satisfy the equation below:

$$V_d = \{(u, r) \mid u \in [u_a - \dot{u} \cdot \Delta t, u_a + \dot{u} \cdot \Delta t] \wedge r \in [r_a - \dot{r} \cdot \Delta t, r_a + \dot{r} \cdot \Delta t]\} \quad (3.7)$$

Where $[u_a, r_a]^T$ is an admissible velocity vector.

The Resulting Search Space

The resulting constraints on the velocities leaves the set V_r within the dynamic window. V_s can be seen as the search space, containing all the candidate velocities, which enables V_r to be defined as follows:

$$V_r = V_s \cap V_a \cap V_d \quad (3.8)$$

Selecting Optimal Velocities

After the search space has been determined, a velocity needs to be selected from the set V_r. By maximizing a objective function that gives a score to each velocity vector based on the criteria *heading*, *distance* and *speed*. The objective function has the form:

$$G(u, r) = \sigma(\alpha \cdot heading(u, r) + \beta \cdot dist(u, r) + \gamma \cdot speed(u, r)) \quad (3.9)$$

Where $heading(u, r)$ represents the alignment of the vehicle with the target heading/direction, and is denoted as $180 - \theta$. θ is the angle of the target point relative to the vehicle predicted heading direction at the tip of a trajectory. $dist(u, r)$ represents the distance to the closest obstacle along that intersects with a given trajectory. If no obstacle is present on a given trajectory it is set to a large constant. $speed(u, r)$ is used to evaluate the speed the vehicle will achieve along a given trajectory. The constants α, β and γ are weighing constants, which needs to be tuned in order to set the importance of each criteria. The σ function is meant to be a smoothing function, such as a low-pass filter, in order to reduce fluctuations.

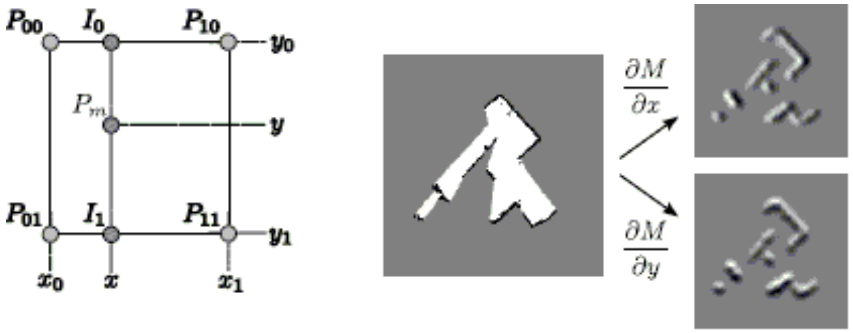
All three components of the objective function $G(u, r)$ are necessary in order to avoid obstacles in a satisfying manner. For example, if one solely maximizes the $speed(u, r)$ and $dist(u, r)$ functions there would be nothing to keep the vehicle on track towards the goal. By solely maximizing the $heading(u, r)$ function, the vehicle will be stopped by the first obstacle it encounters on the trajectory towards the goal, since there will be taken no action to try and circumvent it. By including all the three parameters, and tuning their weight constants the vehicle will circumvent collisions as fast as it can under the dynamic constraints, while still making progress towards reaching its goal.

3.3 Simultaneous Localization and Mapping

3.3.1 The Hector Slam algorithm

The Hector SLAM algorithm is an open source 2D SLAM algorithm, and the theory presented in this chapter can be seen in (Kohlbrecher et al., 2011). To be able to represent the arbitrary environment, an occupancy grid map is used. The LIDAR platform might exhibit 6 DOF motion, the estimated platform orientation is used to convert the scan into a point cloud of scan endpoints. In the hector SLAM approach, filtering based on the endpoint z coordinate is used, so that only endpoints within a threshold of the intended 2D plane is processed.

Because of the discrete nature of an occupancy grid, an interpolation scheme allowing sub-grid cell accuracy through bi-linear filtering is employed. This scheme is used for estimating both the occupancy probabilities and derivatives. Intuitively, grid map cell values can be viewed as as samples of an underlying continuous probability distribution.
From a continuous map coordinate P_m one can estimate the occupancy value $M(P_m)$ and the gradient $\nabla M(P_m) = (\frac{\delta M}{\delta x}(P_m), \frac{\delta M}{\delta y}(P_m))$ using the four closest integer coordinates $P_{00}, P_{01}, P_{10}, P_{11}$ as is depicted in figure 3.1.



(a) (b)

Figure 3.1: (a) bilinear filtering of the occupancy grid map, where point P_m is the point whose value shall be interpolated (b) Occupancy grid map and spatial derivatives, courtesy of Kohlbrecher et al. (2011)

If the sample points /grid cells of the map are situated on a regular grid with distance 1 from each other, the linear interpolation along the x- and y- axis will yield:

$$\begin{aligned} M(P_m) \approx & \frac{y - y_0}{y_1 - y_0}\left(\frac{x - x_0}{x_1 - x_0} M(P_{11}) + \frac{x_1 - x}{x_1 - x_0} M(P_{01})\right) \\ & + \frac{y_1 - y}{y_1 - y_0}\left(\frac{x - x_0}{x_1 - x_0} M(P_{10}) + \frac{x_1 - x}{x_1 - x_0} M(P_{00})\right) \end{aligned} \tag{3.10}$$

The derivatives can be approximated as:

$$\frac{\delta M}{\delta x}(P_m) \approx \frac{y - y_0}{y_1 - y_0}(M(P_{11}) - M(P_{01})) + \frac{y_1 - y}{y_1 - y_0}(M(P_{10}) - M(P_{00})) \tag{3.11}$$

$$\frac{\delta M}{\delta y}(P_m) \approx \frac{x - x_0}{x_1 - x_0}(M(P_{11}) - M(P_{10})) + \frac{x_1 - x}{x_1 - x_0}(M(P_{01}) - M(P_{00})) \tag{3.12}$$

Scan Matching

Scan matching is the process of aligning laser scans with each other, or with an already existing map. Modern laser scanner has very low distance measurement noise and high scan rates, and the precision of a laser scanner might be much higher than the odometry data for many robot systems. By optimizing the alignment of the scan beam endpoints with the map built so far, one gets an estimate of the pose of the vehicle. As scans get aligned with the existing map, the matching is implicitly preformed with preceding scans. This was inspired by computer vision (Lucas and Kanade, 1981), and uses a Gauss-Newton approach. By using this approach there is no need for a data association search between beam endpoints or an exhaustive pose search.

The goal of the scan matching is to find the rigid transformation $\xi = (p_x, p_y, \psi)^T$ that minimizes the equation:

$$\xi^* = arg\ min_\xi \sum_{i=1}^{n} [1 - M(S_i(\xi))]^2 \tag{3.13}$$

In other words, the goal is to find the transformation that gives the best alignment of the laser scan with the map. Here the $S_i(\xi)$ is the world coordinates of the scan endpoints $s_i = (s_{i,x}, s_{i,y})^T$. These coordinates are a function of the vehicle pose ξ in world coordinates given as:

$$S_i(\xi) = \begin{bmatrix} cos(\psi) & -sin(\psi) \\ sin(\psi) & cos(\psi) \end{bmatrix} \begin{bmatrix} s_{i,x} \\ s_{i,y} \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \end{bmatrix} \tag{3.14}$$

The function $M(S_i(\xi))$ gives the map value at position $S_i(\xi)$. Given an estimate of the initial position, one can estimate $\Delta\xi$ which is used to optimize the error measure:

$$\sum_{i=1}^{n} [1 - M(S_i(\xi + \Delta\xi))]^2 \to 0 \tag{3.15}$$

By first order Taylor expansion of $M(S_i(\xi + \Delta\xi))$ one gets:

$$\sum_{i=1}^{n} \left[1 - M(S_i(\xi)) - \nabla M(S_i(\xi)) \frac{\delta S_i(\xi)}{\delta \xi} \Delta\xi\right]^2 \to 0 \tag{3.16}$$

This equation is then minimized by setting the partial derivative with respect to $\Delta\xi$ to zero.

$$2 \sum_{i=1}^{n} \left[\nabla M(S_i(\xi)) \frac{\delta S_i(\xi)}{\delta \xi}\right]^T \left[1 - M(S_i(\xi)) - \nabla M(S_i(\xi)) \frac{\delta S_i(\xi)}{\delta \xi} \Delta\xi\right] = 0 \tag{3.17}$$

Further, solving the minimized equation for $\Delta\xi$ will yield the Gauss- Newton equation for the minimization problem:

$$\Delta\xi = H^{-1} \sum_{i=1}^{n} \left[\nabla M(S_i(\xi)) \frac{\delta S_i(\xi)}{\delta \xi}\right]^T [1 - M(S_i(\xi))] \tag{3.18}$$

Where:

$$H = \left[\nabla M(S_i(\xi))\frac{\delta S_i(\xi)}{\delta \xi}\right]^T \left[\nabla M(S_i(\xi))\frac{\delta S_i(\xi)}{\delta \xi}\right] \tag{3.19}$$

Using this relation, and combining it with equation 3.14, one gets:

$$\frac{\delta S_i(\xi)}{\delta \xi} = \begin{bmatrix} 1 & 0 & -sin(\psi)s_{i,x} & -cos(\psi)s_{i,y} \\ 0 & 1 & cos(\psi)s_{i,x} & -sin(\psi)s_{si,y} \end{bmatrix} \tag{3.20}$$

The Gauss-Newton equation 3.18 can now be evaluated using $\nabla M(S_i(\xi))$ and $\frac{\delta S_i(\xi)}{\delta \xi}$, yielding a step $\Delta\xi$ towards the minimum. It is important to note that the algorithm works on non-smooth linear approximations of the map gradient $\nabla M(S_i(\xi))$, meaning the local quadratic convergence towards a minimum cannot be guaranteed. Still, in (Kohlbrecher et al., 2011) it was concluded that the algorithm works with sufficient accuracy in practice.

Chapter 4

Experimental Setup

4.1 The CS Saucer

As stated before, the vessel used in this thesis is the model-scale surface vessel CS Saucer. The CS Saucer is a fully actuated vessel and was designed to be omnidirectional with a spherical shaped hull. Its top and bottom diameter are $548mm$ and $398mm$, respectively. The mass of the vessel is approximately 3.4 kg. The vessel is fitted with three azimuth thrusters, which are driven by three Torpedo 800 motors.

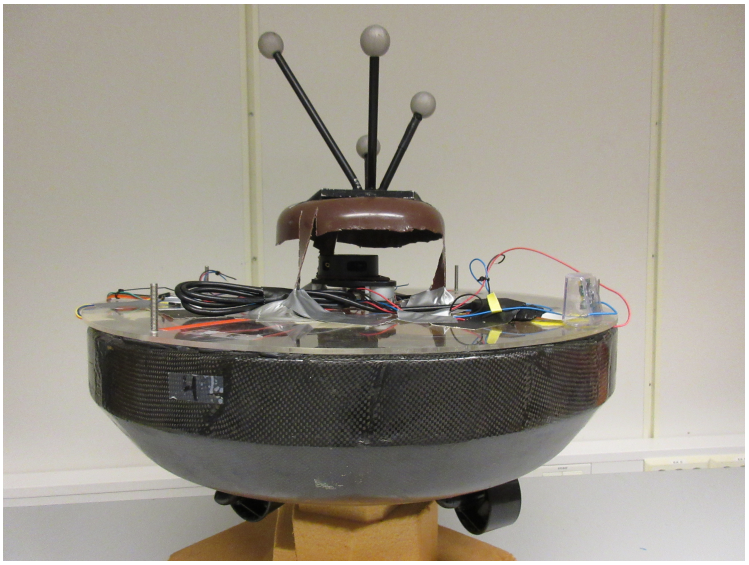


Figure 4.1: A picture of the CS Saucer

4.1.1 Software Architecture

For the CS Saucer, a Raspberry Pi 2 (RPi2) serves as the onboard computer running Linux as operating system. The programming platform utilized on the vessel is the Robot Operating System (ROS). The vessel utilizes a Audrino Mega which is connected to the RP2 through a USB. The Audrino is installed with a flash memory, and is responsible for relaying inputs to the actuators.

Robot Operating System

ROS is a software framework for developing robot applications. Similar to a conventional operating system, ROS provides services such as message parsing between processes, package management, hardware abstraction, device control and implementation of commonly used functionality (Wikipedia, f). A ROS system consists of one or several independent processes called nodes, that shares information through message passing. An example of a subscriber/publisher system can be seen in figure 4.2.

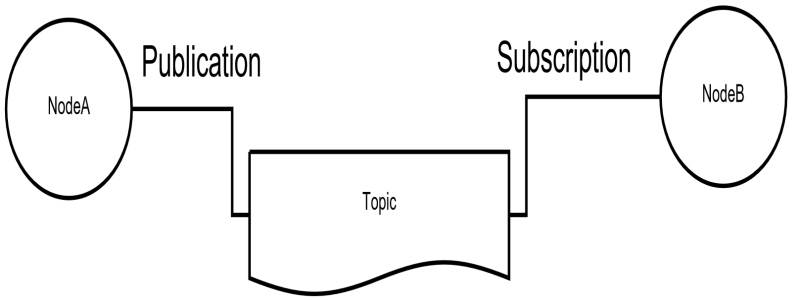


Figure 4.2: ROS publisher/subscribe architecture, courtesy of Ueland (2016)

Figure 4.2 shows Node-A publishing a message to a topic, while Node-B subscribes to the same topic. This way, the two nodes are connected through the topic, but in a way not dependent on each other as long as the topic is there. A node may subscribe and publish to several topics simultaneously. A topic may also receive information from-, and give information to several nodes at the same time. A overview of the node network during simulations can be seen in figure E.1. A suggested software architecture can be seen in figure 4.3, which is taken from (Ueland, 2016).

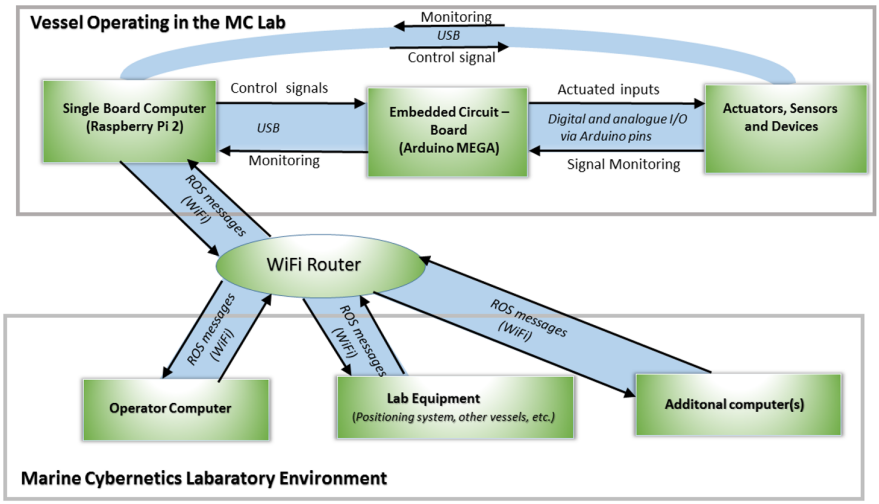


Figure 4.3: Signal flow between the components of a system with the suggested architecture, courtesy of Ueland (2016)

4.1.2 Hardware Architecture

Raspberry Pi 2

A Raspberry Pi 2 installed with Ubuntu serves as the onboard computer. The RPi2 is a single-board computer, and is responsible for running a number of ROS nodes. The RPi2 is connected with the LIDAR and Arduino via USB, and is also equipped with a USB wireless adapter that is able to connect to wireless networks.

Arduino Mega

The Arduino Mega is a low-cost embedded circuit board, equipped with 54 digital pins, where 14 may be used as Pulse Width modulation (PWM) outputs. The Arduino is used for low-level control of the servos and motors. The Audrino is powered directly from the USB connection to the RPi2. An overview of the pi connections is given in table 4.1

Pin Number	Description	Type
3	Angle Servo-1	PWM (Output)
5	Angle Servo-2	PWM (Output)
6	Angle Servo-3	PWM (Output)
9	Revolution Speed Motor 1	PWM (Output)
10	Revolution Speed Motor 1	PWM (Output)
11	Revolution Speed Motor 1	PWM (Output)
DGND	Ground Servos and motors	Ground
GND	Ground Battery	Ground
V-In	Power	Power

Table 4.1: Pin overview

Motors And Servos

Three azimuth thrusters driven by Torpedo 800 motors actuates the vessels. The thrusters can be rotated in 360° by servos of the type Graupner Schottel drive unit II assigned to each thruster.

RPlidar

The Lidar installed on the vessel is a RPlidar. The RPlidar scans the 2D plane by the use of a rotating head that turns 360°. The rate of turn is customizable and can be set between 2-10 Hz, while the sampling frequency is 2000 Hz. The RPlidar has a range of approximately 6 meters. The RPlidar is placed on the lid of the vessel, causing the 2D scan plane to be approximately 10 cm above the water surface during operations. The RPlidar is connected to the RPi2 over USB, and is interfaced to the ROS framework through a separate node.

Battery

Two 640mAh, 11.1V Lithium Polymer (LiPo) were utilized as a the power source for the CS Saucer. The LiPo Batteries was three celled, but had the balance wires integrated with the connector. Monitoring of the voltage as it was done in (Ueland,

2016) was not done because of this. This means that the voltage level of the batteries needed to be checked manually. Fully charged the voltage of the battery is 12.5V, and will supply the CS saucer for about 4 hours when the thrusters and RPlidar is operational. It takes approximately 1 hour to charge one of the batteries, and they should be charge when the voltage is near 11.1V.

Laptop Computer

A laptop computer running ROS and connected to the system through WiFi was also used. The purpose of the laptop was to establish a Human Interface for the operator, where the human in the loop set the desired point the vessel should reach and monitor the process. Due to the limited power of the RPi2, the computer was also used to run the most computational heavy ROS nodes.

During the experiments, an additional computer was used to run the motion controller node. This was recommended in (Ueland, 2016) and (Spange, 2016) and was done to reduce the workload of the RPi2, and to act as an extra safety in order to turn off the motors should any unexpected behavior occur.

4.2 The NTNU Marine Cybernetics Laboratory

The environment the experiments were preformed in is the Marine Cybernetics Laboratory (MC-lab) basin. The MC-lab is a small basin equipped with a wave generator, a towing cart and a positioning system. The basin has a depth of 1.5m, a length of 40m and a width of 6.45m. The wave generator is not used in this thesis. In figure 4.4 a picture of the MC-lab can be seen.

Figure 4.4: A photograph of the MC-Lab

4.3 Qualisys Motion Capture System

The positioning system that is available in the MC-lab is a Qualisys motion capture system. The Qualisys motion capture system has the ability to give the 6 DOF pose of a vessel. The system has millimeter precision, works in real time and and is configured to have a refresh rate of 50Hz.

The positioning system consists of three Oqus high speed infrared cameras. The vessel must be equipped with infrared reflectors, which will enable the cameras to register the position of the reflectors. The data collected from the cameras are transmitted to a dedicated computer over a peer-to-peer (P2P) network. The dedicated computer runs the Qualisys Track Manager (QTM) software, which processes the acquired data. QTM preforms triangulation in order to the deduce the 6 DOF position of the vehicle, and broadcasts the position over the wireless network. The Qualisys system's infrared Oqus cameras can be seen in figure 4.5.

Figure 4.5: The infrared cameras utilized by the Qualisys system

By using a ROS qualisys driver, and editing it to contain the IP-address for the dedicated computer, one can import the Qualisys data into ROS. The Qualisys data will then be published as a ROS topic, and the published data can be used by other ROS nodes. A manual on how to accomplish this is referred to in the appendix B.2.

Chapter 5

Mathematical Model

5.1 Reference Frames

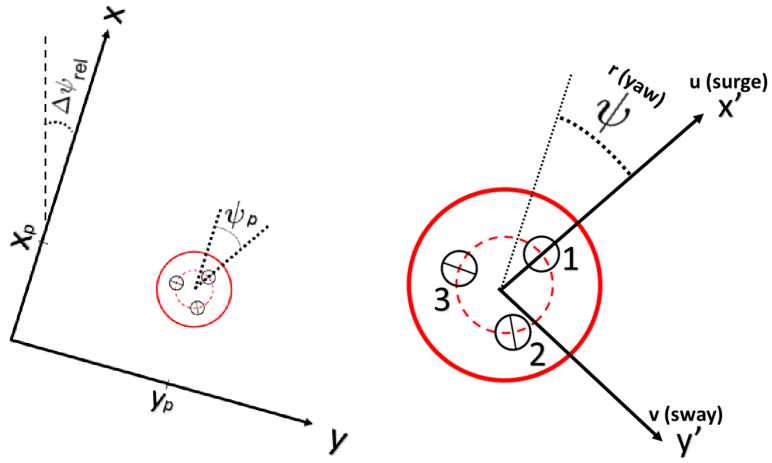
5.1.1 Hector-SLAM Reference Frame and Basin-Relative Reference Frame

The Basin relative frame is the frame that is applied for local control of the vessel, as it was discussed in (Ueland, 2016) and (Spange, 2016). The Basin relative frame has its positive x-axis in the direction the LIDAR is pointing when the Hector-SLAM nodes are initialized. The origin of the reference frame is located at the position of the vessel at the initialization of the Hector-SLAM node. This means that each time the Hector-SLAM is initialized and makes a "new map", the resulting relative basin frame is dependent on the position of the vessel in the basin, as well as the orientation of the LIDAR. One thing to note however, is that in the coordinate system generate by the Hector-SLAM package, the z-axis is pointing upwards. In the Basin-relative frame the z-axis points downwards. This means that the positive y-axes and positive yaw in the Hector-SLAM frame points in the opposite direction of the Basin-relative reference frame. In (Ueland, 2016) a ROS node ($hector2VesselPos - node$) is responsible for converting the Hector-SLAM generated reference frame to the basin relative reference frame. Since the orientation of the vessel is given in quaternions, the node also converts the orientation into Euler angles.

For the Basin-relative frame the heading of the vessel is defined as zero when thruster 1 is pointing along the x-axis relative to the center of origin of the vessel. The z-axis is pointing downwards, and the heading is defined as positive in the clockwise direction. The vessel position and heading in the Basin-relative frame in vectorial frame is given as follows:

$$\eta = [x \;\; y \;\; \psi]^T \tag{5.1}$$

Figure 5.1a illustrates the vessel in the Basin-relative reference frame. $\Delta\psi_{ref}$ represents the angular rotation between the Basin-relative frame and the basin.



(a) Basin relative reference frame (b) Body-fixed reference frame

Figure 5.1: Illustration of the Basin-Relative frame and body frame, courtesy of Ueland (2016)

5.1.2 Body-Fixed Reference Frame

The Body-fixed reference frame, is a reference frame used describe the local behaviour of the vessel, and is a coordinate system that is "fixed" on the body of the vessel. The Body-fixed reference frame can be seen in figure 5.1b. The axes in the body fixed reference frame is denoted as x', y' and z'. The velocity in this reference system is described as u along the x'-axis, v along the y'-axis and r is the angular velocity about the z'-axis, and is positive in the clockwise direction. The velocity vector becomes:

$$v = [u \;\; v \;\; r]^T \tag{5.2}$$

5.1.3 Transformation Between Reference Frames

To establish a relationship between two reference systems one needs to transform a given position from one reference frame, into the corresponding position in the other reference frame. To align a value in one system with another, one could use a rotation matrix to rotate one reference systems axes to become parallel with the other. A 3 DOF rotation matrix commonly used to transform body fixed motion into a global reference frame is presented below:

$$R(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

And rotates a vector, for example the Body-fixed velocities into Basin-relative velocities by doing the transformation:

$$\dot{\eta} = R(\psi)v \tag{5.4}$$

Where $\dot{\eta}$ is the velocity in the Basin-relative frame, and ψ is the heading of the vessel in the Basin-relative frame. This rotation does not account for the cases where the two reference systems have different origins or scaling. However, for the special case

of rotating velocities, a difference in the origin between the systems does not matter. In this thesis it is assumed that the scaling is the same in the reference systems, since all measurements are given in meters. When transferring the position in one reference frame to another one must also take the relative origin offset into account. If given a set of measurements, the conversion can be done by using the relation:

$$\eta = R(\overline{\Delta\psi})\eta' + \overline{\Delta\eta_o} \tag{5.5}$$

Where $\overline{\Delta\psi}$ is the mean heading difference between the two reference frames and η' is the position that is being rotated from one reference frame to the other. $\overline{\Delta\eta_o}$ is the resulting mean offset after the rotation, and the value shows how the origins of the two reference systems is placed relative to each other.

5.1.4 Qualisys Reference Frame- and the Redeployed Hector-SLAM Reference frame- Transformation

Qualisys Reference Frame to Basin-relative Frame

As stated before, the system developed in this thesis uses a map that is generated beforehand. This is done by running the complete system developed in (Ueland, 2016), and using the resulting explored map. The map generation will be further discussed in chapter 6.6. This means that the rootation/conversion of the Qualisys reference frame to Basin-relative frame can be done offline. By creating a ROS node that logs the Qualisys position and the Hector-SLAM position while the map is generated, one can use the resulting information to fully convert the Qualisys reference system.

Since the conversion can be done offline, one can take some small shortcuts in order to find some of the conversion parameters. In order to convert the one reference frame into the other one needs to calculate the parameters $\overline{\Delta\psi}$ and $\overline{\Delta\eta_o}$, by using the logged data sets. As mentioned in chapter 5.1.3 $\overline{\Delta\eta_o}$ is calculated after the rotation. This makes it possible to use a for-loop that attempts to align the Qualisys data into the Basin-relative frame. This is done for every angle in the set $[0; 0.005; 2\pi]$ where 0.005[rad] is the step size. After the rotation of each angle, the mean offset in x- and y- position is compensated for. To find the angle corresponding to the optimal alignment between the data sets, the accumulated absolute error in the x- and y-position is minimized. The optimal rotation angle corresponds to the lowest accumulated error in the alignment, and the offset in x- and y-axis for this angle has already been estimated in the for-loop. The parameters needed to convert the Qualisys measurements into Basin-relative frame has now been estimated, and equation 5.5 can be used to convert the Qualisys measurements.

Calculating the rotation angle in this fashion will introduce inaccuracies in the rotated position. Since the resulting rotation angle and the offset in x- and y-position is not updated as the while the system is operative, these inaccuracies will persist. However, one can almost assume that the maximum error the rotation angle will have in comparison with the "true rotation angle" is a function of the step size used to estimate the rotation angle. This step size was set to be 0.005[rad]. So the error in the rotation angle can be said to be approximately $\pm 0.005[rad]$. Such an error is acceptable, and comparable to using the mean heading difference as the rotation angle. Figure 5.2 illustrates the Qualisys Reference frame. To avoid confusion, the

converted Qualisys reference frame will be called Qualisys Basin-relative frame, or simply Basin-relative frame.

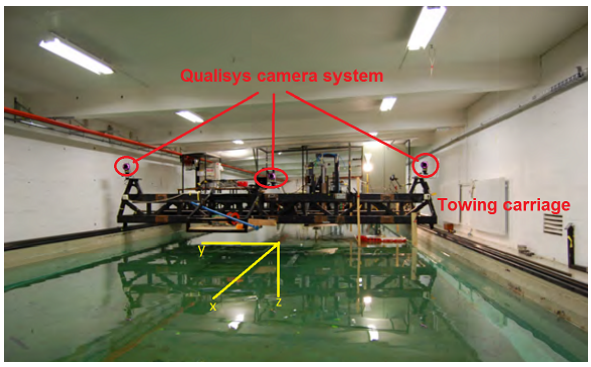


Figure 5.2: A picture describing the different components of the qualisys system. The Qualisys reference frame is drawn into the picture. Courtesy of MC-Lab handbook

Redeployed Basin-relative Frame to Qualisys Basin-relative Frame

As stated before, the Hector-SLAM ROS node is redeployed for the experiments. This means that the Basin-relative Frame for the MAP generated by the redeployed Hector-SLAM differs from the map that was generated beforehand. The position estimated by the redeployed Hector-SLAM node must be rotated to the Basin-relative frame corresponding to the known map before it can be used to navigate within the map. Since the Qualisys position is converted off-line, it will be able to navigate in the known map from the start of the experiment. By using the converted Qualisys measurements, the redeployed Hector-SLAM position can be converted to the Qualisys Basin-relative frame online.

By using a set of measurements for the Qualisys position and the redeployed Hector-SLAM position the conversion parameters can be determined. For the online case, using the mean heading difference will require much less computational power than the for loop utilized when converting the Qualisys position. By determining $\overline{\Delta\eta_o}$ and $\overline{\Delta\psi}$ for the set of measurements, one could use equation 5.5 to convert the redeployed Hector-SLAM position to the Qualisys Basin relative-frame. The set of measurements being evaluated are constantly being updated, where old measurements are replaced with new measurements, and bad measurements are filtered out. This will lead to the rotation parameters changing during the experiments.

By doing this conversion, the redeployed Hector-SLAM position can be used to navigate in the known map in much the same way as the converted Qualisys position. Since a whole set of measurements is used to determine the conversion parameters, the converted Hector-SLAM position regains some of its independence relative to the converted Qualisys measurement. A clear drawback of doing it this way is that the errors already present in the Qualisys conversion will be accumulated in the converted measurements of the redeployed Hector-SLAM. This will be further discussed in chapter 6.5. By comparing figure 7.12a with figure 7.18a one can clearly see that the maps have different orientation, which is caused by that they had a different initial pose relative to each other when the maps were generated.

5.2 Kinetics

In (Ueland, 2015) a model for the CS Saucer was developed. The model was used in the simulation model developed in (Ueland, 2016). The model was also used to develop the observer that was implemented into the control system. In (Fossen, 2011, Eq. 6.1), a 3DOF maneuvering equation for a surface vessel is given as:

$$M_{RB}\dot{v} + C_{RB}(v)v + M_A\dot{v}_r + C_A(v_r)v_r + D(v_r)v_r + Dv_r + g(\eta) = \tau_{external} \quad (5.6)$$

Where:

- v is the Body-fixed velocity vector, describing the speed in surge, sway and the rate of turn in yaw
- v_r is the Body-fixed velocities relative to local current in surge, sway and yaw
- M_{RB} and M_A is the vessel inertia matrix for the mass of the vessel and added mass respectively
- $C_{RB}(v)$ and C_A is the vessel Coriolis centripetal matrix for the rigid body and added mass respectively
- $D(v_r)$ is the nonlinear damping matrix
- D is the linear damping matrix
- $g(\eta)$ is the vector containing the gravitational/buoyancy forces and moments
- $\tau_{external}$ is the external forces acting in surge sway and yaw, excluding those already mentioned

There are no hydrostatic restoring forces present in surge sway or yaw, which means that $g(\eta) = 0$. Because of the assumption of no environmental forces present in the lab, the value for $tau_{external}$ will only depend on the thrust force generated by the propellers. In (Ueland, 2015), the resulting system matrices was found to be:

$$M = M_{RB} + M_A = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & 0 \\ 0 & 0 & Iz - N_{\dot{r}} \end{bmatrix} \quad (5.7)$$

$$C(v) = C_{RB} + C_A = \begin{bmatrix} 0 & -1.5mr & 0 \\ 1.5mr & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (5.8)$$

$$D = \begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & 0 \\ 0 & 0 & N_r \end{bmatrix} \quad (5.9)$$

$$D(v) = \begin{bmatrix} X_{u|u|}|u| & 0 & 0 \\ 0 & Y_{v|v|}|v| & 0 \\ 0 & 0 & N_{r|r|}|r| \end{bmatrix} \quad (5.10)$$

Note that in the general case, the added mass and the damping matrix will depend on frequency. However in this thesis, it is assumed constant for all frequencies. The numerical values used for the CS Saucer in the equation above are the same values that were used in (Ueland, 2016).

Chapter 6

Guidance Navigation and Control

6.1 The Path Planner

To generate a path, the system utilizes the A* algorithm. This A* algorithm is an improved version of the one which was developed and tested in (Follestad, 2016). One of the modification lets the node connections span more than one cell. This was done by increasing the number of neighboring nodes that are investigated when evaluating a node in the open-list. For the standard A* algorithm, the neighboring nodes are the 8 nodes. In this thesis the A* is modified to consider 32 neighboring nodes, which all connects to the parent node with a different angle. This results in a smoother path, since the path may be planned using a broader spectre of possible angles between nodes. This allows an angle incrimination of $\frac{360^\circ}{32} = 11.25^\circ$ instead of $\frac{360^\circ}{8} = 45^\circ$. For the A* algorithm implemented in (Ueland, 2016), the operator could chose to increase the number of neighboring nodes to 54. This means that A* algorithm used in this thesis can be considered a downgrade in comparison with the path planner developed in (Ueland, 2016).

The A* path planner is combined with a scheme for inflating the map. This will make the path keep a safety distance from the walls. A scheme for inflating the map locally has also been implemented, which is used for the re-planning of the path. The inflation of the map will be further discussed in chapter 6.6. The scheme for re-planning the path will be presented in chapter 6.7.

When the vessel reaches the desired position, the the only control action that is taken is to hold the desired heading. It is assumed that position keeping would not be needed, since a new goal would be chosen not long after the goal is reached. The goal is defined as a circle with a radius of X, and is considered reached when the lookahead point is equal to the goal node.

6.2 LOS Steering law

In order to follow the path, a Line-of-Sight (LOS) Lookahead-Based steering law as presented in (Fossen, 2011, Ch. 10.3.2) has been implemented. This steering law uses a point on the path in front of the vessel to calculate a desired heading that

aims the vessel towards the path. Figure 6.1 illustrates the LOS guidance concept.

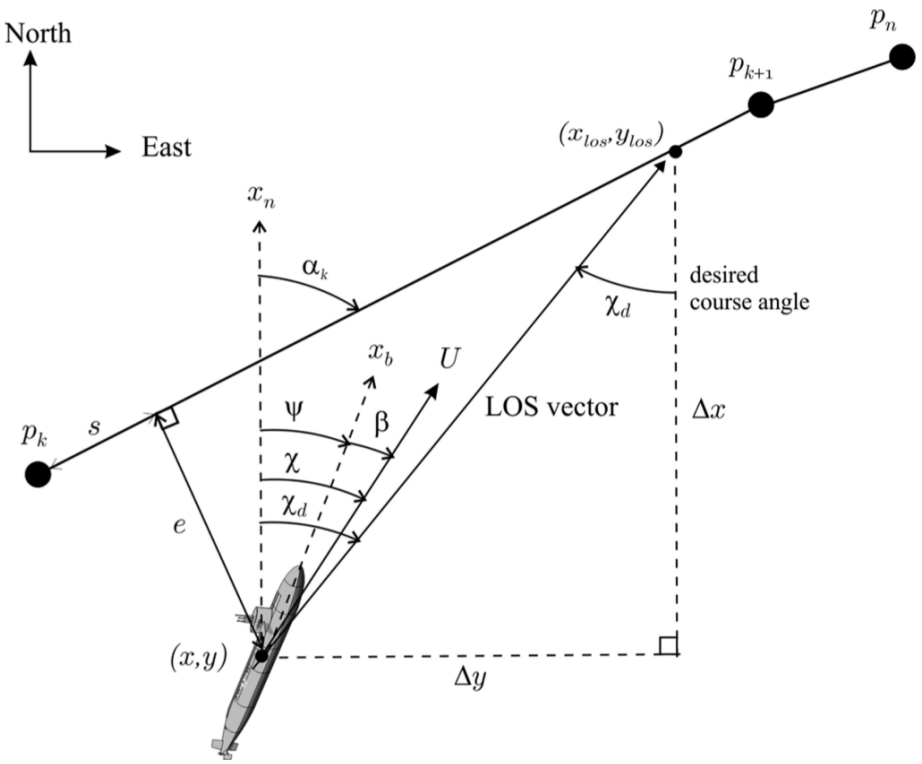


Figure 6.1: An illustration of the LOS-guidance and the parameters used in the calculations, courtesy of (Fossen, 2011)

For the lookahead-based steering the desired course angle is separated into two parts, as can be seen in equation 6.1:

$$\chi_d(e) = \chi_p + \chi_r(e) \tag{6.1}$$

Here, $\chi_d(e)$ is the desired course angle, χ_p is the angle of the path in the Basin-relative reference frame and is determined by:

$$\chi_p = \alpha_k = atan2(y_{k+1} - y_k, x_{k+1} - x_k) \tag{6.2}$$

Where $P_k = [x_k, y_k]^T$ is a point on the path and $P_{k+1} = [x_{k+1}, y_{k+1}]^T$ is a point further ahead on the path. α_k is the tangential angle of the path. p_k, p_{k+1} and α_k can be seen in figure 6.1. $\chi_r(e)$ is the velocity-path relative angle, which ensures that the velocity is directed directed towards a point on the path that is located a lookahead distance $\Delta > 0$ in front of the vessel. $\chi_r(e)$ is determined by:

$$\chi_r(e) = arctan\Big(\frac{-e(t)}{\Delta}\Big) \tag{6.3}$$

Where $e(t)$ is the cross track error of the vessel relative to the path. As it can be seen in figure 6.1, is defined as the error between the vessel and the path, that is normal to the path. The cross track error is found by rotating the vessel position from basin-relative frame into path-relative frame, as demonstrated in the following equations:

$$\varepsilon = R_p(\alpha_k)^T([x(t), y(t)]^T - [x_k, y_k]^T) \tag{6.4}$$

Where

$$R_p = \begin{bmatrix} cos(\alpha_k) & -sin(\alpha_k) \\ sin(\alpha_k) & cos(\alpha_k) \end{bmatrix} \tag{6.5}$$

Which gives:

$$\varepsilon = \begin{bmatrix} s(t) \\ e(t) \end{bmatrix} = \begin{bmatrix} [x(t) - x_k]cos(\alpha_k) + [y(t) - y_k]sin(\alpha_k) \\ [x(t) - x_k]sin(\alpha_k) + [y(t) - y_k]cos(\alpha_k) \end{bmatrix} \tag{6.6}$$

Equation 6.3 can also be interpreted as a saturating control law:

$$\chi_r(e) = arctan(-K_p e(t)) \tag{6.7}$$

Which makes the lookahead-based steering law equivalent to a saturated proportional control law, where K_p is the proportional gain. One could also introduce integral action into this control law. Integral action would be particularly useful for under actuated vessels under the influence of an ocean current and nonzero side-slip angles. Currents are not present in the environment of this thesis, and (Fossen, 2011, Ch. 10.3.2) states that an integral term should only be used when a steady-state off-track condition is detected. Because of these reasons integral action was not included in the steering law. The resulting steering law can be written as:

$$\chi_d(e) = atan2(y_{k+1} - y_k, x_{k+1} - x_k) + arctan(-K_p e(t)) \tag{6.8}$$

The CS Saucer lacks a clear bow because of its round shape. The vessel is also fully actuated. Because of this the course of the vessel does not need to be determined by the heading of the vessel. In this thesis the course is controlled by regulating the velocity directly. This is done by using the course angle, and a set desired speed to determine the velocity needed in the basin-relative frame such that vessel follows the path. The velocity is given as follows:

$$V_{xyd} = \begin{bmatrix} v_{xd} \\ v_{yd} \end{bmatrix} = \begin{bmatrix} v_d \cdot cos(\chi_d) \\ v_d \cdot sin(\chi_d) \end{bmatrix} \tag{6.9}$$

6.2.1 Waypoint Switching Mechanism

In order to progress along the path, a simple waypoint switching mechanism has been implemented. The switching mechanism defines a circle radius around the vessel. If the lookahead-point P_{k+1} is inside this circle, then the new lookahead-point will be P_{k+2}. From this one can see that the lookahead-point in this thesis is discrete. P_{k+1} and P_{k+2} are neighboring nodes on the path, and becomes the new vector pair that determines α_k, as seen from equation 6.2.

6.3 The Collision Avoidance Scheme

6.3.1 The Modified Dynamic Window Algorithm

The Dynamic Window algorithm implemented in the navigation system, is a modified version of the original Dynamic Window that was presented in chapter 3.2.

In the original DW algorithm, the trajectories were described by circular arcs, described by constant values for the vessel speed and rate of turn. In this thesis, the trajectories are estimated using the x- and y- velocity of the vessel. This is done by looking at the reachable accelerations in x- and y-direction, and holding the acceleration constant for a given time interval. By using several smaller time intervals, the curvature for a given trajectory can be described by the equations:

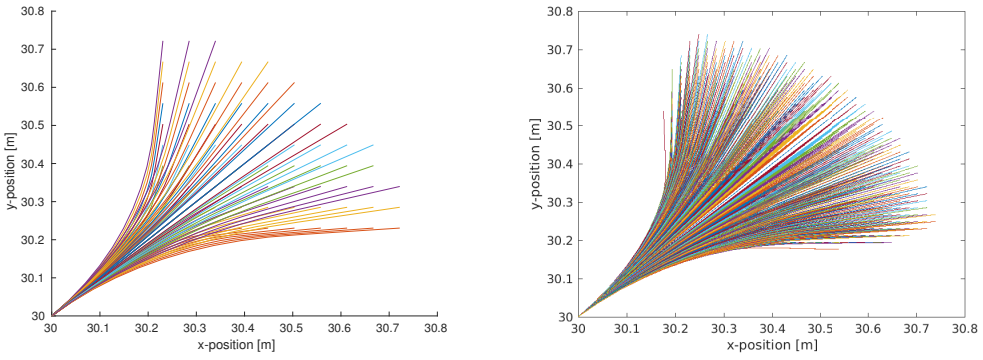
$$V_{traj}(t) = \hat{V}_{xy} + a_{can} \sum_{t=k}^{\Delta t} t \tag{6.10}$$

Where $V_{traj}(t)$ is the velocity along the trajectory, $\hat{V}_{xy}$ is the estimated velocity, a_{can} is the constant candidate acceleration, and t describes a small time step. Where a point along the trajectory can be described as:

$$P_{trajectory}(t) = P_{xy} + \sum_{t=k}^{\Delta t} V_{traj}(t)t \tag{6.11}$$

Where $P_{trajectory}(t)$ is a point on the trajectory, and P_{xy} is the initial vessel position. Δt defines the time interval with constant acceleration.

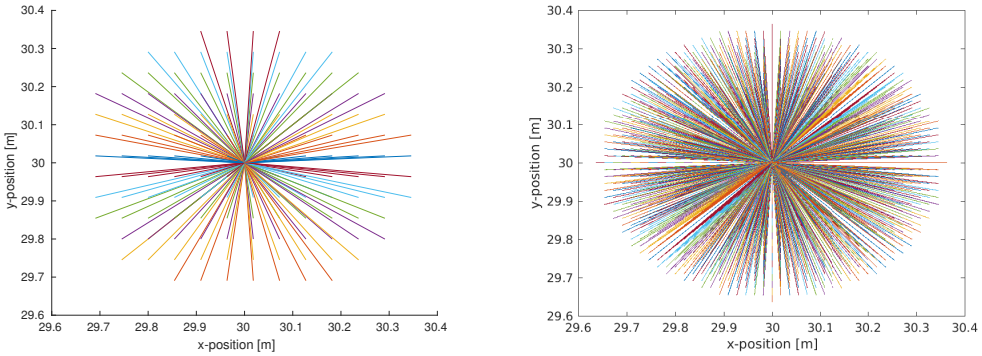
The trajectories are calculated using a set of candidate velocities, estimated from a set of candidate accelerations which will be further described in chapter 6.3.3. The trajectories are estimated for every velocity in the set of candidates. The number of generated trajectories is dependent on the number of candidate accelerations. From figure 6.2 and 6.3 it can be seen that the number of orientations the generated trajectories can achieve is dependent on the number of candidate acceleration steps, the initial velocity and the given time interval. All the generated trajectories are evaluated in order to determine which of them are dynamically feasible and which trajectory will be the optimal.



(a) The trajectories generated while using the same parameters as were used in the experiments

(b) The trajectories generated when the number of candidate acceleration steps is increased

Figure 6.2: The trajectories generated for the vessel when the initial is set to be $[0.2, 0.2, 0]^T$[m/s]



(a) The trajectories generated while using the same parameters as were used in the experiments

(b) The trajectories generated when the number of candidate acceleration steps is increased

Figure 6.3: The trajectories generated for the vessel when the initial is set to be $[0, 0, 0]^T$[m/s]

After the acceleration time interval is finished, the trajectory is extended by holding the velocities constant for a set time interval. This can also be seen in the figures above. This is done to force the trajectories a little further apart, and ahead. This will help diversifying the trajectories from one another when they are evaluated to determine the optimal path. Note that in figure 6.3, there is no initial speed, so there will be no curvature generated in the acceleration phase.

Calculating the Minimum Distance Corresponding to Each Trajectory

The biggest change made compared to the original DW algorithm lies in the distance function. In the original dynamic window algorithm, the distance function determines the distance to the nearest obstacle that lies on the trajectory curvature. The modification done to the distance function is that it checks an area around the endpoint to find the minimum distance to nearby obstacles. As an extra safety measure, the middle point along the trajectory is also checked in this way. An illustration of the resulting search areas can be seen in figure 6.4.

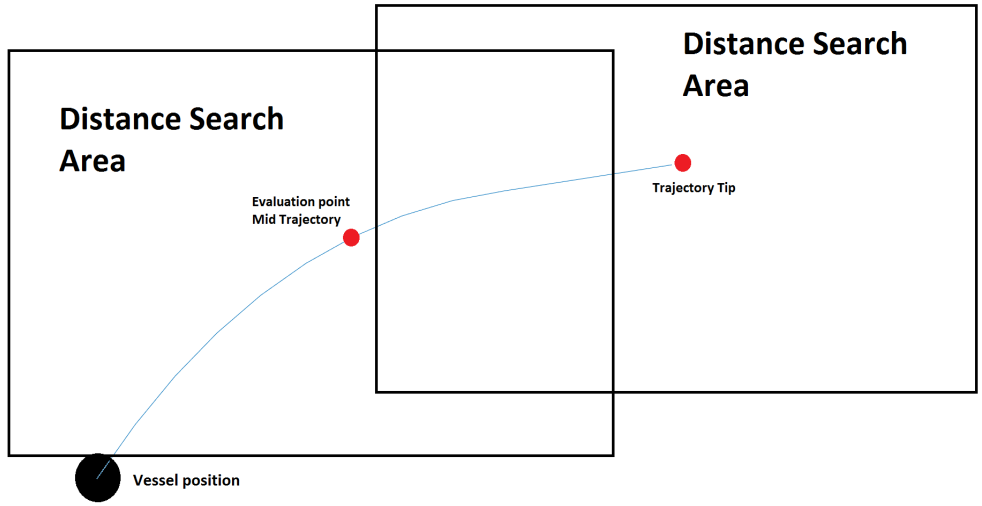
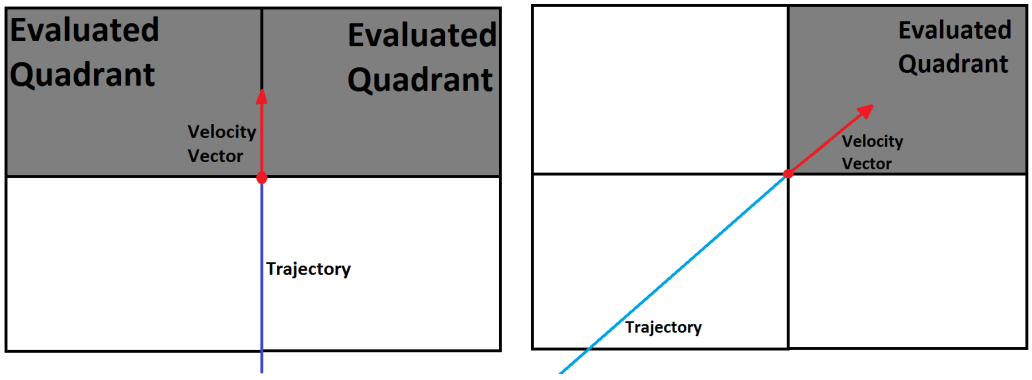


Figure 6.4: A very rough illustration of the two distance search areas that are evaluated in the distance function

A square area is defined as the search space, where the evaluated point on the trajectory is the center of the area. In this search area, the distance from the center to the various obstacles within the search space is calculated. This information is processed in two steps. The first step is to find the absolute minimum distance between the evaluated point and an obstacle. If there is no obstacles in the area, the minimum distance is set to a constant that approximately corresponds to the maximum achievable distance in the search area.

The second step processes the distance information to find the absolute minimum distance in the approximate velocity direction. The predicted velocity at the tip of the trajectory is used to determine which quadrant of the search area should be used to determine the minimum distance. A rough illustration of this can be seen in figure 6.5.



(a) A situation where two quadrants are evaluated

(b) A situation where the direction constraint is satisfied

Figure 6.5: An illustration of the distance functions ability to evaluate distance information in the velocity direction

As can be seen in the figures, there are some constraints and simplifications done

to achieve this. The general rule is that only the two quadrants that are "in front" of the velocity direction needs to be evaluated, as seen in figure 6.5a. This way obstacles that are behind the vessel at this predicted point will not be considered when calculating the minimum distance. This is a big simplification in comparison to what could have been done to reduce the search area. One could for example have defined a cone shaped area corresponding to each of the velocity directions, where the distance to obstacles within the cone shaped area would have been evaluated. However, if the velocity satisfies a constraint regarding the velocity direction, the search area will be reduced to only one quadrant as can be seen in figure 6.5b. To achieve this, the velocity must satisfy the equation:

$$|v_x| \geq V_i(v_x, v_y) \quad \wedge \quad |v_y| \geq V_i(v_x, v_y) \tag{6.12}$$

Where $V_i(v_x, v_y)$ is a function of v_x and v_y and sets the constraint relative to the magnitude of the velocity. V_i is determined by the equation:

$$V_i(v_x, v_y) = C_i \cdot \sqrt{v_x^2 + v_y^2} \tag{6.13}$$

Where C_i is a tuning parameter, in order to approximately set the velocity directions which determines what velocity directions that is eligible for a reduced distance search area.

From this it can be stated that the velocity directions that satisfies this constraint has a clear advantage over the velocity directions that are close to the axes. This often leads to that the vessel will avoid obstacles at an angle that satisfies equation 6.12.

From all this one gets two values for the minimum distance. In order to combine these values into a resulting distance function, the values are weighed and added together as can be seen in the equation:

$$dist(v_x, v_y) = d_v w_v + d_t w_t \tag{6.14}$$

Where d_v is the minimum distance relative to an obstacle in the velocity direction and d_t is the minimum distance in all quadrants of the search area. The weight constants must satisfy the equation:

$$w_v + w_t = 1 \tag{6.15}$$

Since d_v only represents the minimum distance to obstacles in one or two quadrants of the search area, the full scope of the situation in regards to potential hazards will not be represented in d_v. But because of this, d_v will more likely promote trajectories that circumvents obstacles. However, these trajectories might become too risky when circumventing dynamic obstacles. In order to better evaluate if such a trajectory is safe, d_t is weighed in so that the distance function reflects the hazards of the situation better.

The reasoning for calculating the minimum distance to nearby obstacle in this manner, is to get a better understanding of the situation each trajectory will lead to. The velocity for the obstacles has not been estimated, so the extra information about potential hazards along each trajectory will help the maximization function 6.18.

The belief is that the algorithm will handle dynamic obstacles better this way, and chose trajectories that are less risky than the original DW algorithm. However, this is strongly dependant on how well the G-score function 6.18 is tuned, which will be further explained in chapter 6.3.2.

One thing to note, is that the radius of the CS Saucer is compensated for, by subtracting the Saucer radius from the resulting weighed minimum distance.

Admissible Velocities and The Dynamic Window

The velocity search space V_r, as defined in equation 3.8 is defined by determining the admissible velocities V_a and the Dynamic Window V_s. The admissible velocities are determined by evaluating the listed parameters for each of the trajectories corresponding to the velocities:

- Weighted minimum distance to the obstacles along the path
- Weighted minimum distance from the current vessel position to the nearest obstacles
- Obstacles intersecting with the trajectory and the trajectory extensions

As described in chapter 3.2, the admissible velocities are determined by the distance function. To determine if the velocity is admissible the weighted minimum distances listed above must satisfy equation 6.16 in order to be deemed safe.

$$V_a = \left\{(v_x, v_y)|\ |v_y| \leq \sqrt{2 \cdot dist(v_x, v_y) \cdot \dot{V}_b}\ ,\ \wedge\ |v_y| \leq \sqrt{2 \cdot dist(v_x, v_y) \cdot \dot{V}_b}\right\} \tag{6.16}$$

This way, the velocities that might lead the vessel into a hazardous situation will be excluded from the velocity search space. In equation 6.16, $\dot{V}_b$ is the magnitude of the maximum break acceleration, while v_x and v_y describes the velocity at the endpoint of the trajectory. One thing to note, is that in this case, the distance function also contains the information about the weighted minimum distance to an obstacle relative to the actual vessel position. This might force a whole spectrum of velocities out of the admissible velocity sets, and promote that the velocity is reduced when the vessel is very close to obstacles. However, this might not always be desired, as it may limit the vessels ability for a quick escape when it is very close to an obstacle.

In addition to the constant velocity phase already described, the trajectory is extended even further with a second constant velocity time interval. This trajectory extension is not evaluated by a distance function, but is only used to check for intersections with obstacles further along the trajectory. If there is an intersection, the velocity will be deemed hazardous, and excluded from the velocity search space. This was implemented to help the vessel in head-on situations.

The Dynamic Widow constraint is set to limit the search space to velocities that are reachable within the given time window. This is done by evaluating if the magnitude of acceleration is feasible, and that the velocity is actually achievable. This is done by enforcing the constraint:

$$\sqrt{a_x^2 + a_y^2} \leq a_{max} \ , \ \sqrt{v_x^2 + v_y^2} \leq V_{xyd} \tag{6.17}$$

Where a_{max} is the magnitude of the maximum achievable acceleration and V_{xyd} is the desired velocity. It is assumed that the desired velocity is realistically chosen, and set to be within the dynamic limitations of the vessel. The velocities that satisfy equation 6.17 will form the set of velocities V_d describing the dynamic window. From this, the search space of possible velocities, V_r, is determined as in equation 3.8.

6.3.2 Determining the Optimal Trajectory

The optimal trajectory is determined by maximizing the G-score function, in much the same way as described in chapter 3.2. However, some small changes were made to the G-score function:

$$G(v_x, v_y) = \alpha \cdot heading(v_x, v_y) + \beta(dist) \cdot dist(v_x, v_y) + \gamma \cdot speed(v_x, v_y) \tag{6.18}$$

From equation 6.18 it can be seen that the tuning parameter $\beta(dist)$ is dependent on the distance function. $\beta(dist)$ is set to be a step function as shown in the equation below:

$$\beta(dist(v_x, v_y)) = \begin{cases} \beta_1, & dist(v_x, v_y) \leq d_b \\ (\beta_1 \frac{d_b}{dist(v_x, v_y)} + \beta_2), & dist(v_x, v_y) > d_b \end{cases} \tag{6.19}$$

Where d_b is the desired bearing to obstacles along the trajectory, and the tuning constants β_1 and β_2 are set to satisfy:

$$\beta_1 \geq \beta_2 \tag{6.20}$$

By setting β_1 high, the distance term will dominate the G-score function compared to the other terms when $dist(v_x, v_y) \leq d_b$. This will ensure that $dist(v_x, v_y)$ is the most important parameter of maximization function while the distance function has a lower value than d_b. This will promote trajectories that keeps a bearing to to obstacles, corresponding to the value set for d_b.

The terms $\alpha \cdot heading(v_x, v_y)$ and $\gamma \cdot speed(v_x, v_y)$ are much the same as described in chapter 3.2. The only difference is that they are now a function of v_x and v_y. Another thing to note is that the smoothing function is not applied in equation 6.18.

The tuning parameters of the equations 6.14, 6.18 and 6.19 is listed in table D.1. One thing to note is that the parameters are tuned differently for the simulations and the model-scale experiments. This will be further discussed in chapter 6.3.4. Another thing to note is that α is set very small compared γ, β_1 and β_2. this is due to the function $heading(v_x, v_y)$ in degrees. *gamma* is set relatively high in order to promote circumvention of obstacles, instead of reducing the velocity to keep distance.

6.3.3 Determining the Dynamic Constraints

The dynamic constraints of the vessel describes break acceleration, and what velocities are reachable within a certain time interval. To determine this constraints, the

step response of the CS Saucer was evaluated. By applying the desired velocity as a step, the resulting response was analyzed. The response was analyzed from the start of the step, and up until the time constant of the step response. The time constant is the time it takes a step response to reach $1 - 1/e \approx 63.2\%$ of its asymptotic value. This was done both for the case of accelerating the vessel up to a desired velocity velocity, and bringing the vessel to a complete stop down from the desired velocity. This can be seen in figure 6.6, which depicts data from one of the tests that were used to determine the values.

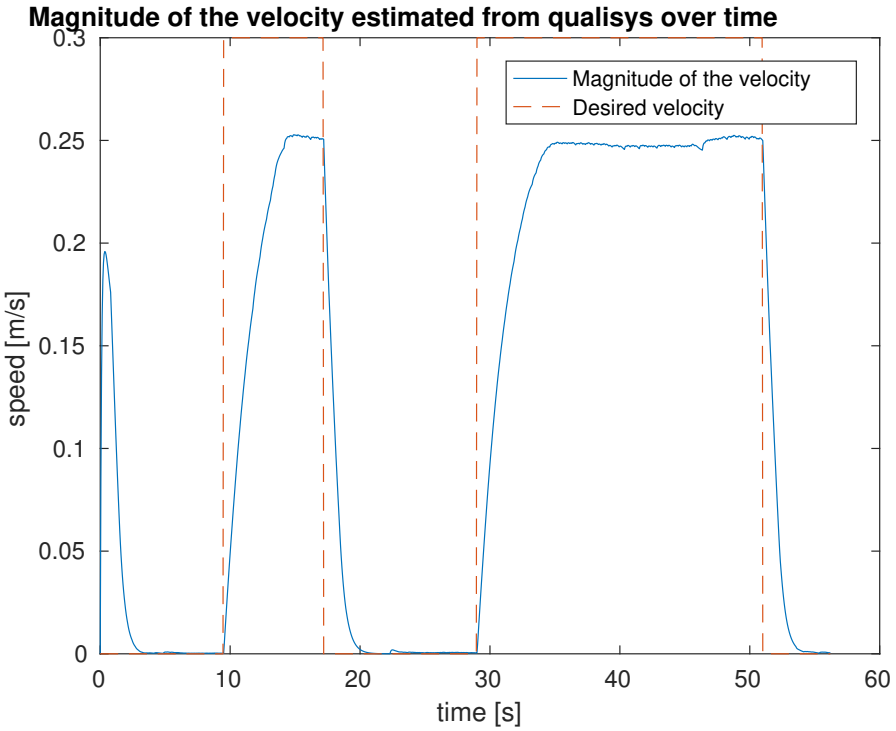


Figure 6.6: One of the tests that were preformed in order to determine the dynamic constraints set in the Dynamic Window algorithm

The acceleration were found by simplifying the problem to become linear so that only two points needed to be used. Since the vessel is omnidirectional, one can assume that one only needs to evaluate the magnitude of the velocity. This lead to that the simple equation listed below could be used:

$$a = \frac{V_{xy}(T) - V_{xy}(t)}{T - t} \tag{6.21}$$

Where T is the time constant and t is the simulation time. Equation 6.21 was used to find both the break acceleration, and the maximum acceleration reachable by the vessel. The values for these accelerations are listed in table D.1.

One thing to keep in mind, is that these acceleration estimates are very rough. They are strongly dependent on the controller gains, and how well the observer estimates the velocity. This means that the estimated acceleration constraints are only valid for the controller gains presented in appendix D, and the accuracy of the estimates is strongly dependent on how well the observer estimates the velocity.

6.3.4 Practical Considerations

In this section, some considerations regarding the limitations and performance of the modified dynamic window algorithm.

Time Usage and Time Delays

The modified DW algorithm developed in this thesis needs much more computational power than the original algorithm. The time it takes to generate and evaluate the trajectories, when the parameters are set as seen in table D.1, was found to be approximately 0.05 seconds. This corresponds to the number of trajectories seen in figure 6.2a. In a way the computational time, or the time delay will be the reaction time the collision avoidance operates in.

A collision avoidance scheme is very sensitive in regards to time delays and time lag. The modified DW algorithm is implemented in the "*exploration_pathplanner*" simulink node. In this node, both the map processing and the global path planning takes place. In other words, the ROS node is one of the most computational expensive ROS nodes in the system. This ROS node contains a real time-pacer block, that tries to force the simulink time into real-time. This means, that the implemented DW algorithm is dependent on the computational time of the entire "*exploration_pathplanner*" node, as well as the real-time pacer.

The resulting time delay could have been avoided in a fashion, and the delay has a very negative impact on the collision avoidance. The time delay could have been avoided by implementing the modified dynamic window as a separate ROS node. However, since the experiments were done at a very late stage in the semester, there was not enough time to rework the system in such a way, and collect new results. The benefits of reducing the time delay, is that it will improve the reaction time of the collision avoidance scheme.

Diversifying the Trajectories

Due to the resolution of the map, and the nature of the distance function, a problem occurred where the g-score functions where to similar. This was mostly due to the trajectories being to close together, which would lead to that the trajectories had very similar distance functions. On order to diversify the trajectories, the time for the acceleration phase was set relatively high (2 seconds). In addition, the constant velocity phase after the acceleration phase was added. This would lead to that the spacing between the trajectories increased, and that the diversity in regards to the $heading(v_x, v_y)$ and $dist(v_x, v_y)$ functions is greatly increased.

The obstacle distance search area is set to be very large, and is defined as a square of (60x60) nodes. In hindsight, one could have reduced this distance Search area to diversify the trajectories more in regards to the distance function, and save time on the computations. The reduced search are may have made the vessel a little less risk averse, and maybe even work in favor for trajectories that circumvents obstacles.

Difficult Tuning Process

As seen previously described, there has been introduced many new tuning variables for the modified DW, compared to the original DW. This lead to that the tuning process for the modified DW became quite difficult and time consuming. The distance function made the vessel more risk averse. This made it more difficult to tune the vessel to circumvent dynamic obstacles while retaining the ability to satisfyingly follow a path relatively close to static obstacles.

As can be noted from table D.1, the modified DW algorithm was tuned differently for the experiments compared to the simulations. The reason for this was that the CS saucer behaved very differently in the experiments compared to the simulations. In the experiments, the CS Saucer was a little unpredictable when accelerating and when sudden direction changes were needed. This lead to that the CS Saucer would often fail behave as predicted by the local trajectories. This lead to that the weighing function described equation 6.14 was set so the distance function was mostly described by d_t. This promotes the trajectories that holds the clearance during the whole process of the obstacle avoidance, rather than choosing trajectories that leads a more sudden change in direction to steer towards the path after the initial avoidance maneuver. This made the vessel more risk averse in the experiments compared to the simulations.

Local Minima

In some cases, the dynamic window algorithm will lead the vessel into a local minima. In most cases, the modified DW algorithm will not be able to find its way out of a local minima on its own. In such cases the path needs to be re-planned in hopes of guiding the vessel out of the local minima situation.

Dynamic Feasibility

As will be discussed in chapter 6.4.4 there will be a constant offset in the desired velocity and the estimated velocity. This offset was found to be approximately 20% of the desired value. This offset is compensated for in the modified DW algorithm, by multiplying the desired velocity by 0.8 before it is used to evaluate the trajectories.

As previously mentioned, the time-step for the acceleration phase was set relatively high. This will lead to that some of the velocity commands will require the full time interval to develop. This might even have caused some of the problems that were deemed to be because of the time lag.

An important thing to note, is that the acceleration for the vessel is not measured or estimated in this system. This means that the acceleration of the vessel is not taken into account when determining the dynamic window. In other words, this might lead to that a given candidate acceleration used in the calculations might not be reachable within the time interval. This will lead to that some of the trajectories that are evaluated are actually not dynamically feasible.

6.4 Motion Control System

6.4.1 Thrust Allocation

The vessel is equipped with three rotatable Azimuth thrusters. The orientation of the thrusters and the corresponding thrust allocation has not been altered from what was used in (Ueland, 2016) and (Sharoni, 2016). A brief summary of the thrust allocation used in (Ueland, 2016) will be presented.

The three thrusters are placed tangentially to a circle with a radius of 0.138m ($r_t = 0.138$), and with a spacing of 120 degrees. The circle shares it center of origin with the vessel. The thruster orientations are fixed. Relative to the Body-frame of the vessel the three thrusters have the following orientations: $\alpha_1 = 90°$, $\alpha_2 = -30°$ and $\alpha_3 = -150°$. An illustration of this can be seen in figure 6.7, where the black arrows shows the positive force direction. The relationship between local thrust force and body-fixed thrust force on the vessel becomes:

$$\tau = Tf \tag{6.22}$$

Where the individual thrust force vector is $f = [f_1\ f_2\ f_3]^T$ and the generalized force vector is $\tau = [X\ Y\ Z]^T$. In (Ueland, 2016) the thrust allocation was found to be:

$$T = \begin{bmatrix} 0 & sin(\frac{2\pi}{3}) & sin(\frac{4\pi}{3}) \\ 1 & cos(\frac{2\pi}{3}) & sin(\frac{4\pi}{3}) \\ r_t & r_t & r_t \end{bmatrix} \tag{6.23}$$

The desired force given from each thruster can be determined by equation 6.22 by applying the inverse of the thrust allocation matrix.

$$f = T^{-1}\tau \tag{6.24}$$

According to (Fossen, 2011, Ch. 1.2.2),), marine crafts with equal or more control inputs than generalized coordinates are deemed as fully actuated. This means that the vessel is fully actuated for the case with fixed thruster orientations.

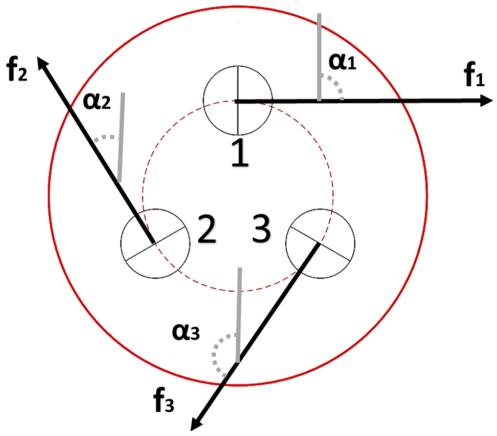


Figure 6.7: Position and orientation of the thrusters, courtesy of Ueland (2016)

6.4.2 Reference Model

A reference filter has been implemented to smooth the desired velocity as it is passed to the controller. The reference filter implemented in this thesis is a first order Butterworth filter. A Butterworth filter is a lowpass filter, that smooths or filters out high frequency components of a signal. The transfer function of the filter in the frequency domain can be written as (J. Sørensen, 2013, Ch. 6.1.1):

$$b(s) = \frac{\omega_c}{s + \omega_c} \tag{6.25}$$

Where ω_c is the cut-off frequency. A drawback of using a lowpass filter to smooth a signal, is the time delay it introduces to the resulting signal. To solve this, the butterworth filter was tuned to give a fast response. A satisfying value for the cut-off frequency was found to be $\omega_c = X$. This gave a fast response, while still smoothing the velocity demand in a satisfactory manner. In figure 7.22 the filtering can be seen during an experiment.

6.4.3 Observer

The observer used in this thesis is a non-linear passive observer. This observer was developed for the CS Saucer in (Ueland, 2016). No changes has been made to the non-linear passive observer in this thesis, other than that the tuning gains has been changed. For further reading on the theory and implementation of the observer, the author refers to (Ueland, 2016).

As stated in chapter 6.2 and chapter 6.3.1, the vessel navigates only by the use of velocity commands. It is the non-linear passive observer that estimates the velocity of the vessel. In order for the velocity to make sense in a control perspective, the observer needed to be re-tuned. Satisfying values for the tuning parameters were found to be:

$$T = diag(0.1, 0.1, 0.1) \tag{6.26}$$

$$K_2 = diag(x, y, z) \tag{6.27}$$

$$K_3 = diag(0.2, 0.2, 0.15) \tag{6.28}$$

$$K_4 = diag(5, 5, 0.5) \tag{6.29}$$

This tuning made it so that the modeled dynamics of the vessel has a bigger role when estimating the velocity. This has its drawbacks, since there are still some uncertainties regarding the dynamical model (Ueland, 2016). Another drawback with this approach is that the thrusters are unreliable, especially when the thrust direction changes (Sharoni, 2016). This might lead to that the velocity estimates does not reflect the actual velocity in some cases.

6.4.4 Velocity- and Heading- Controller

A PD-controller has been implemented to control the heading of the vessel, while a P-controller has been utilized for the speed.

Velocity Controller

The P-Controller for the velocity determines the desired control force based on the error between the actual velocity and the desired velocity. This error is multiplied by a gain vector K_{pv}, which makes the controller proportional to the error in velocity. A proportional controller on its own could in a general case lead to oscillations, overshoot and a steady state offset.

To dampen the controller one could introduce a derivative term. For this case however, that would require that the acceleration of the vessel was known. In the current state the system does not estimate the acceleration of the vessel, nor does it have any means of measuring it. However, the controller was tuned for having an offset in regards to the desired velocity, which circumvented a potential overshoot, and oscillations around the desired speed in a satisfactory manner.

An integral term would have compensated for the offset. Since the vessel needs to react fast, and often changes its course, a integral term might have lead to undesirable controller dynamic. An integral term could have been applied if it was limited by an anti-windup term, and tuned to react fast to changes in the desired velocity. However this was deemed unnecessary, since the implemented DW algorithm already takes the offset into account when determining the optimal speed, as stated in chapter 6.3.4. The offset in the demanded velocity and the actual velocity makes the proportional term produce a constant thrust, that keeps the "actual desired velocity" as can be seen in figure 6.6.

Heading Controller

The heading controller is a regular PD controller. The heading is controlled to a constant value, to stop the CS Saucer from rotating around its own axis. This will help control the vessel in a satisfactory manner, and in addition to this the laser scanner/SLAM algorithm works better for a steady heading. The desired heading is usually set to 0[rad].

The Resulting Controller

Given that the gain matrices are of the form:

$$K_p = diag(k_{pv}, k_{pv}, k_{ph}), \quad K_d = diag(0, 0, k_{dh}) \tag{6.30}$$

Where k_{pv} is the proportional gain for the velocity, k_{ph} is the proportional gain for the heading and k_{dh} is the derivative gain for the heading. The desired force in body-frame can be written as:

$$\tau_{body} = R^T(\hat{\psi}) K_p \begin{bmatrix} V_{xyd} - \hat{v}_{xy} \\ \psi_d - \hat{\psi} \end{bmatrix} - K_d \begin{bmatrix} 0_{2x1} \\ \hat{r} \end{bmatrix} \tag{6.31}$$

Where $R^T(\hat{\psi})$ is a rotation matrix, that rotates the force from basin-relative frame into body fixed frame. V_{xyd} is the desired velocity along the x- and y-axis, $\hat{v}_{xy}$ is the estimated velocity vector along the x- and y- axis in the basin-relative frame, and $\hat{r}$ is the estimated rate of turn in yaw.

6.5 Sensor Fusion

This chapter will present the sensor fusion between the converted Qualisys position and the converted Hector-SLAM position. Both the positions have been fully converted to the basin-relative frame. The conversion is described in chapter 5.1.4. The sensor fusion is responsible for fusing the two position estimates into a more robust position signal, and detect loss of signal. If both of the position signals are lost, the motors will be stopped.

6.5.1 Central Limit Theorem

The converted measurements from Qualisys (P_q) and Hector-SLAM (P_H) were combined by applying the Central Limit Theorem. The Central Limit Theorem uses the measurement noise of the signals, in this case σ_q^2 and σ_H^2 to combine the measurements:

$$P_F = \sigma_F^2(\sigma_q^{-2}P_q + \sigma_H^{-2}P_H) \tag{6.32}$$

Where P_F is the fused signal, and σ_F^2 is the noise of the combined signal, defined as:

$$\sigma_F^2 = (\sigma_q^{-2} + \sigma_H^{-2})^{-1} \tag{6.33}$$

Where σ_q^2 and σ_H^2 are the noise from the Qualisys and Hector-SLAM systems respectively.

Equation 6.32 is a weighted function, and the signal with the lowest variance will be weighed the most. Equation 6.32 and 6.33 is taken from (Wikipedia, d), and theory and proofs for the central limit theorem is presented in (Bulinski and Shashkin, 2007, Ch. 3).

For the sensor fusion implemented in this system, the residuals after the non-linear passive observers are used as a measurement of the position variances. Both the position measurement of the converted Qualisys and converted Hector-SLAM are treated by a non-linear passive observer. The residual after the observer, is the error between the measured position and the position estimated by the non linear passive observer. By logging a set of the absolute values of the residuals over time, one could use the mean of these as a measure of the position variance. This way, the "variance" of the positions are estimated online. This is done for both of the position signals by using the equation:

$$\overline{|r_e|} = \sum_{i=1}^{n} \frac{|r_e(i)|}{n} \tag{6.34}$$

Where $\overline{|r_e|}$ is the resulting mean of the absolute values of the residual and $r_e(i)$ is the i'th residual in the logged set.

By using the residuals, one introduces a dependency on the accuracy of the non-linear observers. This is a drawback compared to actually calculating the variance online, which would only be dependent on the signals themselves. One could also use a simpler approach, by calculating the variance beforehand, and use constant variances in the Central Limit Theorem. However, this will lead to that changes

in the variance of the signals are not taken into account during the experiments. The benefits of using the residuals, is that it is quite a simple approach, and works as intended in the weighing function. However, it this approach will not weigh the signals as accurately against each other as using the online variances.

6.5.2 Detecting Loss of Signal and Switching Between Positions

As already stated, the sensor fusion scheme has been implemented with the ability to detect loss of signal, and switch between the position signals if necessary. A set of position measurements is logged and evaluated when checking for loss of signal. The first thing to check is the value of the current position. If it is not a number (NaN), the current position measurement will be filtered out from the logged set, and the corresponding position signal is switched out from the sensor fusion. After this, the data set is evaluated to determine if the position signals are frozen. If the set of logged positions is completely equal to each other, the signal is considered frozen, and switched out from the sensor fusion. Since the sensor fusion only consists of two position signals, the sensor fusion position signal will be equal to the remaining signal.

Of the two signals, it is the converted Qualisys signal that is most trusted. This is due to the fact that the Hector-SLAM algorithm does not handle dynamic obstacles very well, which might lead to problems in the position estimate. The scan matching method described in chapter 3.3.1, will end up failing if the map contains too many obstacles that has changed positions during the map generation, which might lead to either a jump or complete failure. In the case of a jump, the Hector-SLAM position might get a persistent offset in the position estimate, since it will start generating a new map from this position. The scan matching will then identify this area as the current position, rather than where it was located before the jump. Another problem with the Hector-SLAM position, is that it struggles in featureless areas. An example of this is a long corridor, or the MC-lab without any static obstacles. In other words, the Hector-SLAM algorithm needs static obstacles present in order to determine its position. This means that there is an additional constraint set on the Hector-SLAM position signal.

This constraint is imposed such that the converted Hector-SLAM position must not deviate more than a maximum allowed limit relative to the converted Qualisys position. This is determined by both looking at a set of logged position, and the instantaneous position. The mean of the absolute error is estimated from the data set as:

$$\overline{\Delta\eta} = \sum_{i=1}^{n} \frac{|\eta_q(i) - \eta_H(i)|}{n} \tag{6.35}$$

Where $\overline{\Delta\eta}$ must satisfy the limit:

$$\overline{\Delta\eta} \leq \overline{\Delta\eta}_{max} \tag{6.36}$$

Where $\overline{\Delta\eta}_{max}$ is the maximum allowed deviation. If $\overline{\Delta\eta}$ does not satisfy equation 6.36, the Hector-SLAM position estimate is most likely having trouble with the scan matching. As an early safeguard for jumps in the Hector-SLAM position, the instantaneous position is also evaluated. Here it is the instantaneous error that is

evaluated in much the same manner as in equation 6.36. The allowed limit for the instantaneous error is set to be a bit higher than that of the mean error.

Qualisys can also fail to provide measurements. At least four of the infrared reflectors mentioned in chapter 4.3 must be seen by at least two of the infrared cameras at all times. These conditions will often fail to hold, resulting in a frozen position measurement in a zero-order hold from the Qualisys system. It is therefore important that the converted Hector-SLAM position can be used independently. Since the Hector-SLAM position is converted using mean values as described in 5.1.3, the converted position will keep its independence as long as the mean values are not updated while the Qualisys system is out of commission.

There is always a possibility that both of the position measurements will fail to provide information. This case is not accounted for, and might happen in some cases. To be able to better handle a situation like this, dead-reckoning should have been introduced to the system. However the non-linear passive observer created in (Ueland, 2016) has not been modified to preform dead-reckoning in this thesis.

6.5.3 Refresh Rate

As stated in chapter 4.1.2 and 4.3 the refresh rate for the Hector-SLAM is around 2-10Hz, and 50Hz for the Qualisys system. The Sensor Fusion simulink node operates at a fixed time step of 0.1s, which corresponds to 10Hz. This means that the position values from the Qualisys system and Hector-SLAM will be extracted at a sampling frequency of 10Hz. The LIDAR refresh rate is set to be about 8Hz, which is not far off from 10Hz. From this it can be concluded that the time delay between the signals will be so small that there is no need to compensate for this time delay in the sensor fusion.

6.6 Map Processing

6.6.1 Map generation

In this thesis the map is assumed be known a priori, in either a global or a local sense. The already known map is created by running the complete system developed in (Ueland, 2016). This means that the map is generated by the open-source Hector-SLAM package. By adjusting the parameters in the hector SLAM launch file, the resolution of the map is set to 0.1m and the grid size is set to (256x256). The map is plotted, and visualized the same way as it was done in (Ueland, 2016), however the map is updated with new information. The visualization process needs quite a lot of processor power in order to plot the new information to the map. For the laptop utilized in this thesis, the visualization process needed approximately 0.5 seconds to be completed. This process was done for each iteration in the *"exploration_pathplanner"* node.

As mentioned in chapter 5.1.4 the Hector-SLAM ROS node is run during the experiments. This means that a map is being generated in the background. This map will potentially have a different orientation and origo compared to the map that is known in advance, as was stated in chapter 5.1.1. In order to determine the status

of each cell, (Ueland, 2016) applied the a threshold function, which can be seen in table 6.1.

Cell status	Cell- and Threshold Value
Explored and occupied	Cell value > 50
Explored and free	$0 \leq$ Cell value ≤ 50
Unexplored	Cell value = -1

Table 6.1: Cell Threshold Values

In Matlab, the a priori map is represented as a (256x256) matrix, where each cell is considered as either unexplored, occupied or free. The values denoting the status of the cell can be listed as follows:

- -1: Unexplored
- 0: Explored and free
- 1<Node value: Explored and occupied

6.6.2 Online processing

During experiments and simulations, the map is continually updated and processed online. The map is updated with new information from the raw laser scans, as well as the redeployed SLAM map. The information represented in the resulting map is processed in various steps before the path planning and obstacle avoidance algorithms are applied.

Including the Proximity Data From the Laser Scans

The laser scans are included in the map, in order to represent information about the proximity of the vessel. The laser scan is represented as [1x360] vector, where the LIDAR reference system can be seen in figure 6.8. So in order to properly represent this information in the map, one needs to rotate the laser scan to the Basin-relative frame. Since the position of the vessel is known one could use the x and y position as the "origin" of the laser scans in the map, and use the heading to rotate the laser scan. Using figure 6.8, the angle the laser scans needed to be rotated with was found to be $\psi_r = -\psi - \frac{\pi}{2}$. Using this, one could rotate each of the 360 points in the vector and include them in the map. The resulting laser points are then rounded to represent their corresponding cell/node in the map.

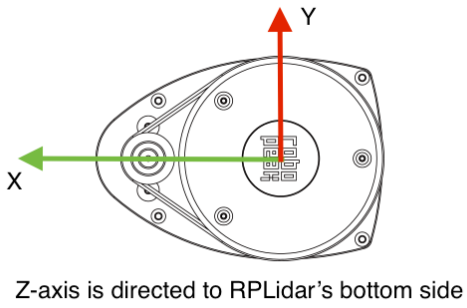


Figure 6.8: A figure showing the frame of the rpLIDAR, courtesy of Robo Peak

There are drawbacks of using the laser scan information after it has been processed in this way. As stated before, the map is discretely represented by a set of nodes. When the laser scan is converted to be a node, the accuracy of the laser will be limited by the resolution of the map. This means that the raw laser scans are much more accurate than the processed laser scans. However, the processed laser scans are much easier to work with regards to the path planning and obstacle avoidance algorithms.

Inflating the Map

To inflate the map, means to set nodes near an obstacle to be deemed as inaccessible, and thus seen as an occupied cell. This process is done for a defined inflation radius, where the free cells around the occupied cell will be labeled as occupied. This is done to define a safety radius around an obstacle. For the implemented system this is done both globally, and locally.

The global inflation is only preformed only when planning the path, and the resulting map is only used for global path planning. Different methods for local inflation is done for both re-planning, and for the local planner. The three methods and their uses, are presented below:

- **Global inflation**: The global inflation of the map is only preformed when the path is planned. This scheme inflates every explored occupied node. This will ensure that the generated path will keep a distance from obstacles.
- **Local node weighing for re-planning**: For the case of re-planning a local weighing is used to process the globally inflated map. The reasoning for this will be further discussed in chapter 6.7. The local weighing processes the map that is within the LIDAR range. For the case of re-planning the local weighing scheme takes the distance between the vessel and the obstacle into account. It uses this distance to determine a weight for the affected nodes, rather than just labelling them as occupied. Cells that are placed further away from the vessel gets a lower weight.
- **Local inflation for the local planner**: The local inflation is done in the same fashion as the local weighing. Instead of weighing the affected nodes are set as occupied. Local inflation was applied to the map before it is sent

to the modified Dynamic Window algorithm. The local inflation is preformed each iteration in the *"exploration_pathplanner"* node, and uses a computational time of approximately 0.05 seconds.

The Hector-SLAM map, and updating the existing map

The existing map might not be fully explored. In order to update the missing pieces, the information generated in the re-deployed Hector-SLAM map is used. Since the position of the vessel is known in both maps, one can use the vessel position to convert information from one map to the other. The distance between the vessel position and a given node in the re-deployed Hector-SLAM map, will be approximately the same for the corresponding node in the known map. By using the relative position between the vessel position and the node in the re-deployed map frame, one can rotate the relative position to the existing map frame.

The relative position of the node is rotated and converted in the same way as described in chapter 5.1.3. The rotation angle is still defined in the same manner, and is the heading difference between the unconverted position in the re-deployed map and the vessel heading in the known map. For this case it is not the mean values that are used, but the instantaneous errors, and the offset compensation uses only the initial offset to compensate for the offset in the origin. After the conversion, the corresponding node in the known map is updated if it is not already marked as explored. This updating process is done for every node within the range of the LIDAR.

However, some constraints set to hinder the known map being updated with bad information. One of the constraints is that the offset between the vessel position in the known map and the re-deployed map frame must be within a certain limit. This is enforced by comparing the offset with the initial offset. The initial offset is set as the offset between the vessel position in the known map and re-deployed map at the very first time interval. If the offset is persistently changing, it implies that the scan matching in the Hector-SLAM algorithm is failing, as it was discussed in chapter 6.5.2. If this fails to hold, the position in the re-deployed map will be wrong relative to the known map, since it is only defined by the Hector-SLAM position.

The other constraint is that neither of the Qualisys position or the Hector-SLAM position can be switched out from the sensor fusion. This is an extra safety measure, to ensure that the position in the known map is not only defined by the Hector-SLAM position.

Due to the discrete nature of the maps, this approach will introduce some uncertainties as it will not be able to represent the information in the re-deployed map with full accuracy.

Simple Simulated Dynamic Obstacles

The map can also be set to contain pre-defined simple simulated dynamic obstacles. These virtual obstacles can be applied to the map both for simulations and experiments in the basin. The virtual obstacles can be defined to be a square of different

dimensions, and set to have a constant velocity. One can also define a time interval where the velocity changes direction.

The drawback with these obstacles are that they are simple, meaning that they does not contain any dynamics other than a constant velocity. They are discretely defined in the map, and are displayed in the map if they are in the line of sight of the LIDAR. However, for testing purposes they function quite well and are very simple to implement to customize.

Simulated dynamic obstacles were originally planned to be created as a separate simulation node or an extension of the "Mapping Simulator node" developed in (Ueland, 2016), but was implemented directly into the map processing instead. More work should have been invested in this, to make it represent the situation in the MC-lab better. For instance, the time lag may have been represented better in the simulations had the obstacles been implemented as a separate ROS node.

Plans for obstacle velocity estimation which were not implemented

One of the plans that was not implemented, was to use the laser scan data to estimate the velocity of obstacles relative to the vessel. The velocity information would have been used to influence the local inflation, in such a way that the predicted position of the vessel would be deemed occupied, much the same way as the POA described in chapter 2.2.2. This would have greatly increased the situational awareness of the vessel. The relative velocity would have been estimated from a set of laser scans over time. However, estimating the full velocity vector for the obstacles would have been challenging, and might have required some sort of tracking scheme. Still, using only the relative velocity to compensate for velocities that points towards the vessel would have been a great aid for the collision avoidance system.

6.6.3 Blind Spots

In figure 4.1, one can see a cap that was introduced to the vessel. This cap was used to elevate the infrared reflectors, so that they could easier be seen by the infrared cameras described in chapter 4.3. Another reason was to get the infrared reflectors at the center of the vessel, and reduce the interference with the LIDAR scan plane. However, this particular solution will still interfere with the laser scan. As can be seen by figure 4.1, three pillars will block the view of the LIDAR at certain angles. This introduces a blind spot in the laser scan. Figure 6.9 shows a laser scan illustrating one of the blind spots quite clearly. In the slam algorithm, and when the laser scan is processed the minimum range of the laser scans are set to be the radius of the ship.

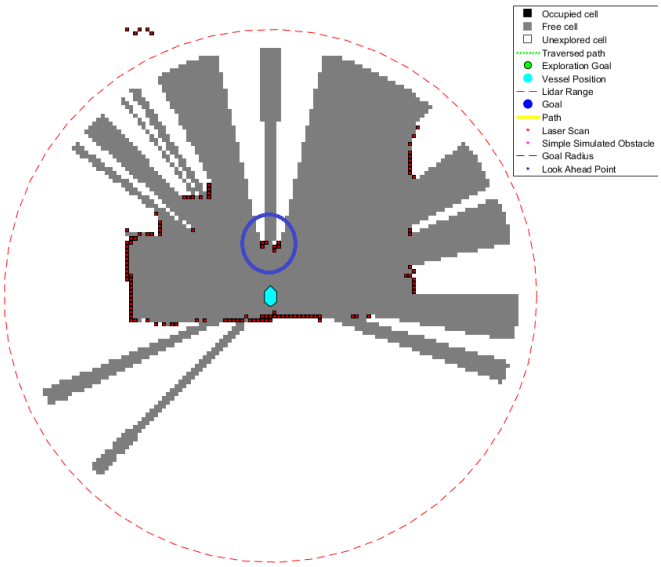


Figure 6.9: A raw laser scan of the MC-lab control room. One of the blind spots is marked with a blue circle

To compensate for this, the plan was to include the acoustic proximity sensors introduced in (Spange, 2016). However, due to time limitations this was not done. There are many ways the blind spot could have been minimized. One way to do this could be to use another solution for the placement of the infrared reflectors. Another solution could be to use four pillars attached with reflectors placed at the edge of the vessel radius. This would put the obstacles further away from the LIDAR, and reduce the blind spot. However, for the existing installation, the best solution was to twist the pillars and then fasten them in the twisted position. This would minimize the blind spot for the given installation.

The blind spot will have a negative impact on the system as a whole. The blind spots is not compensated for in any way. It will introduce uncertainties to the Hector-SLAM algorithm, as well as decrease the overall situational awareness. In a worst case scenario a blind spot might even lead to collision. However, the blind spots are relatively small, and shrinks the closer one gets to the vessel. The proximity information represents the hazards in the vicinity adequately, despite the blind spots.

6.7 Re-planning the path

Certain criteria has been introduced in regards to the re-planning of the path. These criteria are set so that re-planning will be preformed only when it is strictly needed. These criteria are based on how well the vessel follows the path, and the progress along the path. The following criteria are used to determine this:

- **Cross-track error limit**: When avoiding obstacles, the vessel will most likely deviate from the path. From chapter 6.2 it can be seen that the cross-track error becomes a measure of this deviation. In chapter 6.2.1 the switching mechanism defines a circle to determine the next lookahead point. From this it can be deduced that if the cross-track error is much larger than this circle radius, the switching mechanism will fail. This might lead to that the vessel

will hold a desired course that does not follow the path. To prevent this from happening, there has been set a limit for the maximum allowed cross-track error. When this limit is reached, the path will be re-planned form the current vessel position to the goal.

- **Along-track distance limit**: Since the tolerated cross-track error is set somewhat larger than the radius of the circle defined in 6.2.1, an additional criteria needs to be set. In this thesis, the along track distance is defined as the path-relative distance between the lookahead point and the vessel position. This means that an abnormally large along track distance $s(t)$ implies that the vessel is off track relative to the path. One possible scenario for this to occur is a collision avoidance where $e(t)$ is larger than the switch circle for quite some time, but not larger than the cross-track limit. This might result in a situation where the vessel does not switch the lookahead point along the track, resulting in that a change in the path course will not be taken into account. Consequently the LOS steering will minimize $e(t)$ and follow the course defined by the now constant lookahead point. By evaluating $s(t)$ this behaviour can be detected. By setting a re-plan criteria based on $s(t)$ the path will be re-planned to avoid the vessel going off-course.

- **Time limit**: If the progress along the path is halted, it can imply that the vessel is trapped in a local minima as described in chapter 6.3.4. Since the path lookahead point is discretely defined, one can easily use the change in the lookahead point as a measure of the progress along the path. If the lookahead point remains constant for a set amount of time, the path will be re-planned.

As mentioned in chapter 6.6.2, the map is inflated locally before the path planner is allowed to re-plan the path. This local node weighing is applied in addition to the global inflate. The reasoning for this scheme, is that it might greatly help the global path planner to avoid generating paths that will lead to a local minima for the local path planner. To mention one situation where this is of great help, is for re-planning when faced with a narrow gap. The local path planner won't allow the vessel through this gap, but the global path is generated to lead the vessel through this path. By re-planning with an additional local node weighing, the narrow path may be closed off due to the additional inflation. This will force the global path planner to find a path that does not lead through this particular narrow gap. The method for weighing the cells, as described in chapter 6.6.2, is to avoid the vessel being trapped between two or more gaps. In other words if the only routes that can be taken are through narrow gaps, the path re-planner will choose the gap that is furthest away from the vessel. There is no guarantee that the local planner will allow to navigate through this gap, and the vessel might end up in a local minima yet again. However, this scheme forces the path re-planner to try new routes if one path fails.

For the case of re-planning when off track, the local node weighing will act as an extra safety. When the vessel goes off-course relative to the path it is almost implied that the vessel has encountered a dynamic obstacle. This means that the resulting extra space between the re-planned path and eventual obstacles, will in many cases be a safer path to follow.

Chapter 7

Results

In this chapter, the results from the experiments and simulations will be presented. The result was logged in simulink, by writing the data to .m files, as well as saving the visualization of the map at important moments as a figure. Logging information this way requires quite a lot of processor power, as much data is being logged.

7.1 Sensor Fusion

The sensor fusion results corresponding to the experiment done in chapter 7.4 will be presented in this section, to showcase the sensor fusion performance.

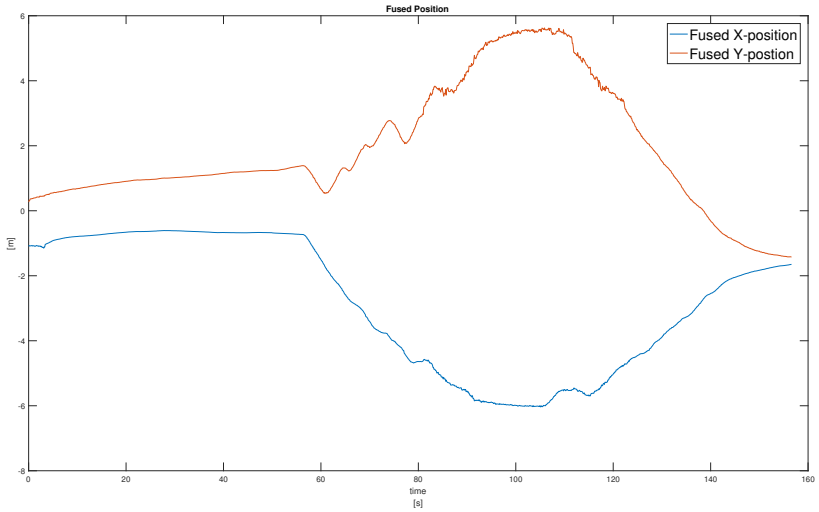


Figure 7.1: Fused position along the x- and y-axis over time

In figure 7.1, the fused position over time can be seen. Note that the measurements becomes more noisy at around 80 seconds into the experiment. From figure 7.2 one can see that the converted Qualisys position is frozen, and holds the same value for some time. This is most likely due to that the infrared cameras do not see the reflectors. From 7.3 one can see that during the frozen period, the Qualisys position is switched out from the sensor fusion. This means that the noise is actually the Hector-SLAM position noise.

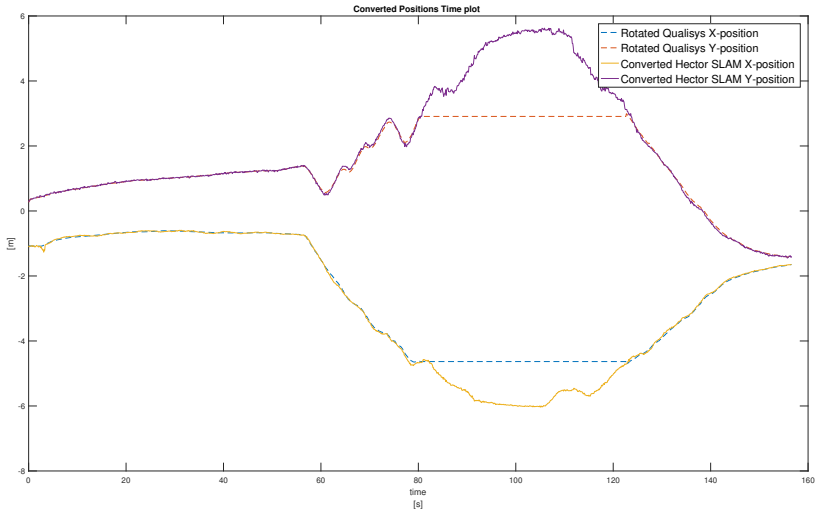


Figure 7.2: Converted Qualisys and converted Hector-SLAM position along the x- and y-axis over time

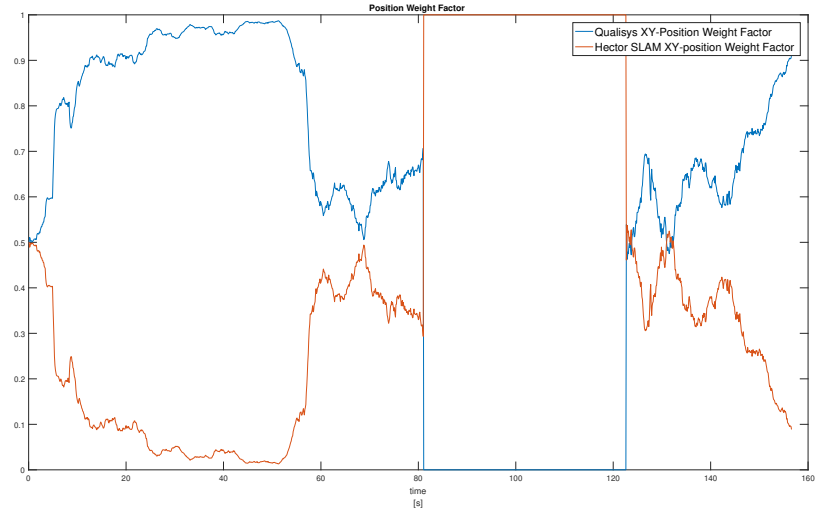


Figure 7.3: Weight, determined by the observer residuals

One can also note from figure 7.3 that the weighing function works quite well in this case. However, there are some instances where both the position estimates are available that the converted Hector-SLAM position is weighed higher than the Qualisys system. This can be seen at around 132 seconds. Normally the Qualisys system has a precision much greater than the Hector-SLAM position estimate. By studying the graph in figure 7.2 one can conclude that this is also the case for the positions at 132 seconds.

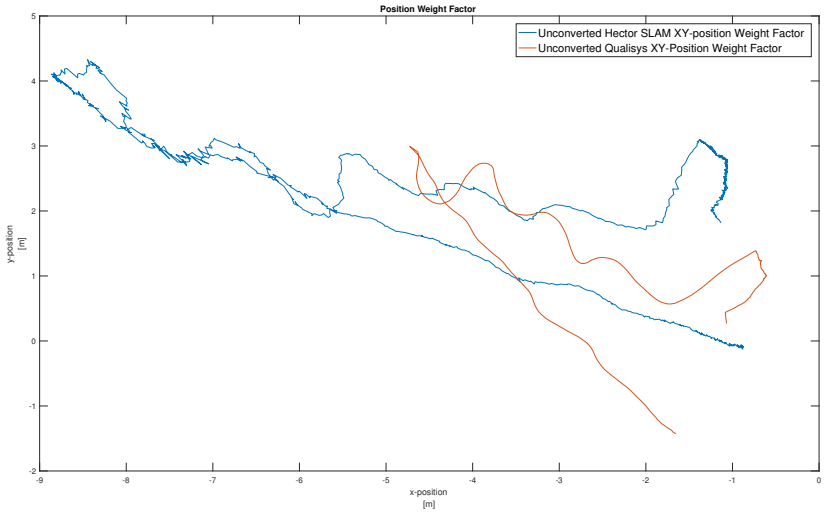


Figure 7.4: Weight, determined by the observer residuals

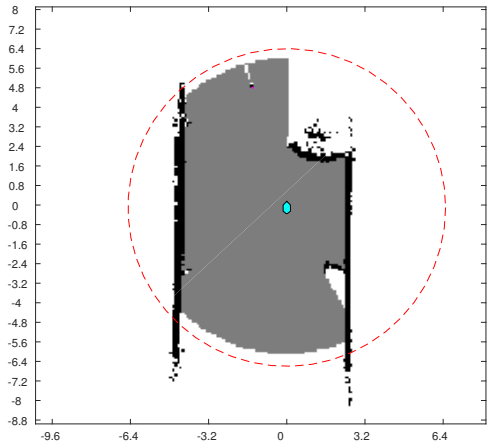
In figure 7.4, the unconverted XY-position for both of the systems can be seen. From this figure it can be seen just how different the unconverted positions are compared to one another. One thing to note is that the Qualisys system seems shorter, only because of the loss of signal, described in figure 7.2. This can be seen at position $[-4.7, 3]^T$ in figure 7.4.

7.2 Simulations

7.2.1 Dynamic Obstacle, Parameters Tuned for Simulations

As stated in chapter 6.6.2, the ability to simulate simple dynamic obstacles has been added in the online map processing. The simulated obstacles can be added both for the simulations and for the experiments.

The map implemented in the simulator can be seen in figure 7.5a. The map was generated in the MC-lab, and has a resolution of 0.1m. The legend describing hoe the information is displayed in the map is shown in figure 7.5b. The dynamic obstacle in this case, is a simulated obstacle described by (7x4) nodes with a constant velocity of [-0.1,0.1] [m/s].



(a) The Simulator map, and the initial position of the vessel

(b) Figure Legend

Figure 7.5

The complete simulation run is presented in figure 7.6. From figure 7.6a shows that the path has been generated and that the simulated obstacle can be seen at the edge of the laser scan range. From figure 7.6b the vessel can be seen keeping its distance to the static obstacle, and that the simulated osbtacle is getting closer. In figure 7.6c, 7.6d and simulate 7.6e one can see the obstacle avoidance maneuver. The vessel moves into the course of the simulated obstacle in this case, which does not satisfy the COLREGS regulations. However, if one does not take COLREGS into account, the collision avoidance was successful. And in figure 7.6f the vessel can be seen approaching the goal.

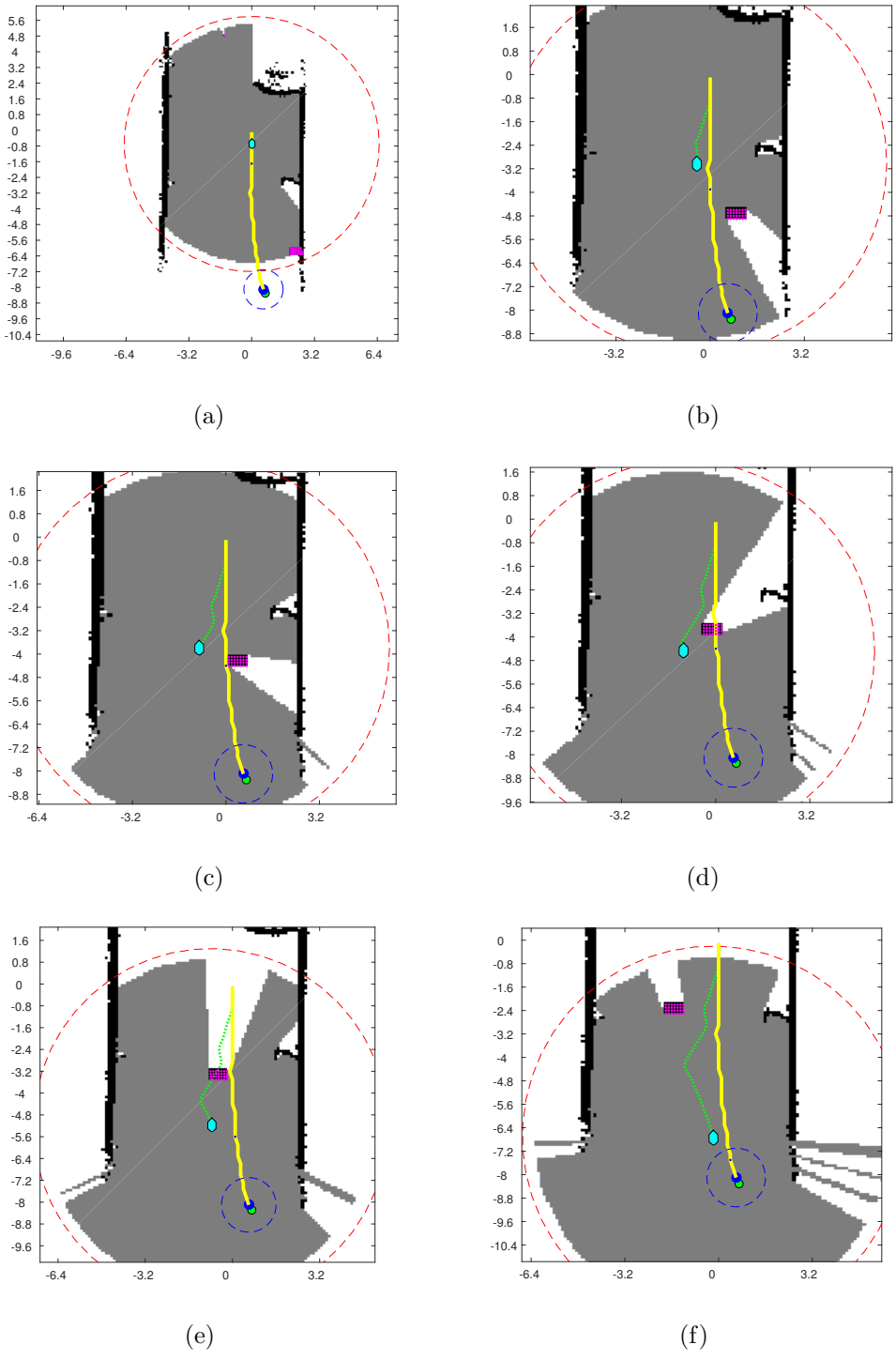


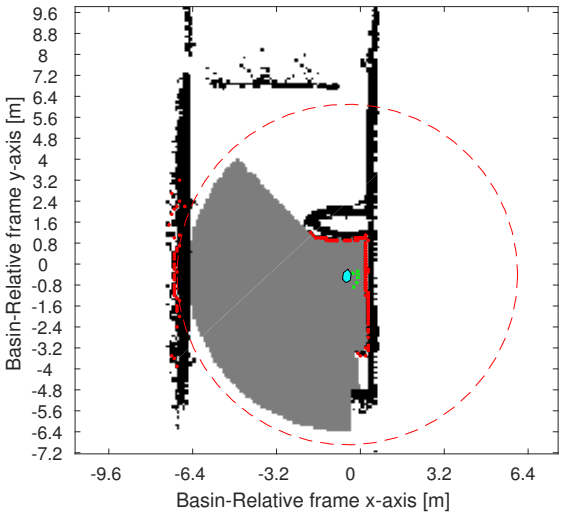
Figure 7.6: A simulation preformed with a simple simulated obstacle

7.3 Static Obstacles

In this experiment, the collision avoidance is tested against unknown static obstacles. From figure 7.7 parts of the area of operation can be seen. The basin is cut off by the use of trash bags, that are taped on to the carriage. This works as a wall, as they were registered during the mapping process, which can be seen in figure 7.8a.

Figure 7.7: A picture of the static experiment environment

Figure 7.8a shows the a proiri map the vessel is deployed in along with the initial position of the vessel. Note the axes of the figure. Figure 7.8b shows the legend of the plots.



(a) The a priori map, and the initial position of the vessel

(b) Figure Legend

Figure 7.8

The following figures depicts the complete experimental run. Note that the axes are denoted the same way as in figure 7.8a.

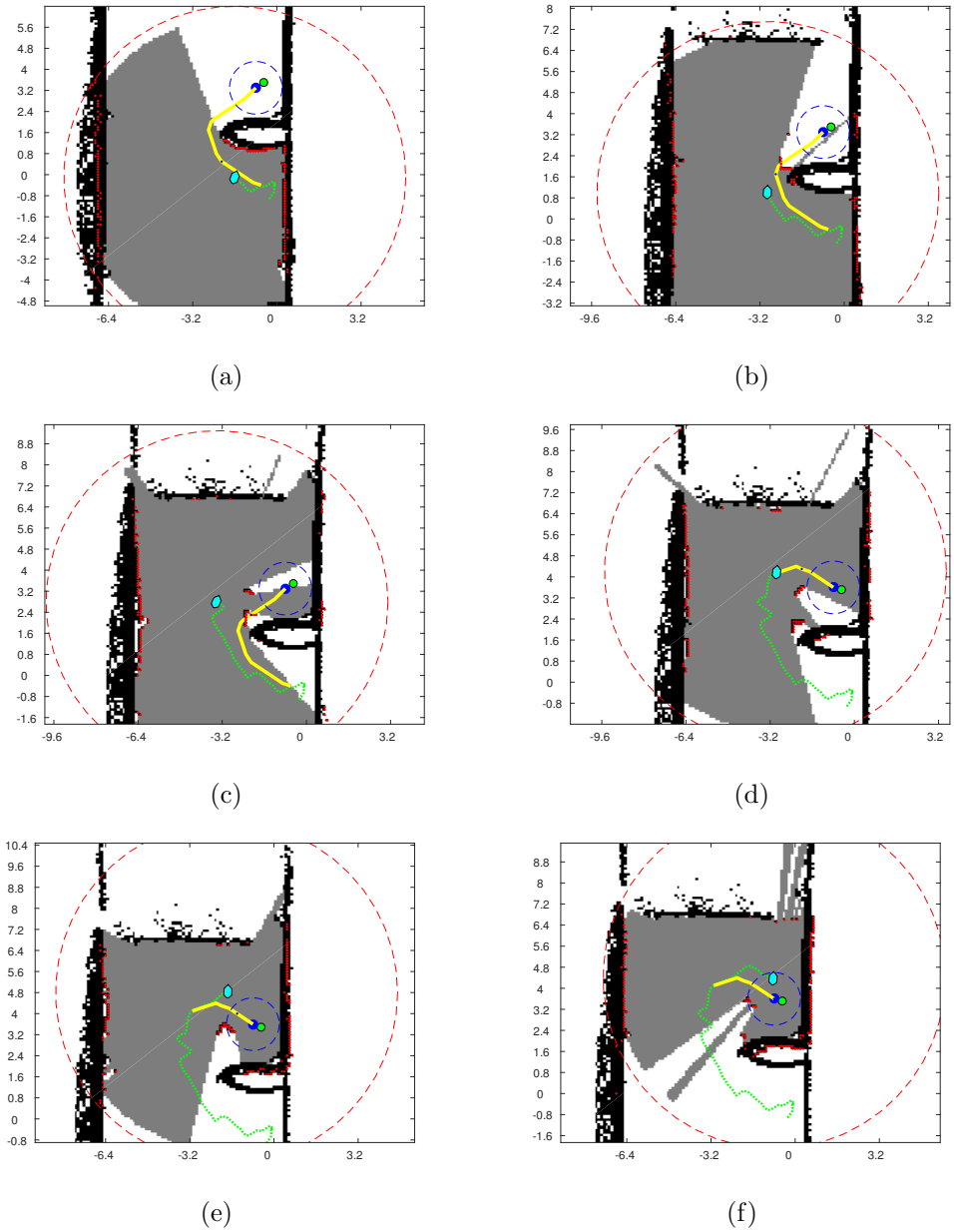


Figure 7.9: Experiment with unknown static obstacles

From figure 7.9b it can be seen that the laser scan detects one of the static obstacles, and that it is located on the path. As it tries to avoid this obstacle, it discovers another obstacle, which can be seen in figure 7.9c. This results in that the vessel is driven off-course to avoid the obstacles, and the resulting cross-track error relative to the path causes the re-planner to take action. The re-planned path can be seen in figure 7.9d, and the goal is reached figure 7.9f. Since almost the whole map is known a priori, the map is not updated with information from the re-deployed Hector-SLAM map. This can be seen by comparing figure 7.9b and 7.9f, which shows that the information regarding the unknown obstacles are only mapped locally by the laser scans.

To better illustrate the choices made by the modified DW algorithm, the optimal

trajectories generated by the modified DW algorithm is presented in figure 7.10.

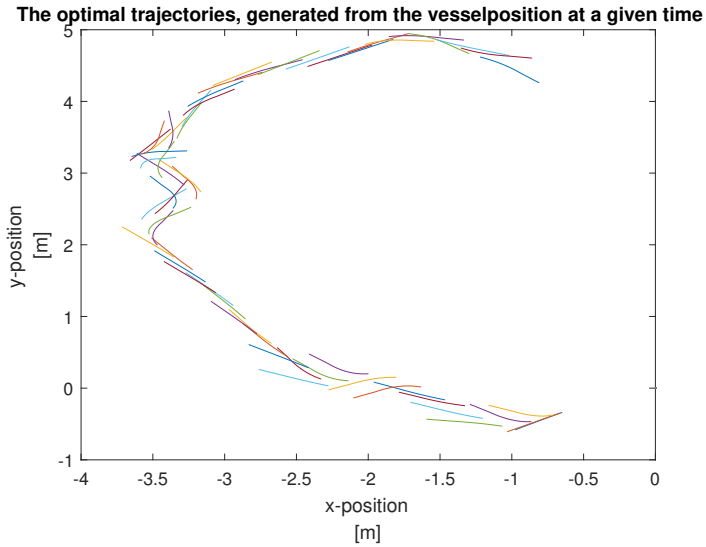


Figure 7.10: The optimal trajectories for the static obstacle experiments

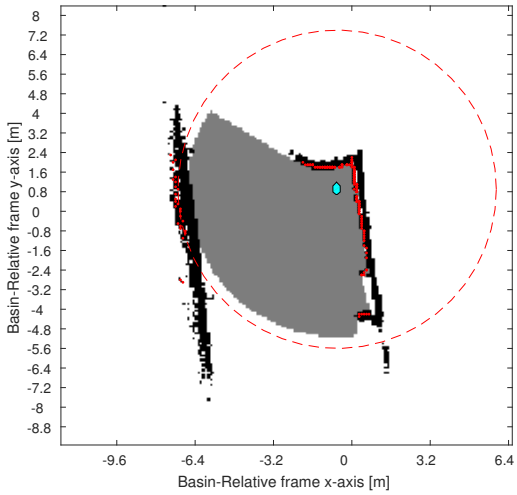
One thing to note from figure 7.10 is that one can see the local trajectory corresponding the situation depicted in figure 7.9c point right towards one of the obstacles. Or rather that the local planner tried to guide the vessel towards the gap between the obstacle in a small time window. However, the vessel steers away and tries to circumvent the obstacles, instead of going through the gap. The reason for the trajectory leading towards the gap being considered optimal might be because of a blind spot making the gap seem less hazardous than it really was.

7.4 Slowly Moving Thin Obstacle

In this experiment, the collision avoidance system was tested against a thin obstacle, while the map was not completely explored beforehand. Figure 7.11 depicts the dynamic obstacle in this experiment, which is a trash bag taped to a carriage. The carriage was pushed by hand to make the obstacle dynamic.

Figure 7.11: A picture of the thin dynamic obstacle

Figure 7.12a shows the initial map along with the initial position of the vessel. Note the axes of the figure. Figure 7.12b shows the legend of the plots.

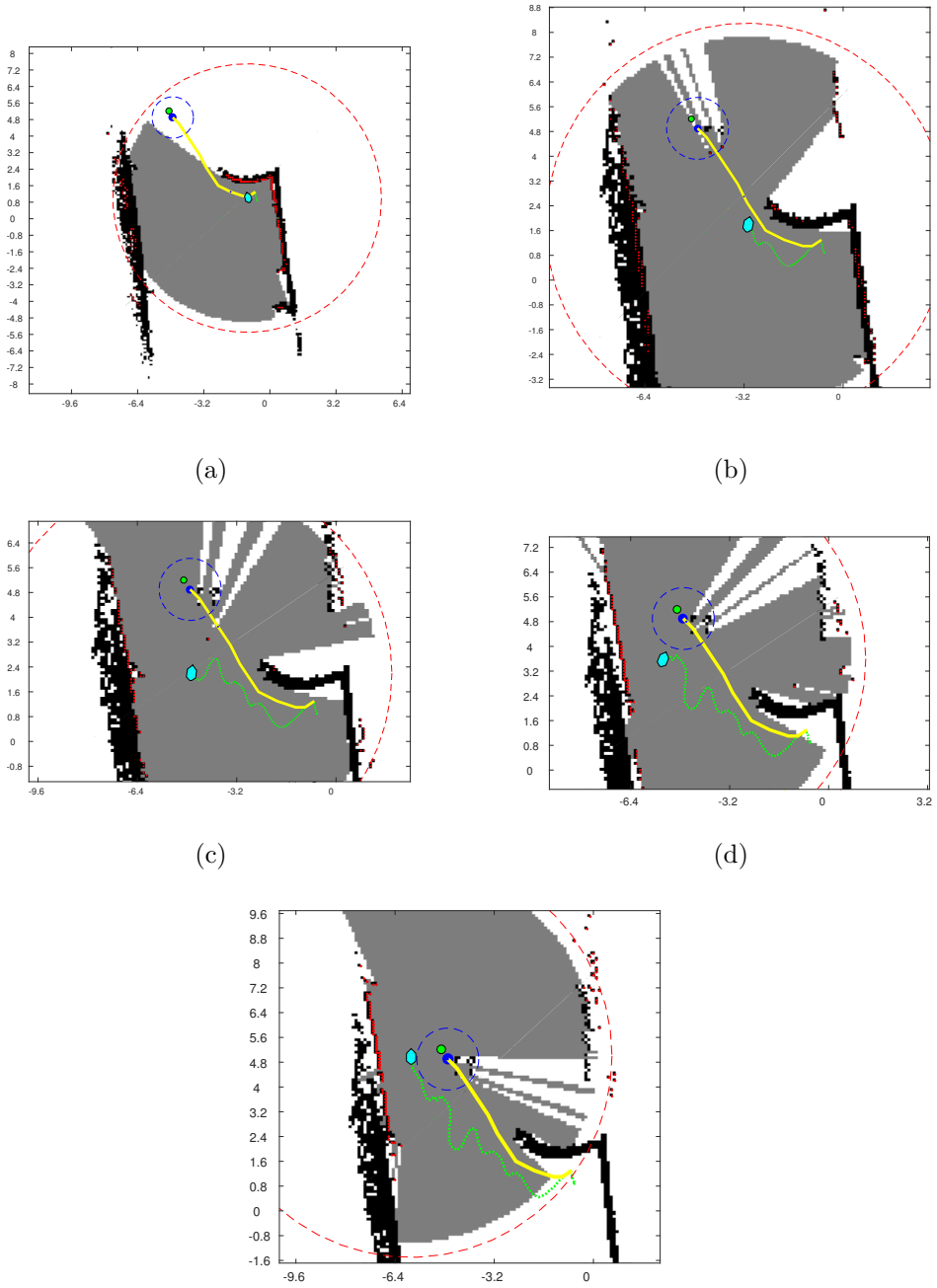


(a) The a priori map, and the initial position of the vessel

(b) Figure Legend

Figure 7.12

The full experimental run is presented in the following figures. Note that the axes are denoted the same way as in figure 7.12a.



(a) (b) (c) (d) (e)

Figure 7.13: Experiment with thin dynamic obstacle

In figure 7.13b one can see that the obstacle has been detected and is close to the goal node. In figure 7.13c only a single node of the obstacle is detected, this might be due to the a blind spot, or that the LIDAR rays are deflected away because of the angle of the bags. However, one thing to note from this situation is that the modified DW algorithm avoids the single obstacle node that is detected. In figure 7.13c the obstacle is no longer detected. However, since the obstacle was spotted outside of the a priori map, the re-deployed Hector-SLAM map had mapped some of the obstacle positions as occupied. This lead to that the map was updated with

bad information, which did not reflect the actual situation. From figure 7.13e it can be seen that the vessel fails to reach the goal without re-planning because of this.

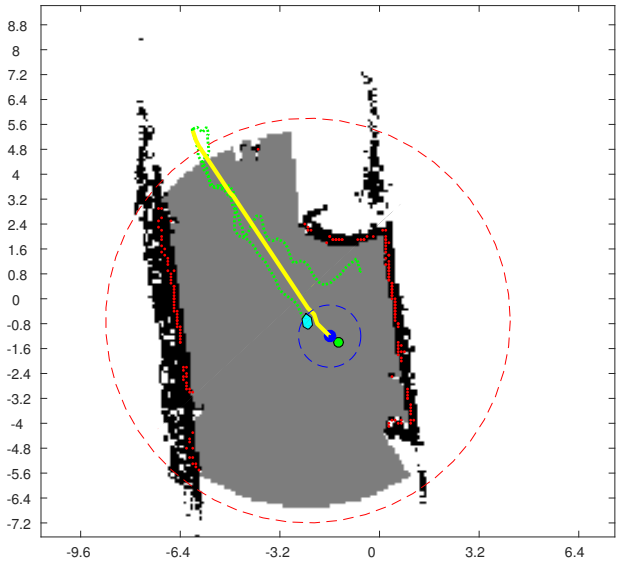


Figure 7.14: The vessel returns to the start area of the experiment

By comparing figure 7.14 with the initial map shown in figure 7.12a, one can clearly see that the map has been updated with new information gathered by the Hector-Slam algorithm.

7.5 Dynamic Obstacles

In this chapter, the experiments done with "regular" dynamic obstacles will be presented. The dynamic obstacle used in this case is shown in figure 7.15, and is one of the cardboard boxes that was utilized as a static obstacle in chapter 7.3.

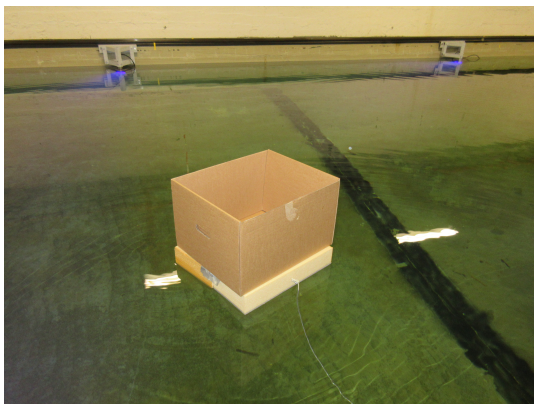
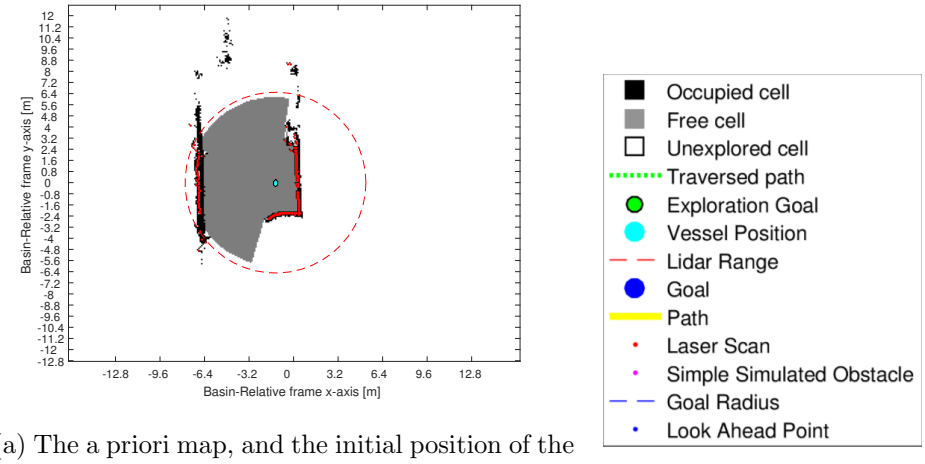


Figure 7.15: A picture of the cardboard box that plays the part of dynamic obstacle

7.5.1 Crossing situation



(a) The a priori map, and the initial position of the vessel

(b) Figure Legend

Figure 7.16

In this section, a crossing like situation will be presented. Figure 7.16a shows the map that was generated before the experimental run, which in this case is partially explored. The full experimental run is shown in figure 7.17.

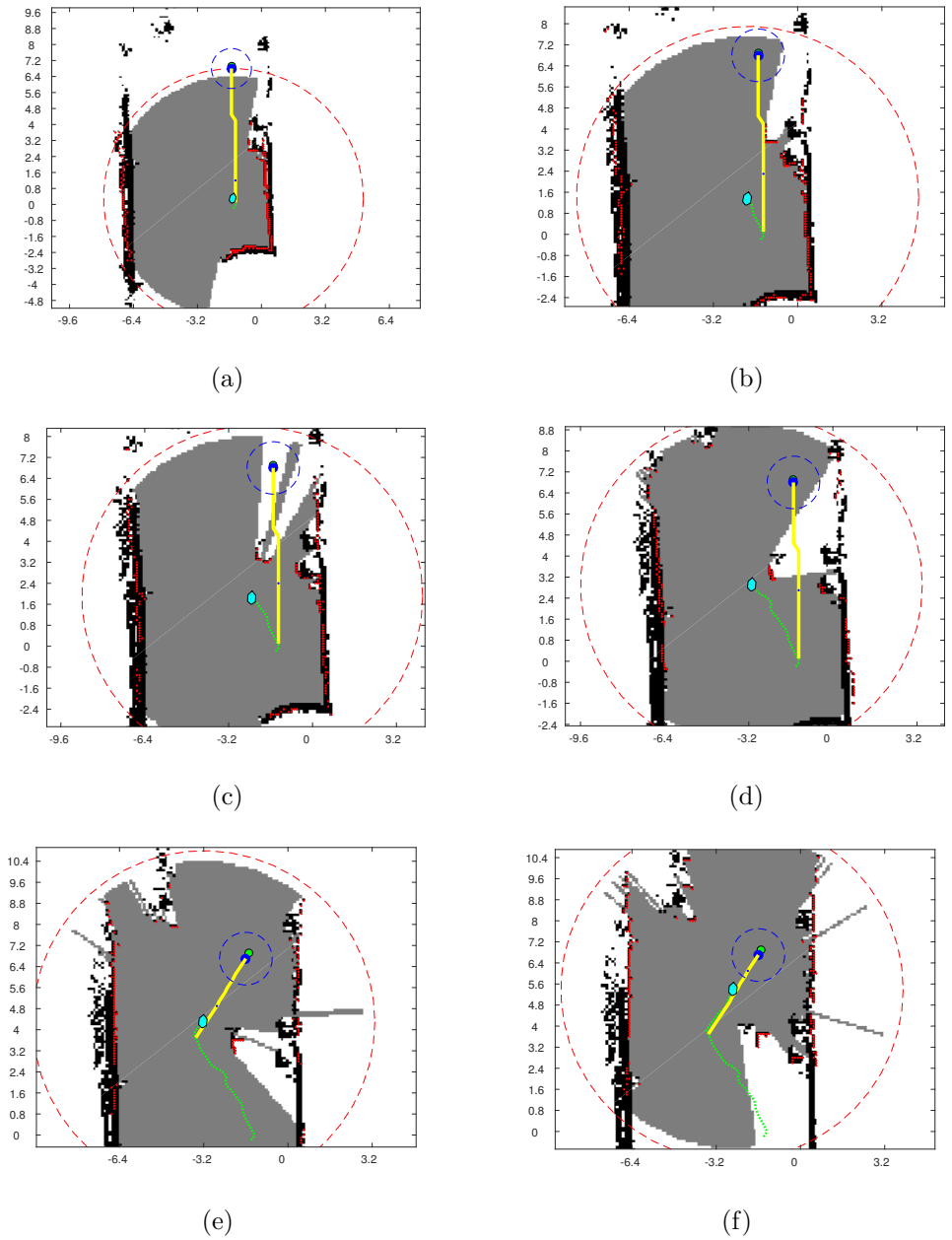


Figure 7.17: An experimental run in the MC-lab depicting a crossing situation

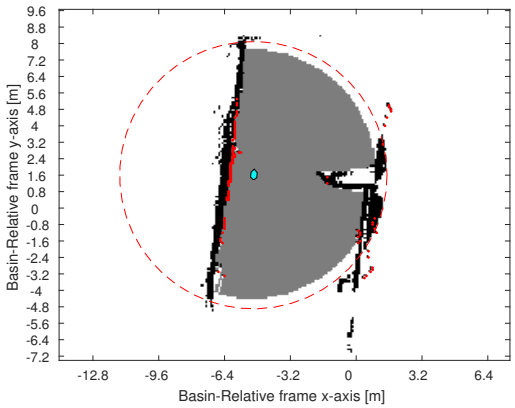
From figure 7.17a one can see that the path has been generated, and that the CS Saucer is following the path. From figure 7.17b one can see that the vessel is already reacting to the situation ahead. However, from figure 7.17c, 7.17d, one can see that the collision avoidance leads the vessel into a dangerous situation. The vessel is brought into the course of the moving obstacle, which in itself is a dangerous maneuver. However, one must note that the relative speed, or the velocity of the obstacle is not known. So if one regards each time frame as static, one can see that the obstacle maneuvers makes sense from a static obstacle perspective. From figure 7.17e and 7.17f one can see that the collision was actually avoided, and that the path has been re-planned.

One thing to note, is that the cardboard box was pushed quite hard for this case,

leading to a rapid approach and a high initial velocity. However, the obstacle has no means of generating force, so the velocity is dampened out over time. This can be seen by comparing the figures 7.17b, 7.17c and 7.17d.

7.5.2 Head On Situation

In the following run, the system is tested in a head-on situation. Figure 7.18a shows the initial position of the vessel along with the initial map.



(a) The a priori map, and the initial position of the vessel

(b) Figure Legend

Figure 7.18

The full experiment is presented in figure 7.19. In figure 7.19a and 7.19b one can see that the path is generated, and that the dynamic obstacle appears. In figure 7.19c the obstacle is very close, which results in the avoidance maneuver seen in figure 7.19d. In figure 7.19e one can see that the obstacle has lost its velocity, and that the vessel is progressing towards the goal. In figure 7.19f the goal is reached.

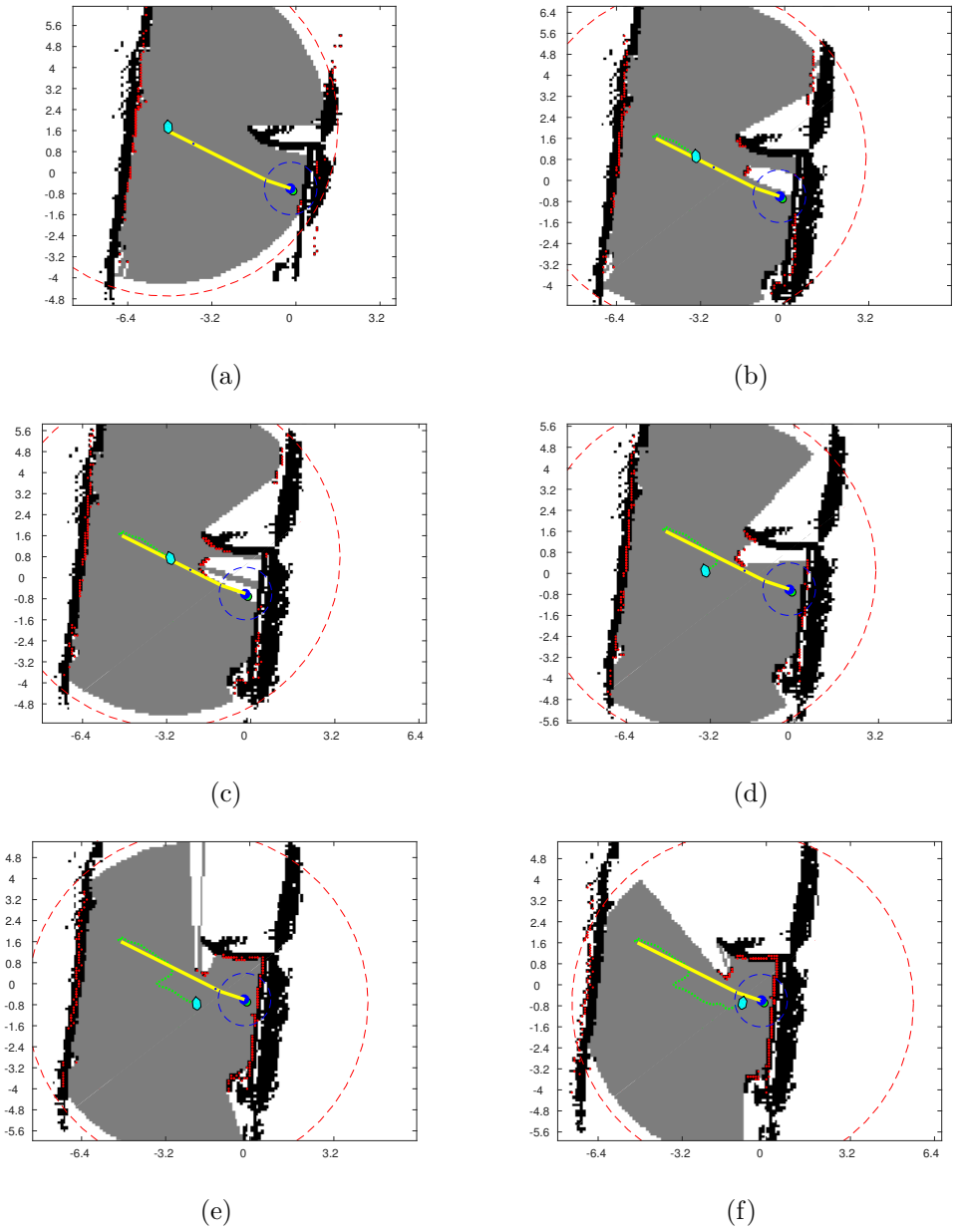


Figure 7.19: An experimental run in the MC-lab depicting a crossing situation

From figure 7.19e once can see that the vessel steers itself backwards at an angle before circumventing the obstacle. From figure 7.20 one can see that the trajectories generated by the modified DW algorithm is not being followed as intended. This can be seen by comparing the course of the trajectories with the course of the obstacle maneuver depicted in figure 7.19d. But, one must note that the trajectories are only predictions made by the modified DW to evaluate the candidate velocities, and are really meant to be followed. They are supposed to represent the predicted behavior of the vessel, if the corresponding velocity command is chosen. In this case, one can see that the predicted behaviour does not reflect the behaviour of the vessel in the real world.

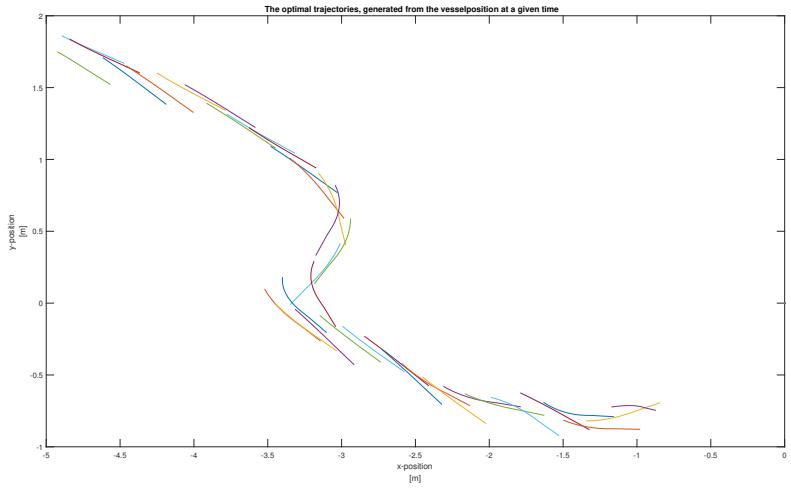
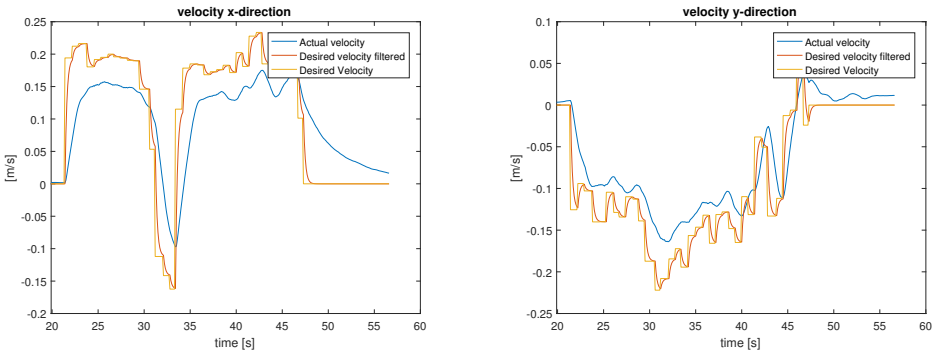


Figure 7.20: The Optimal Local Trajectories generated by the modified DW algorithm



(a) Speed along the x-axis

(b) Speed along the y-axis

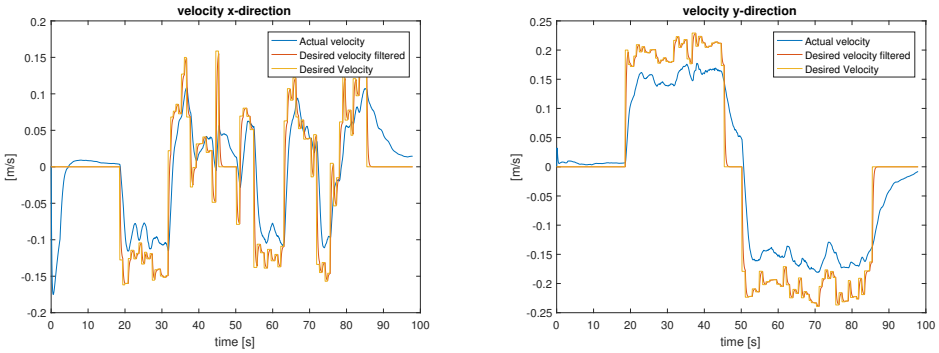
Figure 7.21: Velocity commands and actual velocity during the experiment

From 7.21 one can see the estimated velocity plotted against the desired velocity. From this figure, one sees that the velocity controlled functions much the same way as describe in chapter 6.4.4. From what can be seen in figure 7.21 it looks like the reference is held quite well, with no abnormalities that would suggest a maneuver like what is shown in figure 7.19. Still the velocity estimates may be somewhat unreliable when the thrust changes direction as stated in chapter 6.4.3. In other words, the abnormal nature of the avoidance maneuver may be caused by unreliability in the thrust when the thrust changes direction.

7.5.3 The Enclosed Video

In the digital appendix there is enclosed a video that showcase two separate experiments done in the MC-lab. One of the experiments is recorded as a screen capture from the map, while the other is a recorded video of the second experiment. The situations in the two experiments are similar, and shows two crossing like situations. The avoidance response in both of the cases can be seen to be very similar as well. In this section some results corresponding to the experiment done in the computer screen recording situation is presented. The video has also been uploaded to Youtube: `https://www.youtube.com/watch?v=DMAe6jWT5aE&feature=youtu.be`

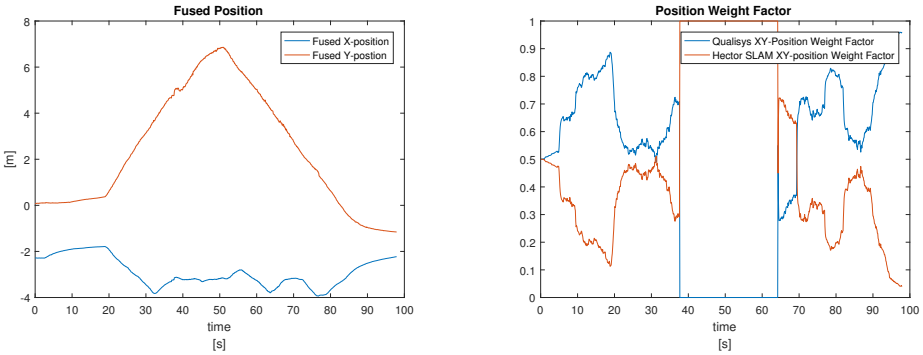
One thing to note, is that the velocity of the obstacles is relatively fast compared to the SC Saucer's velocity, however the velocity of the obstacles are damped out as they approach the Saucer. The high initial velocity will lead to that the CS Saucer will have little time to react and accelerate. In figure 7.22, the estimated velocity is plotted against the desired velocity. The figure shows that the desired velocity is being followed as intended, as described in chapter 6.4.4. Figure 7.22a shows that the vessel commands in the x-direction changes quite rapidly.



(a) Speed along the x-axis

(b) Speed along the y-axis

Figure 7.22: Velocity commands and actual velocity during the experiment



(a) Speed along the x-axis

(b) Speed along the y-axis

Figure 7.23: The fused position of the CS Saucer during the experiment preformed in the enclosed video

In figure 7.23 the fused x- and y-position of the vessel can be seen plotted over time. From figure 7.23b the weighing of the position estimates can be seen. One thing to note is that the Qualisys system no longer detects the vessel at about 38 seconds

into the experiment. This corresponds to when the CS causer is at its closest to the recording camera.

7.6 Virtual Obstacle in the MC-lab

In this experiment, a virtual obstacle is added to the map. This virtual obstacle has a velocity of the same magnitude as the simulated obstacles used in the simulations. Figure 7.24 shows the complete experimental run.

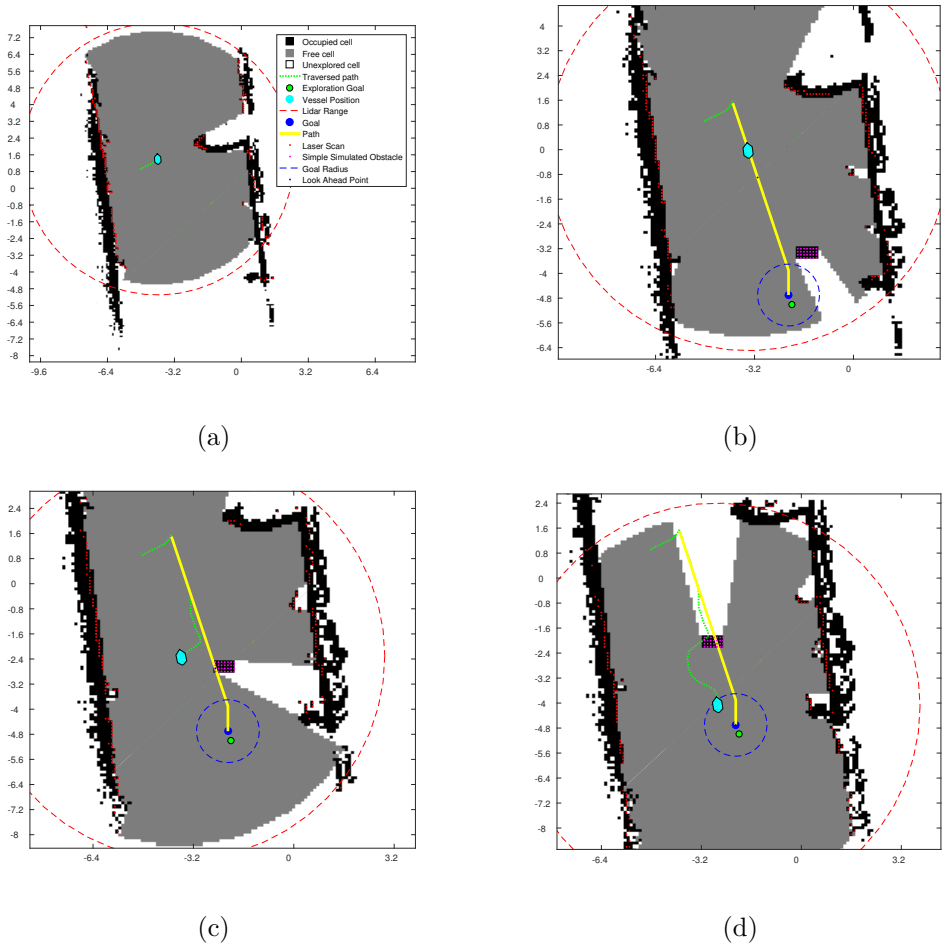


Figure 7.24: Experiment with a virtual dynamic obstacle

As seen in figure 7.24b, the virtual obstacle is not shown in the map before it is in the line of sight of the LIDAR. Figure 7.24c and 7.24d shows the collision avoidance maneuver. What can be stated immediately, is that the obstacle avoidance leads the vessel in front of the velocity direction of the obstacle. However, the vessel reaches its goal collision free. In figure 7.25 shows the optimal collision avoidance trajectories that were chosen throughout the run.

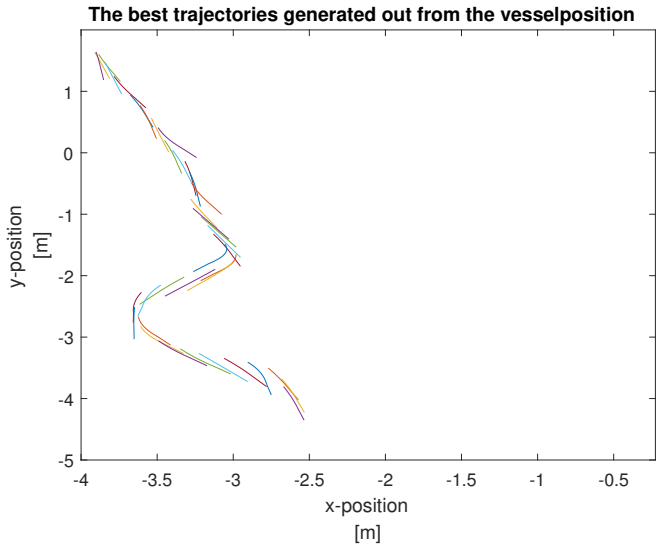


Figure 7.25: The vessel returns to the start area of the experiment

The tuning of the modified DW algorithm for the experiments was done using virtual obstacles for the most part. Virtual obstacles were much easier and efficient to use in comparison to using real obstacles for a long tuning process. The scenario presented in this chapter was one of the tuning scenarios. From figure 7.24 and 7.25 one can see that the vessel circumvents the vessel quite smoothly, as one should expect from a tuning scenario.

Chapter 8

Discussion

The results presented in chapter 7 show that the performance of the collision avoidance system developed in this thesis. The collision avoidance system is shown to handle unknown static obstacles very well. This is no surprise, considering that the modified DW algorithm was designed to be more risk averse than the original DW algorithm. However, there is still a chance that collisions with static obstacles might occur, due to the blind spots in the laser scans. The chance for this occurring is very low, and a thin obstacle as seen in 7.4 would be a worst case scenario where collision due to the blind spots could occur.

The collision avoidance system is also shown to handle dynamic obstacle to a certain degree. However, as illustrated in the results presented in chapter 7.5.1 and 7.6 the collision avoidance system might lead the vessel into performing some dangerous maneuvers that does not satisfy the COLREGS rules. This is as expected, since the velocity of the obstacles are not estimated or known beforehand. This leads to that the COLREGS rules could not be implemented to constrict maneuvers that would lead the vessel into the course of dynamic obstacles. Velocity information regarding obstacles could have been used to update the map with predicted obstacle positions, or made the local inflation of the map a function of the obstacle velocity. Since this is not done, the collision avoidance system is forced to see the situation in each time frame as static. This will make a maneuver as seen in chapter 7.5.1 seem like a good choice, since it will hold a greater bearing to the obstacles in the vicinity of the vessel.

One huge limitation for the collision avoidance system as a whole is the time lag. To avoid this, either the visualization process of the map, or the modified DW algorithm itself should have been implemented as a separate ROS node. In this thesis, the collision avoidance system's reaction time depends on the computational time required to do one iteration in the "*exploration_pathplanner*" node. The computational time of the node as a whole was found to be approximately 0.5-0.6 seconds. This is a very slow reaction time, even in comparison with the reaction time of an average human. This time delay can be devastating for the performance of the collision avoidance system, especially for head on situations. As already stated, the time delay problem could have partly been solved by discretizising the "*exploration_pathplanner*" into smaller ROS nodes. This would have made the nodes less dependent on each others' computational time. In other words, one of the main strengths of ROS has not been utilized properly in the system developed in this thesis.

There is also the problem regarding the velocity estimate. As stated in chapter 6.4.3, thrusters are unreliable when the thrust changes direction. Considering how the observer was tuned, this will lead to velocity estimates being flawed in certain situations. This will lead to some unexpected behaviour, since the course of the vessel is controlled by velocity commands for the x- and y-velocity of the vessel. Combining this with the time delay may lead to unwanted behaviours for a given control command. Especially when avoiding obstacles, where the thrust must often change direction in order to alter the the vessels course.

Because of this, the modified DW algorithm that was initially tuned only in the simulator, had to be re-tuned in order to function properly in the experiments. Still, the results in the simulator and the experiments are still comparable and very similar in some ideal cases. In the simulator, there will be no time delays and the vessel will behave as the mathematical model predicts. However, the simulations were still a valuable tool in order to test and evaluate the collision avoidance system. Two cases that are especially comparable are those that are presented in chapter 7.2 and 7.6. Since the experiment in chapter 7.6 uses virtual obstacles, it is a little less sensitive to the time delay. However, comparing the results one can see a much smoother collision avoidance response in the simulations.

As stated in chapter 6.3.4, the acceleration is not estimated or measured in this thesis, which might lead to that some of the speed commands generated by the modified DW algorithm might be unfeasible. This will lead to that some of the desired velocities are not possible for the vessel to reach within the given time frame. One could also question the accuracy of the dynamic constraints that were calculated in chapter 6.3.3, due to the possible inaccuracies present in the velocity estimates. An inertial measurement unit (IMU) could have been used in order to measure the acceleration of the vessel. Using the data gathered from a IMU, the vessel acceleration could have been taken into account in the modified DW algorithm, resulting in a more dynamically feasible control. The acceleration measured by the IMU could also have been integrated, in order to gain additional velocity information. This velocity information could have been fused with the velocity estimate from the observer, and perhaps used for preforming dead-reckoning.

From chapter 7.1, one can see that the sensor fusion works quite well, as long as at least one of the position systems are operational. However, there has been some cases during the experiments where both of the position signals were lost. Dead reckoning should have been introduced to the system to enable the system to handle situations where the position signal was lost. This would enable the vessel to maneuver for a short period of time by estimating the position from the last measured position, and would result in a more robust position signal. The combination of SLAM, a known map and the laser scan proximity information complements each other quite well, and results in a very good situational awareness. The dynamic obstacles are mapped into the already known map, as well as new information generated by the SLAM algorithm. However, the situational awareness could have been greatly improved by estimating the velocity of obstacles and taking this information into account when updating the occupancy grid. The blind-spots should also have been accounted for, either by controlling the heading to point them away from the velocity direction, or re-introducing the acoustic proximity sensors used in (Spange, 2016). In other words,

improving the situational awareness would increase the autonomy of the guidance layer, as seen in chapter 2.3.1. In this thesis, the guidance layer needs the human operator to select the desired goal point in order to generate a path. Using the taxonomy for autonomous systems introduced in chapter 2.3.2, the path generation will be classified as level 3:"Human Decision Select Stage", while the path following and obstacle avoidance can be classified as level 8:"Autonomous Control Stage".

The modified DW algorithm was specifically designed to be used for the CS Saucer. This means that the collision avoidance system is designed to be used for a omnidirectional vessel. In order to be used for a conventional under-actuated ship, the algorithm needs to be reworked to make the trajectories more similar to the circular arcs used in the original DW algorithm. The situations and environment of the situations presented in chapter 7 may be seen as similar to navigating in a narrow canal, or a small port area. The sensors that are used in the experiments are also comparable to real world scenarios. A radar could have been utilized to map the environment, and detect the presence of dynamic obstacles almost in the same manner as the LIDAR used in this thesis. A nautical-chart as described in 2.2.1 could have been used as an a prioi map of the static obstacles in area of operation. A GPS system could have been utilized to function the same way as the Qualisys system did in this thesis. In other words, sensors and tools that are equivalent to those used in the experiment are among the most common sensors and tools used in navigation for full-scale vessels. One should note however, that the experiments were preformed in a small-scale sheltered environment. This means that the developed system has been tested in an environment without environmental forces. This suggests that the system would need to be reworked in such a way that it would handle a much harsher environment in order to even be considered for a full-scale scenario.

From the results presented in this thesis, one can conclude that the obstacle avoidance system handles static obstacles very well. For dynamic obstacles, more work is needed in order to take the velocity of the obstacles into account, and use this information to make the collision avoidance system CORLEGS compliant. As it stands now, the obstacle avoidance system will often lead the vessel into the course of dynamic obstacles, where the vessel must rely on its superior maneuverability and velocity capabilities in order to avoid collision. From this it can be concluded that the collision avoidance system is capable of handling dynamic obstacles to a certain degree. This will enable the guidance system to autonomously follow a path in an uncertain environment, while the path will be autonomously re-planned only if deemed necessary by the system. However, the need of a human in the loop to define the goal, as well as the need for operating computers has a diminishing effect on the overall autonomy of the system. If the goal is defined however, the vessel will autonomously plan and follow the path, while autonomously deploying reactive avoidance maneuvers in a attempt to avoid dynamic obstacles. The sensors and tools used in this thesis to build up the situational awareness of the vessel are comparable to the most common sensors and tools used for navigation of full-scale vessels. This implies that the system developed in this thesis will be compatible with sensors used for full-scale vessels. The collision avoidance system is designed for an omnidirectional vessel operating in a small scale sheltered marine environment. The modified Dynamic Window algorithm would have to be reworked in order to function

properly for a conventional under-actuated vessel. However, for a full-scale vessel the collision avoidance system needs to be COLREGS compliant, as well as being able to handle a much harsher environment than a small scale sheltered laboratory environment.

Chapter 9

Concluding Remarks

9.1 Conclusion

In this thesis, a guidance system that combines a global path planner, and a local obstacle avoidance scheme has been developed for the CS Saucer. In addition, a sensor fusion scheme for combining the position estimates from the MC-lab Qualisys system and SLAM based on a 2D LIDAR has been developed. The resulting system has been extensively tested in computer simulations, and in experiments preformed in the MC-lab.

From the results gathered during the experiments, the collision avoidance system can be seen to work very well when encountering unknown static obstacles. The obstacle avoidance system is shown to be able to handle dynamic obstacles to a certain degree, however additional work is needed in order to enable the collision avoidance to promote COLREGS compliant maneuvers. This would prevent the collision avoidance system from guiding the vessel into the course of the dynamic obstacle, and lead to a much safer collision avoidance. Since the vessel is not designed to be COLREGS compliant and is designed for a sheltered environment, one can easily conclude that the collision avoidance system does not meet the demands of a full-scale scenario.

A collision avoidance system stands and falls on the information made available to it. In this thesis, blind spots were affecting the laser scans. In order to improve the situational awareness, the system should have been made aware of these blind spots or utilized the acoustic sensors used in (Spange, 2016) to measure distances in these directions. These blind spots could potentially lead to situations where collisions could occur, despite the fact that these spots generally are very small. Additionally, the relative speed between the CS Saucer and dynamic obstacles should have been estimated and used to update the map in order to reflect this information. This would have greatly increased the situational awareness of the system, and in return improved the overall performance of the collision avoidance system.

The sensor fusion between the position estimated by the Hector-SLAM algorithm and the position measurement from the Qualisys system made the position signal more robust, and better able to handle a dynamic environment. However, there is still a significant possibility for both of the position measurements to simultaneously fail at providing a measurement for the vessel's position.

Despite all the limitations, the collision avoidance system demonstrated through experiments and computer simulations that it could handle both static and dynamic obstacles. The global path planner preformed very well, and in combination with the LOS steering law it was able to guide the vessel towards a given goal. With further work on the situational awareness and the modified Dynamic Window algorithm could have improved the performance of the guidance system as a whole.

9.2 Further Work

In this thesis, many of the strengths of the Robot Operating System has not been properly utilized. In order to eliminate some of the time delays that is affecting the guidance system, the ROS node "*Exploration_pathplanner*" should have been split up into smaller ROS nodes. This would have made the computational time of each node more independent of each other. A suggestion for how this could be done would be to split up the path planning, the visualization of the map and the modified Dynamic Window algorithm into three different nodes. It should be noted that it is the visualization of the map that needs most of the computational time. This means that finding a less computational demanding way to visualize the map to the operator would also solve much of the time lag problem.

As stated in the further work chapter in (Sharoni, 2016), the thrusters are unreliable in certain situations. This unreliability is especially present in the zero thrust region, and when thrust direction is changed. Another concern is that the thrust does not increase or decrease in a smooth way. Furthermore, glitches are experienced where sudden jumps in thrust can be detected. There is also a problem when the thrusters rapidly changes direction of rotation. The motors are also very sensitive to electromagnetic disturbances, which may cause unexpected behaviours for the thrusters. Solving some of these issues would make the CS Saucer better able to follow given control commands.

Because of the thruster unreliability, and possible flaws in the mathematical model of the CS Saucer, the velocity estimated by the non-linear passive observer might be flawed in some situations. A possible solution for acquiring more accurate velocity information could be to install an IMU on the CS Saucer and integrate the acceleration measurement to get an estimate of the velocity. This would have enabled the option of fusing the two velocity estimates if deemed necessary. The information gathered from the IMU could also be used to preform dead reckoning in the case of loss of position. From what has been discussed in chapter 8, dead reckoning would strengthen the position signal estimated in the sensor fusion. As stated in chapter 6.3.4, acceleration measurements could have been used to further ensure that a control command is dynamically feasible.

As stated many times earlier in this thesis, the blind spots that were introduce by the cap atop of the CS Saucer must be accounted for. Either by making the blind spots smaller in the same fashion as described in section 6.6.3, or re-introducing the acoustics sensors that were used in (Spange, 2016) for the CS Saucer. Another possible solution is to simply make the system aware of the blind spots, and compensate for them by controlling the heading point the blind spots away from the velocity direction.

The relative speed between the CS Saucer and the dynamic obstacles should have been estimated and taken into account during the inflation of the map. This would have greatly improved the potential of the collision avoidance system. The LIDAR could have been used to calculate the relative speed of an obstacle, by using a set of laser scans and looking at the difference in them over time. By compensating for the vessel's own speed, one could get a measure of how fast a obstacle is approaching the CS Saucer. This does not provide the full scope of the basin-relative velocity of the obstacle, only the speed the vessel is approaching the CS Saucer. This information could potentially be used to rework the collision avoidance system to be COLREGS compliant, however it would be much easier to make the system COLREGS compliant had the full scope of the obstacle's velocity been known. In other words, further work should prioritize to find a solution for estimating the obstacle's velocity and use this information to ensure that the avoidance maneuvers satisfies COLREGS.

Bibliography

Simon David Adams. Revolt – next generation short sea shipping, 2014. URL `https://www.dnvgl.com/news/revolt-next-generation-short-sea-shipping-7279`. Accessed:03.06.2017.

Farzin Amzajerdian, Diego Pierrottet, Larry Petway, Glenn Hines, and Vincent Roback. Lidar systems for precision navigation and safe landing on planetary bodies. *Proceedings of SPIE - The International Society for Optical Engineering*, 8192, 2011.

Aliasghar Arab, Kaiyan Yu, Jingang Yi, and Dezhen Song. Motion planning for aggressive autonomous vehicle maneuvers. *International Conference on Automation Science and Engineering (CASE)*, 2016.

Alan Bole, Alan Wall, and Andy Norris. *Radar and ARPA Manual*. Third Edition. Butterworth-Heinemann, 2014. pages: 255—275.

J Borenstein and Y. Koren. The vector field histogram -fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

J. Borenstein and Y. Koren. Collision avoidance systems for autonomous underwater vehicles part a: A review of obstacle detection. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1999.

Alexander Bulinski and Alexey Shashkin. *Limit Theorems For Associated Random Fields And Related Systems*. World Scientific, 2007.

S. Campbell, W. Naeem, and G.W. Irwin. A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres. *Annual Reviews in Control*, 36(2):267–283, 2012.

T.J. Chong, X.J. Tang, C.H. Leng, M. Yogeswaran, O.E. Ng, and Y.Z. Chong. Sensor technologies and simultaneous localization and mapping (slam). *Procedia Computer Science*, 76:174–179, 2015.

Felix Endres, Jurgen Hess, Jurgen Sturm, Daniel Cremers, and Wolfram Burgard. 3-d mapping with an rgb-d camera. *IEEE Transactions on Robotics*, 30(1):177–187, 2014.

Bjørn Olav Holtung Eriksen. Horizontal collision avoidance for autonomous underwater vehicles (master's thesis, norwegian university of science and technology), 2015.

Maurice F. Fallon, John Folkesson, Hunter McClelland, and John J. Leonard. Relocating underwater features autonomously using sonar-based slam. *IEEE Journal of Oceanic Engineering*, 38(3):500–513, 2013.

P. Fiorini and Z Shiller. Motion planning in dynamic environments using the relative velocity paradigm. *Proceedings IEEE International Conference on Robotics and Automation*, 0:560–565, 1993.

Roar Fjellheim, Einar Landre, Roger Nilssen, Tor Olav Steine, and Aksel Andreas Transeth. Autonomous systems: Opportunities and challenges for the oil and gas industry.

Mats Håkon Follestad. A prestudy on autonomous path-planning and -following for a marine surface robot (project thesis, norwegian university of science and technology), 2016.

Thor I. Fossen. *Handbook Of Marine Craft Hydrodynamics And Motion Control.* John Wiley and Sons, Ltd, 2011.

D. Fox, W. Burgard, and S Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33, 1997.

R.J. Geraerts and M.H Overmars. A comparative study of probabilistic roadmap planners. *Springer tracts in advanced robotics*, 7:43, 2004.

Arturo Gil, Oscar Reinoso, Monica Ballesta, and Miguel Julia. Multi-robot visual slam using a rao-blackwellized particle filter. *Procedia Engineering*, 58(1):68–80, 2010.

P. E. Hagen, Ø. Hegrenæs, B. Jalving, Ø. Midtgaard, Wiig Ø., and O. K. Hagen. Making auvs truly autonomous underwater vehicles. underwater vehicles. *InTech*, pages 129–152, 2009.

P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *IEEE Transactions on Robotics*, 30(1):177–197, 2014.

Tor Kvestad Idland. Marine cybernetics vessel cs saucer: Design, construction and control (master thesis, norwegian university of science and technology), 2015.

IMO. Convention on the international regulations for preventing collisions at sea, 1972 (colregs), *International Maritime Organization.* URL `http://www.imo.org/en/About/conventions/listofconventions/pages/colreg.aspx`. Accessed:07.06.2017.

Asgeir J. Sørensen. *Marine Control Systems: Propulsion and Motion Control of Ships and Ocean Structures.* Department of Marin Technology NTNU, 2013.

Sungyoung Jung, Jungmin Kim, and Sungshin Kim. Simultaneous localization and mapping of a wheel-based autonomous vehicle with ultrasonic sensors. *Artificial Life and Robotics*, 14(2):186–190, 2009.

0. Khatib. Real-time obstacle avoidance for robot manipulator and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

Philipp Koch, Stefan May, and Sigurd A. Michael Schmidpeter. Multi-robot localization and mapping based on signed distance functions. *Journal of Intelligent & Robotic Systems*, 83:409—-428, 2016. doi: 10.1007/s10846-016-0375-7.

Sven Koenig and Maxim Likhachev. D*lite: Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354(10), 2005.

Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. *IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 155–160, 2011.

Y. Koren and J Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. *The International Journal of Robotics Research*, pages 1398–1404, 1991.

Y Kuwata, M.T Wolf, D Zarzhitsky, and T.L Huntsberger. Safe maritime navigation with colregs using velocity obstacles. *IEEE International Conference on Intelligent Robots and Systems*, pages 4728–4734, 2011.

Jacoby Larson, Michael Bruch, and John Ebken. Autonomous navigation and obstacle avoidance for unmanned surface vehicles. *Proceedings of SPIE - The International Society for Optical Engineering*, 6230, 2006.

S. M LaValle. Rapidly-exploring random trees: A new tool for path planning. *Department of Computer Science, Iowa State University*, 1998.

Thomas Lemaire, Cyrille Berger, Il-Kyun Jung, and Simon Lacroix. Vision-based slam: Stereo and monocular approaches. *International journal of Computer Vision*, 74(3):343–364, 2007.

Scott Littlefield. Anti-submarine warfare (asw) continuous trail unmanned vessel (actuv). URL `http://www.darpa.mil/program/anti-submarine-warfare-continuous-trail-unmanned-vessel`. Accessed:03.06.2017.

Ø. A. G. Loe. Collision avoidance concepts for marine surface craft (project report, norwegian university of science and technology, trondheim). 2007.

Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. *DARPA Image Understanding Workshop*, pages 121–130, 1981.

Yasir Mohd Mustafah, Amelia Wong Azman, and Fajril Akbar. Indoor uav positioning using stereo vision sensor. *Procedia Engineering*, 41:575–579, 2012.

Mark Prigg. The self driving warship: Us navy's 132ft-long 'sea hunter' drone that will scour oceans for enemy subs takes to the seas, December 2016. URL `http://www.dailymail.co.uk/sciencetech/article-4042298/The-self-driving-warship-Navy-s-132ft-long-Sea-Hunter-drone-scour-oceans-enemy-subs-html`. Accessed:03.06.2017.

Brual C Shah and Satyandra K Gupta. Speeding up a* search on visibility graphs defined over quadtrees to enable long distance path planning for unmanned surface vehicles. *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling*, pages 527–535, 2016.

Rotem Sharoni. Marine inverted pendulum (master thesis, norwegian university of science and technology), 2016.

Joachim Spange. Autonomous docking for marine vessels using a lidar and proximity sensors. (master thesis, norwegian university of science and technology), 2016.

Thomas Stenersen. Guidance system for autonomous surface vehicles (master's thesis, norwegian university of science and technology), 2015.

A. Stentz. Optimal and efficient path planning for partially-known environments. *IEEE International Conference on Robotics and Automation*, pages 3310–3317, 1994.

Asgeir J. Sørensen and Martin Ludvigsen. Towards integrated autonomous underwater operations. *Department of Marine Technology, Norwegian University of Science and Technology (NTNU)*, 2015.

C. S. Tan, R. Sutton, and J. Chudley. Collision avoidance systems for autonomous underwater vehicles part a: A review of obstacle detection. *Journal of Marine Science and Environment*, Part C(No C2)):39–50, 2004a.

Einar Skiftestad Ueland. Preparing the thruster and control systems on the cs saucer for autonomous tasks. (project thesis, norwegian university of science and technology), 2015.

Einar Skiftestad Ueland. Marine autonomous exploration using a lidar (master thesis, norwegian university of science and technology), 2016.

Marialena Vagia, Aksel A. Transeth, and Sigurd A. Fjerdingen. A literature review on the levels of automation during the years. what are the different taxonomies that have been proposed? *Applied Ergonomics*, 53:190–202, 2015. doi: http://dx.doi.org/10.1016/j.apergo.2015.09.013.

G Valhalla, 2010. URL `https://se.mathworks.com/matlabcentral/fileexchange/29107-real-time-pacer-for-simulink`. Accessed:10.06.2017.

James Vincent. The us navy's new autonomous warship is called the sea hunter, 2016. URL `https://www.theverge.com/2016/4/8/11391840/us-navy-autonomous-ship-sea-hunter-christened`. Accessed:03.06.2017.

Wikipedia. Nautical chart, *Wikipedia the free encyclopedia*, a. URL `https://en.wikipedia.org/wiki/Nautical_chart`. Accessed:07.06.2017.

Wikipedia. International regulations for preventing collisions at sea, *Wikipedia the free encyclopedia*, b. URL `https://en.wikipedia.org/wiki/International_Regulations_for_Preventing_Collisions_at_Sea`. Accessed:07.06.2017.

Wikipedia. Doppler effect, *Wikipedia the free encyclopedia*, c. URL `https://en.wikipedia.org/wiki/Doppler_effect`. Accessed:07.06.2017.

Wikipedia. Sensor fusion, *Wikipedia the free encyclopedia*, d. URL `https://en.wikipedia.org/wiki/Sensor_fusion`. Accessed:16.06.2017.

Wikipedia. Radar, *Wikipedia the free encyclopedia*, e. URL `https://en.wikipedia.org/wiki/Radar`. Accessed:07.06.2017.

Wikipedia. Robot operating system, *Wikipedia the free encyclopedia*, f. URL `https://en.wikipedia.org/wiki/Robot_Operating_System`. Accessed:10.06.2017.

Wikipedia. Rapidly-exploring random tree, *Wikipedia the free encyclopedia*, g. URL `https://en.wikipedia.org/wiki/Rapidly-exploring_random_tree`. Accessed:07.06.2017.

Øivind Aleksander G. Loe. Collision avoidance for unmanned surface vehicles (master's thesis, norwegian university of science and technology), 2008.

Appendix A

Electronic Attachments

The files described in this appendix are included in the electronic attachment of the thesis.

A.1 Parameter Generation Files

VesselParametersSet.m

A matlab script for setting parameters for both the "*fuseMotionController2016a.slx*" simulink model, the "*VesselSimulator.slx*" simulink model and "*MotionControl.slx* simulink model. The script runs another script called "*rotasjon.m*".

VesselParametersSetSimulations.m

Same as above, only that it sets the parameters for the simulations.

rotasjon.m

This script is responsible for calculating the conversion parameters for the Qualisys system in regards to being converted to the Basin-relative frame. To do this it uses data logged by the simulink model *qualisys_convert* during the initial map generation phase, where the system developed in (Ueland, 2016) is used. The "*rotasjon.m*" script is dependent on the result files *"det_rot.mat*", "*heading_actual.mat*", "*heading_diff.mat*", "*Pq.mat*" and "*Ph.mat*"

Froniter256_original.mat

Files containing stored MATLAB workspaces. For simple setting of the dimensions of the Simulink nodes.

posedata.mat

Files containing stored MATLAB workspaces. For initializing of simulations.

setPathPlanParams.m

This script sets the parameters for the "*Exploration_pathplanner.slx*" Simulink model.

setPathPlanParamsSimulations.m

Same as above, only for simulations.

A.2 ROS nodes that are Launched During Deployment

A.2.1 Exploration_pathplanner.slx node

This is the Simulink model responsible for generating the path, deploying reactive obstacle maneuvers, process the map, and visualizing the map for the operator. The node subscribes to the vessel's position, estimated velocity in Relative-basin frame, the laser scans and the redeployed Hector-SLAM map, while it publishes the velocity command form the modified DW algorithm. The node depends on the following MATLAB functions:

- **PathplannerAndVisualization.m**

  This function is responsible for generating the path, and visualizing the map to the operator, as well as receiving input from the operator regarding the goal that is to be reached. This function relies on several other MATLAB functions which will be listed below:

  1. **inflatemapMats.m**
     Responsible for inflating objects according to the desired inflation radius.
  2. **inflatemapLocalMatsExpMain.m**
     Responsible for inflating/weighing nodes around obstacles that are within LIDAR range when the path is re-planned.
  3. **LidarUpdate.m**
     Assumes that the vessel can see ahead in sections where there are no obstacles detected in the map. Developed and used in (Ueland, 2016).
  4. **astar.m**
     Generates a path using the information present in the map, the position of the vessel, and the position of the goal. This will only be run when the goal is chosen by the operator. It will only run once, unless a re-plan is deemed necessary by the guidance system.

- **dynamic_obstacles.m**

  This MATLAB function is responsible for adding simple simulated obstacles to the map. This can be done in the simulations and during the experiments, making it possible to tune the system using virtual dynamic objects in the MC-lab

A.2.2 Hector2VesselPos

Hector2VesselPos.slx

Simulink model used to generate the node in C++. Transforms position vector from LIDAR coordinate system to that of the vessel. Includes a quarternion transformation.

Hector2VesselPos node folder

Is used to be run as a regular ROS node. Developed by (Ueland, 2016).

A.2.3 fuseMotionControl2016a.slx Node

The control system of the vessel. It controls the ship to hold the desired velocity, with a small off-set. The sensor fusion and observer velocity estimation is also preformed in this node. The observer and actuator mapping used in this node was developed in (Ueland, 2016)

A.2.4 Hector-SLAM nodes

Open source nodes that are used for SLAM, see: `http://wiki.ros.org/hector_slam`. Implemented for the CS Saucer in (Ueland, 2016).

A.2.5 RP-LIDAR node

Open Source nodes that are used as a driver for the RP-LIDAR, see: `http://wiki.ros.org/rplidar`. Implemented for the CS Saucer in (Ueland, 2016).

A.2.6 ROS serial node

Open source package for the Audrino, see: `http://wiki.ros.org/rosserial`. implemented for the CS Saucer in (Ueland, 2016)

A.2.7 Audrino code

CSSaucerThrustRPMVoltage.inu

Audrino code responsible for publishing PWM signals to the actuators and publishing the Acoustic signals (in cm) if the acoustic sensors are connected as in (Spange, 2016). The code is written in C++ and utilizes Audrino/ROS libraries. Implemented in (Ueland, 2016) and edited by (Spange, 2016).

CSSaucerSimple.inu

A simpler version of "CSSaucerThrustRPMVoltage.inu" that only publishes signals to the actuators. Implemented in (Ueland, 2016).

A.2.8 qualisys_convert.slx

This simulink model is run while the system developed in (Ueland, 2016) is used to generate the initial map. It logs position data from both the Qualisys system, and the Hector-SLAM positioning system. This data is saved to separate files, and processed by the MATLAB function "*rotasjon.m*".

A.3 Simulator nodes

A.3.1 VesselSimulator node

"*VesselSimulator.slx*" is a Simulink model used to simulate the CS Saucer in the simulations. This node was developed in (Ueland, 2016).

A.3.2 MotionControl.slx

A Simulink model that is used for simpler simulations, if one does not have all the necessary drivers for the Qualisys system. It is a slightly modified version of the motion controller developed in (Ueland, 2016).

A.3.3 Exploration_pathplannerSimulations.slx

In many ways a simpler version of the "*Exploration_pathplanner.slx*", since the ROS subscription to the Hector-SLAM map and the laser scans are removed. This allows for easier simulations if the original versions won't start due to ROS errors.

A.4 Launch Files

Res01.launch

Launches the RP-LIDAR and Hector-SLAM nodes for mapping, with a resolution of 0.1 [m] and a gridsize of [256x256]

LaunchNoLidar.launch

Launches nodes for deployment of vessel, excluding the RP-LIDAR and Hector-SLAM nodes.

A.5 Other

A.5.1 Real Time Pacer

Open source simulink block for slowing simulations approximately down to real-time. (Valhalla, 2010)

A.5.2 The enclosed Video

The enclosed video showcase two separate experiments done in the MC-lab. One of the experiments is recorded as a screen capture from the map, while the other is a recorded video of the second experiment. The situations in the two experiments are similar, and shows two crossing like situations. The avoidance response in both of the cases can be seen to be very similar as well. The video has also been uploaded to youtube: `https://www.youtube.com/watch?v=DMAe6jWT5aE&feature=youtu.be`

Appendix B

Software Set Up And Installation

B.1 Manual for getting started with ROS and to install the RP-lidar driver and Hector-SLAM package

A very good manual for helping NTNU students who wants to use ROS as their software framework for projects in the NTNU marine cybernetics laboratory is presented in the appendix of (Spange, 2016). The manual was originally presented in (Ueland, 2016), but some slight modifications were made in (Spange, 2016). Hence the reader is referred to read the manual in (Spange, 2016) for guides on how to: Installing ROS and UBUNTU on your personal computer, installing ROS and UBUNTU on a RaspberryPi-2, getting started with ROS, Communicating between RaspberryPi-2 and personal computer, getting the Audrino on ROS and installing the drivers for the RP-LIDAR and Hector-SLAM in ROS. There are also presented several ROS tutorials for beginners in this manual.

B.2 Qualisys and ROS

For a guide on how to import data form the Qualisys motion capture system, the reader is referred to read the appendix of (Sharoni, 2016), or use the guide on Qualisys found in the MC-lab handbook, at: `https://github.com/NTNU-MCS/MC_Lab_Handbook`

Appendix C

Launch Manual

C.1 Deploy vessel for Experiment in the MC-Lab

This section describes how to deploy the vessel for operations as seen in this thesis. The manual assumes that the system has not changed since this thesis. In this manual, lines that start with **$** are written in a Ubuntu terminal.

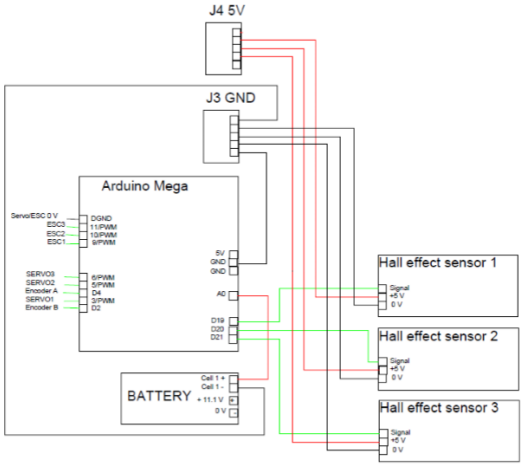


Figure C.1: Wiring diagram of the Audrino Mega, courtesy of Sharoni (2016)

- Make sure that the Arduino is connected as intructed in table 4.1. Also see figure C.1.
- Connect the battery to the Raspberry Pi 2. And place the lid as seen in figure C.2

Figure C.2: Lid placement and LIDAR placement, courtesy of Ueland (2016)

- Connect to the MC Lab network on the operator computer.
- **Map Generation phase**

  Perform a run with autonomous exploration with the system developed in (Ueland, 2016). The launch manual for his can be found in appendix C.1 in (Ueland, 2016). However, in order to log the Qualisys system while the map generation phase takes place, some additional steps must be taken. This means that the map generated, and the logging of qualisys data will be done by following these steps:

  1. Download the system developed in (Ueland, 2016) from `https://github.com/NTNU-MCS/CS_Saucer_ROS`.
  2. SSH into the RP2 and launch the RP2 nodes that are listed below by preforming the following commands in the terminal window:

     $ ssh ubuntu@ubuntu

     $ cd catkin_ws/src

     $ roslaunch LaunchNoLidar.launch
  3. If not already done, place the vessel on the water
  4. Open another terminal, and ssh into the RP2, and launch the mapping file according resolution 0.1[m]

     $ ssh ubuntu@ubuntu

     $ cd catkin_ws/src

     $ roslaunch Res01.launch
  5. Now launch the simulink exploration node, in order to generate the map. This is done by launching the "*Exploration_pathplanner.slx*" that was used in (Ueland, 2016) (It may be very confusing that the old exploration system in (Ueland, 2016) has the same name as the path planner and collision avoidance node in this thesis) by doing the following in MATLAB:

     5.1 **rosinit('ubuntu')(workspace commando)**

     5.2 **load Frontier256.mat (workspace commando)**

     5.3 **run Exploration_pathplanner.slx (Old exploration version)**
  6. Now one must launch the Qualisys system by opening a new terminal window and writing the command:

     $roslaunch qualisys qualisys.launch
  7. Make sure that all the reflectors are seen by the qualisys cameras, and that at least the first measurements will be valid for further use before running the "*qualisys_convert.slx*" Simulink model in order to log the Qualisys data and Hector-Slam position estimates.
  8. Start the exploration either by:

     Option a: Connecting a second computer to the system. Use this computer to run the motion controller node developed in (Ueland, 2016), and autonomously explore and generate the map.

Option b: Explore area manually by fasting a rope to the CS Saucer and leading it to un explored areas, and not launch the motion controls. This way one only needs to launch the map generation. make sure that it has moved around in the area, since this will make the conversion of the Qualisys system more feasible.

- After the map generation phase, the old system is shut down, along with the ROS nodes currently operative on the network. It is important to let the MATLAB script "*setPathPlanParams.m*" find and load the generated map. One should also check if data gathered from the Qualisys system and the Hector-SLAM algorithm lead to a successful conversion of the Qualisys system into the Relative-basin frame by running the MATLAB script "*rotasjon.m*". If the conversion was complete, then the CS Saucer is ready to navigate in the generated map using the Qualisys positioning system.

- After this, one needs to SSH into the RP2 yet again (if not already connected from the precious run), and launch the nodes as demonstrated below:

  1. **$ ssh ububntu@ubuntu**
  2. **$ cd catkin_ws/src**
  3. **$ roslaunch LaunchNoLidar.launch**

  The vessel should also have been placed on the water.

- Start the mapping file by doing the following:

  1. **$ ssh ububntu@ubuntu**
  2. **$ cd catkin_ws/src**
  3. **$ roslaunch Res01.launch**

- Make sure that the qualisys system sees the reflectors, and launch the qualisys ROS node as folows:

  1. **$ roslaunch qualisys qualisys.launch**

- Run the MATLAB script "*VesselParametersSet.m*". Make sure that the data collected by the Simulink model "*qualisys_convert.slx*" is in the same map as the script "*VesselParametersSet.m*" before doing this,

- After this, the Simulink model "*fuseMotionControl2016a.slx*" must be started and set to run. It is very important that the values of the Qualisys system does not freeze during the startup, as it will potentially ruin the conversion of the re-deployed Hector-SLAM position estimate.

- Use the other computer to run the MATLAB script "*setPathPlanParams.m*", given that it has access to the map generated beforehand.

- Now, the Simulink model "*Exploration_pathplanner.slx*" with collision avoidance can be started (remember that the already known map must be loaded first), and used to navigate in the known map. A Goal can be set by pressing on a pixel on the visualized map.

One thing to note is that the first time the "*Exploration_pathplanner.slx*" is run at a given MATLAB session, because of a "setupimpl error" for the laser scans. If this error occurs, just repeat the two previous steps. One thing to note, is that the laser message needs to be set to a custom size. the standard size is [1x128]. This needs to be set to [1x360] by going to "Tools → Robot Operating System → Manage Array Sizes" And then selecting "sensor_msg/Laserscan", remove the check on the "Use default limits for this message type" and write 360 in the maximum length field for the Ranges and Intensities.

Trouble Shooting

Trouble shooting as presented in (Ueland, 2016)

- Check if you can ping the RPi2 from one of the operator computers.
- Make sure that you can SSH both back and forth between the RPi2 and the operator computer. If not, there might be a problem with either the network connection or the hosts file. Also, make sure that open-ssh is installed, and if using Virtual Box, that you use Bridged Network
- The IP addresses can change. Check the hosts file on both the RPi2 and the operator computer if the IP adresses are up to date.
- Check that the bashrc file contains the following line:
  export ROS_MASTER_URI=http://ubuntu:11311 where ubuntu is the corresponding name to the RPi2 IP as set in the hosts file.
- Check the voltage of the battery
- Check that all pins are connected
- If one of the motors has stopped, check the lights on the motor-controller. They may signal an error described in the following: `http://www.mtroniks.net/download.asp?ResourceID=1973`

C.2 Perform Simulations

C.2.1 Simulations for the Developed System

In this manual it is assumed that all nodes are run on the operator computer, and that they are not compiled to C++. First make sure that the line in the bashrc file that exports the ROS master is uncommented. Make sure that the real time pacer is added to the path. After this, simulations can be launched in the following manner:

- **rosinit** (workspace commando)
- **run setPathPlanParamsSimulations.m**
- **run Exploration_pathplanner.slx**
- If error for a "setupimpl error" for the laser scans, repeat the two previous steps. If this does not work, see appendix C.2.2.
- **run VesselParametersSetSimulations.m**

- **run fuseMotionControl2016a.slx**
- **run VesselSimulator.slx**

Note, that in order for the "*fuseMotionControl2016a.slx*" to work one needs to complete all the steps described in appendix B.2. If still not able to run the simulations see the section below.

C.2.2 Easy Simulations

If the above section fails to yield simulations, then this section will provide an easier solution that still showcase some of the primary functions of the system. Note that it is assumed that these simulations is run with simulink on a Ubuntu operating system. However, the simple simulator should also work on a Windows operating system. In the electronic attachment these files are found in the "SIMULATIONS ONLY" folder. Follow the steps below to perform the simulations.

- **Add the Real Time Pacer to the MATLAB path**
- **rosinit** (workspace commando)
- **run setPathPlanParamsSimulations.m**
- **run Exploration_pathplannerSimulations.slx**
- **run VesselParametersSetSimulations.m**
- **run MotionControl.slx**
- **run VesselSimulator.slx**

If everything went according to the plan, the simulations are now running.

Appendix D

Parameters

Parameter	Experiments	Simulations
α	0.09	0.25
β	200	200
β_2	$\frac{80}{7.5}$	8
γ	50	52
C_i	0.46	0.45
w_v	0.15	0.8
w_t	0.85	0.2
Δt	0.2	0.2

Table D.1: Dynamic Window Tuning Parameters

Appendix E

Simulator ROS architecture

In all the stress surrounding this thesis, the node network for the ROS system in the experiment was saved as a figure. Because of this, the third best thing has been included instead, namely the node network of the easy simulations.

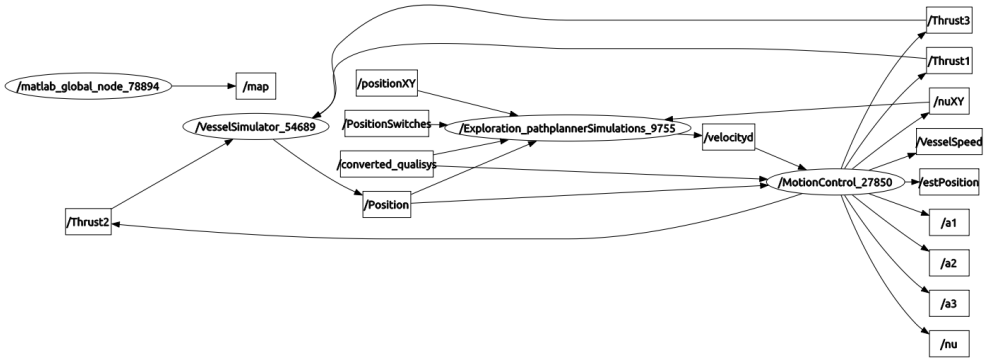


Figure E.1: Node network for the easy simulations. Apparently the author had forgotten to save the node network during the experiments