Trajectory Planning and Following for Autonomous Passenger Ferries

Emil Thyri

December 2018

NTNU
Norwegian University of
Science and Technology

Preface

The work presented in this report is the last work I do in my studies before I finalize my degree with the master thesis. Diving into the field of autonomous vessels and collision avoidance has been interesting, overwhelming and motivating. The knowledge and experience I have acquired is invaluable and will serve me well in the final semester of my 5-year Masters degree in Cybernetics and Robotics and in what follows.

I would like to express a special thanks to my academic supervisor Morten Breivik for the time he has dedicated to guidance and feedback in the scope of this project. His feedback has been crucial in the forming of the final product of my work.

Emil Hjelseth Thyri
Trondheim, December 18, 2018

Abstract

This report treats problems related to collision free transit for autonomous USVs. Specifically it considers the case of an autonomous passenger ferry, milliAmpere, and presents a deliberate COLAV system for short distance transit. In addition, a comparison between three trajectory following control strategies is performed with regards to passenger comfort.

The collision avoidance is solved by a two step trajectory planning process. Firstly, a path that does not collide with any static obstacles is planned, secondly, a velocity profile is planned for the path so that it avoids collision with any moving objects. The approach transforms moving objects onto a two dimensional space in $path \times time$ and hence reduces the trajectory planning to a shortest path node search problem. Special considerations are taken in order to not violate the physical capacities of the ferry, as well as to ensure sufficient passenger comfort. A cost function that favours transit time and passenger comfort is used to select the optimal trajectory from the potential candidates.

A simulator with a vessel model is developed in order to facilitate testing of the COLAV system, as well as the trajectory following methods. The simulator is made to have the same interface as the systems on the milliAmpere ferry in order to facilitate a seamless transition from simulation to testing.

Three trajectory following controllers are implemented and simulated. A reference feed forward, with pose feedback, a LOS guidance law with along track distance feedback, and a MPC is tested in the simulator with a trajectory suggested by the COLAV system. The methods are compared based on tracking error, along with metrics related to passenger comfort.

The COLAV system proves to be effective and intuitive. The two dimensional $path \times time$ representation makes it easy to follow the method of the algorithm. It has no problem finding a collision free trajectory, if one exists. The method lacks robustness towards uncertainty in the environment, and should be used in combination with a reactive COLAV system.

Table of Contents

List of Figures

List of Tables

Abbreviations

ARPA	Automatic Radar Plotting Aid
COLAV	Collision Avoidance
DOF	Degrees of Freedom
DP	Dynamic Positioning
ECEF	Earth Centered Earth Fixed
ENC	Electronical Nautical Charts
GNSS	Global Navigation Satellite System
Lidar	Light Imaging Detection and Ranging
LOS	Line of Sight
MPC	Model Predictive Control
NED	North East Down
NTNU	Norwegian University of Technology and Science
OBC	On Board Computer
PID	Proportional Integral Derivative
ROS	Robot Operating System
USV	Unmanned Surface Vessel

1 Introduction

In this chapter, the motivation for the work in this project is presented along with a problem description and a list of contributions from the author to the project. An introduction to the background and previous work on the field is included. The chapter is concluded with an outline of the report.

1.1 Motivation

The milliAmpere ferry was initially though of as an alternative to a walking bridge across a canal in Trondheim harbour that would block of a small-boat harbour for all boats above a certain height. The idea was taken further and became a project with a goal to develop the first autonomous passenger ferry in the world. A 5 meter long prototype has been build and functions as a platform for development and testing of all systems that are needed on an autonomous passenger ferry. The project is owned by NTNU Trondheim and allows for student to contribute to the development. The potential of the project is huge, where the finished product is highly scalable and can contribute to the way cities and infrastructure is planned. As the populations in cities around the world is growing, the need for flexible and environmentally friendly transportation along the waterways is growing along with it. Electric autonomous passenger ferries can fill this demand, as they are low cost (relative to building solid infrastructure), flexible, where it is easy to add more ferries or new transit routes depending on season or other needs, and good for the city environment, in the way that they do not pollute noise nor exhaust like traditional collective transport systems. The market for autonomous ferries is very real, and the project can contribute to the growth of maritime industry in Norway. The research and development that goes into this project will contribute to keeping Norway a leading country in the maritime industry.

1.2 Problem Description

The problem this project sets out to solve is to develop a planning and control system that can perform a transit with an autonomous passenger ferry, milliAmpere, in an environment defined by a canal in the harbour of Trondheim City. The ferry can be seen in Figure 1, while the environments are displayed in Figure 3. The system is to interface with existing systems already on the ferry, such as thrust allocation and sensor systems. The following objectives are proposed for this project:

- Research methods for COLAV
- Formulate a set of considerations for a trajectory planning system
- Develop a simple COLAV system

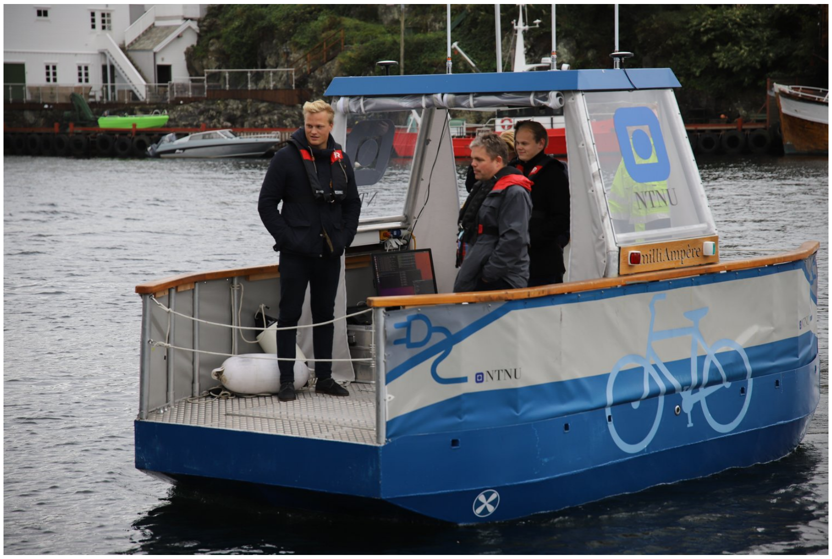

Figure 1: The prototype milliAmpere during a demonstration in Haugesund in September 2018. The ferry functions as a platform for development and testing of sensor and control systems needed for fully autonomous sea-travel. Courtesy of Teknisk Ukeblad.

- Develop a trajectory planning system
- Look into trajectory following methods
- Develop a simulator for the ferry
- Implement a trajectory following system
- Simulate and evaluate the COLAV system
- Simulate and evaluate the trajectory following system

1.3 Contribution

The contributions of this project thesis are threefold.

- A simulator that interfaces with the existing control and sensor-systems on the ferry is developed. The simulator will facilitate rapid testing and prototyping of new and old systems and algorithms in both unit testing as well as full system testing on the OBC of the ferry. This can become a time-saver for everyone working on the project, and a needed safety-measure in the development of COLAV systems. The simulator is generic and easily adaptable. This makes for a simple augmentation of the simulator when the milliAmpere ferry system is expanded with new sensors systems and functionality.

- A deliberate COLAV system is developed and tested. The system is design with respect to the environment the ferry is to operate in. In addition, the COLAV system is simple, predictable and does not introduce a high computational cost to the OBC of the ferry.
- A job is done to compare trajectory following systems in order to see what control strategy best complements the COLAV systems while giving a vessel behaviour close to the desired behaviour with regards to passenger comfort.

1.4 Previous Work

COLAV is an essential part of every traffic system, or transportation system. The task of a COLAV system is to prevent two or more objects from from colliding into each other. This includes detecting objects, predicting the chance of collision and suggesting or even performing a maneuver in order to avoid or reduce the chance for, or damage from a collision.

Traditionally this has been a task for a skilled operator, well known with the rules and laws of the environment he is operating in, as well as the system he is operating. Despite qualified operators, accidents do occur, and the last decades a lot of COLAV systems has been developed to assist or take over for operators. In the automotive industry, this is very apparent, where adaptive cruise control, emergency braking systems and lane assist has emerged on commercially available vehicles. In aviation, Radar based *air traffic control* have been in use for several decades. The system assists pilots and air traffic officers to keep a minimum distance between aircrafts in zones with high traffic, such as the airspace over big airports. For marine traffic, ARPA is in quite extended use, and is an assisting system that broadcasts the positioning, heading and velocity of the vehicle, as well as receives the same information from close by vessels. It then uses this information to estimate the closest point of approach, as well as alerting potential collisions.

As the autonomous technology is advancing, allowing for vehicles to operate autonomously in the presence of other autonomous and non autonomous vehicles, it will not longer be sufficient to have operator assisting systems. The COLAV systems has to be able to match the autonomous level of the craft. This includes a high level of precision in detection of all other objects in close proximity, robustness to noise and full redundancy.

As of today there is a number of commercial actors working hard to develop into the segment of fully autonomous surface vessels, and hence also has to solve the problem of fully autonomous collision avoidance. Between January and April 2018, technology group Wärtsilä was successfully testing its auto-docking technology on the Foglefonn ferry, a 85 meter long passenger and car ferry on the transit Jektevik-Hodnanes in Norway. Later the same year, they expanded the system to include automated transit, and hence had system capable of a dock-to-dock transits [3]. The system is fully autonomous is normal situations, but does not have capability to handle situations due to lack of situation awareness, and therefore the system is dependent on personnel monitoring it, ready to take over in any situation. On the 3.December 2018 Rolls-Royce demonstrated their autonomous ferry transit system that is able to travel from dock to dock, while detecting obstacles and performing maneuvers to avoid collision [4]. The system monitor the environments by a set of sensor systems, and is able to detect and react to a situation, such as a ship on collision course. Kongsberg Maritime also has been developing and testing autonomous vessels for some years, and has started production of a fully autonomous cargo ship, with a vision of full autonomous operation by 2022 [5].

1.5 Outline

In Chapter 2 an introduction is given to some fundamental theory as well as some key terms and concepts. In Chapter 3, an approach to collision avoidance is presented, followed by suggestions to a trajectory following system in Chapter 4. In Chapter 5, the simulator and its functionality is described. Chapter 6 presents the results of the work done as well as a discussion. In Chapter 7, the work is summed up, and suggestions for future work are included.

2 Theoretical Background

In this section, some fundamental theoretical background will be presented, as well as an introduction to some terms and concepts that is used in this project thesis.

2.1 Vessel Modelling

2.1.1 Kinematics

In navigation and motion control of marine crafts, a set of reference frames can be can be used, depending on the area of use and tolerances.

- **ECEF** frame is an Earth-Centered-Earth-Fixed frame denoted $\{e\}$. Its origin is fixed to the center of the earth, and it rotates around the earths spin axis. A point on the surface of the earth will have a fixed set of coordinates in $\{e\}$. The $\{e\}$ frame can also be used to represent Longitude-Latitude-Altitude which is widely used in GNSS based navigation.

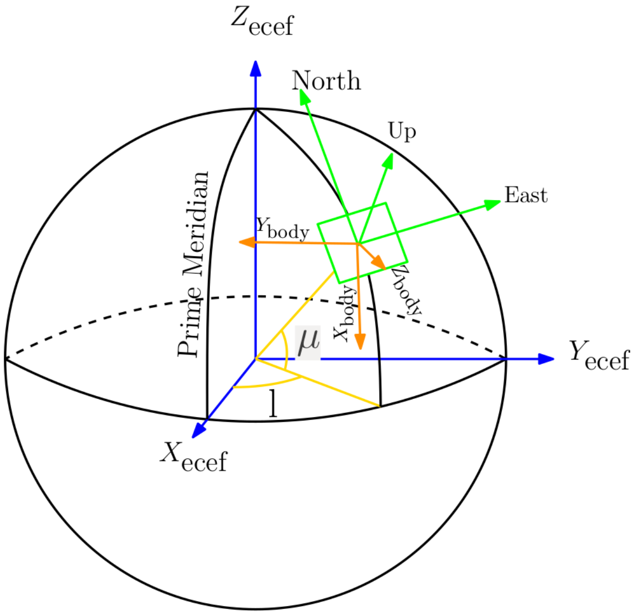

Figure 2: Reference Frames, ECEF frame in blue, NED frame in green and BODY frame in orange.

- **NED** frame is the North-East-Down coordinate system, hereby denoted $\{n\}$. The $\{n\}$ frame is used to describe the position and orientation of the craft. In this frame the x-axis point towards the true north, the y-axix points to the east, and the z-axix points down, normal to the surface of the earth. For flat-earth navigation, one can assume that $\{n\}$ is inertial, implying that Newton's laws still apply.
- **BODY** frame is the *Body-fixed-reference-frame* hereby referred to as $\{b\}$ is a coordinate frame fixed to the craft. The $\{b\}$ frame is used to describe the linear and rotational velocities of the craft. The x-axis of the $\{b\}$ frame is aligned with the longitudinal axis, of the craft, the y-axis is the transversal axis, and points straight to starboard, and the z-axix is the normal axis, and points straight down.

A ferry will typically operate in a local area where flat earth navigation can be assumed. This allows for use of $\{n\}$ in combination with $\{b\}$ to describe the system. Since we only consider 3 DOF, the orientation z-axis of $\{b\}$ and $\{n\}$ are parallel, the rotation from $\{b\}$ to $\{n\}$ becomes a principle rotation about the z-axix

$$\boldsymbol{J}_\Theta(\eta) = R_{z,\psi} = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

the rotation gives the relationship between the pose and the body velocities

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} \tag{2}$$

where $\boldsymbol{R}(\psi) := \boldsymbol{R}_{z,\psi}$ with $\boldsymbol{\nu} = [u, v, r]^T$ and $\boldsymbol{\eta} = [N, E, \psi]^T$

2.1.2 Kinetics

In order to develop model-based control systems as well as a simulation environment, a kinetic model of the vessel in the environments must be made. In this section, a 3 DOF model in the horizontal plane is presented [6]. The maneuvering model is based on the rigid body kinetics

$$\boldsymbol{M}_{RB}\dot{\nu} + \boldsymbol{C}_{RB}(\nu)\nu = \boldsymbol{\tau}_{RB} \tag{3}$$

$$\boldsymbol{\tau}_{RB} = \boldsymbol{\tau}_{hyd} + \tau_{hs} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{waves} + \boldsymbol{\tau} \tag{4}$$

where $\boldsymbol{\tau}$ represents the forces for the actuators on the vessel. Since the model only takes into account the horizontal plane $\boldsymbol{\tau}_{hs} = \boldsymbol{0}$. The hydrodynamic forces

$$\boldsymbol{\tau}_{hyd} = -\boldsymbol{M}_A\dot{\boldsymbol{\nu}}_r - \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r - \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r \tag{5}$$

is a result of the added mass, $\boldsymbol{M}_A$, Coriolis and centripetal matrix $\boldsymbol{C}_A(\boldsymbol{\nu}_r)$ due to the rotation $\{b\}$ with respect to $\{n\}$, as well as the viscous and wave induced damping. By combining (3)-(5) we get the maneuvering equation

$$\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{M}_A\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r\boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu} = \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{wave} \tag{6}$$

In the case of ocean currents, $\boldsymbol{\nu}$ and $\boldsymbol{\nu}_r$ will not be the same, this can be handled by parameterizing $\boldsymbol{C}_{RB}$ independent of linear linear velocity, and assuming constant currents and hence $\dot{\boldsymbol{v}}_c = \mathbf{0}$ [6]. By doing this the system can be written on the form

$$\boldsymbol{M}\dot{\boldsymbol{\nu}}_{\boldsymbol{r}} + \boldsymbol{C}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r = \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{wave} \tag{7}$$

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A \tag{8}$$

$$\boldsymbol{C}(\boldsymbol{\nu}_r) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu}_r) \tag{9}$$

2.2 Trajectory Tracking

A control system that forces the system output $\boldsymbol{y}(t) \in \mathbf{R}^m$ to track a desired output $\boldsymbol{y}_d(t) \in \boldsymbol{R}^m$ solves a trajectory tracking problem [6]. For surface vessels, the trajectory tracking problem usually concerns the 3DOF previously mentioned where $\boldsymbol{y}(t) = \boldsymbol{\eta}(t) = [N(t), E(t), \psi(t)]$ and $\boldsymbol{y}_d(t) = \boldsymbol{\eta}_d(t) = [N_d(t), E_d(t), \psi_d(t)]$, where the objective is to minimize the tracking error

$$\boldsymbol{e}(t) := \begin{bmatrix} N(t) - N_d(t) \\ E(t) - E_d(t) \\ \psi(t) - \psi_d(t) \end{bmatrix} \tag{10}$$

The trajectory tracking strategy is dependent on the vehicle actuator configuration as well as the trajectory to be tracked. A fully actuated vessel allows for independent control of all the the three degrees of freedom and is especially suited for crab-like motions as well as stationkeeping and DP .

2.3 ROS

Robot Operating System is software developed for building robot applications. ROS is not an operating system, but rather a collection of software frameworks for robot software development [7]. ROS is open source and free for commercial and research use. The software provided can be split into three groups: Tools for building and distributing ROS based software, ROS client libraries such as roscpp (C++) and rospy (python), and packages containing application related code which usually uses one or several of the ROS client libraries.

ROS was developed to facilitate collaborative development of robotic systems. With the ROS packages, a team of developers in one field can easily share or combine their work with developers in another field. This is much due to intuitive and standardized message based interface that ROS uses. This also facilitates the use of a variety of programming languages combined. ROS works well for both low level hardware drives, as well as advanced algorithms.

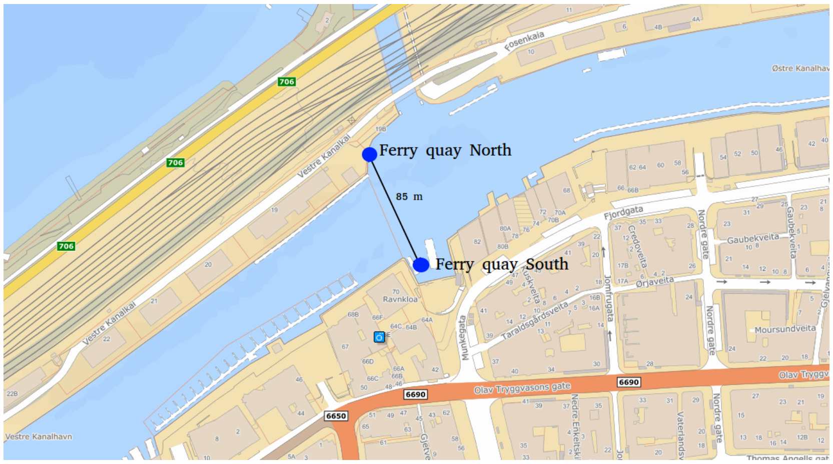

Figure 3: Operational environment of milliAmpere. Two ferry quays with approximately 85 meters across, giving a transit time of about 80 seconds. Small-boat harbour on both sides of the transit area.

3 Collision Avoidance

A common denominator for a lot of the methods developed in the field of COLAV for unmanned surface vessels is the scope of the environments. Often, the vessel guidance systems have a deliberate layer planning a global trajectory without concern of COLAV, and then leave the COLAV to the reactive layer [8], [9] allowing it to deviate from the global trajectory during evasive manoeuvres, and then converging to the trajectory when the situation has passed. In a lot of the cases this separation between deliberate trajectory planning and COLAV is a necessity, since the vessels are traveling great distance over a long period of time, and hence lacks knowledge of potential obstacles. Other methods are augmented with COLAV in the deliberate layer as well as the reactive layer [1]. A similarity in these methods is that the environments facilitates COLAV by allowing great deviations from the global trajectory without introducing any considerable risk or complications [10].

In the case of milliAmpere the situation for the COLAV systems is quite different. The physical environments are displayed in Figure 3. As one can see from the figure the scope of the transit is about 85 metes, which means that deviations of 25 meters from the straight trajectory between the two ferry quays will lead to an increase of trajectory length of about 20%, and a deviation of more than 25 meters might introduce complications with approaching the quay without excessive maneuvering of the vessel. This might lead to another complication that none of the mentioned methods consider; the passenger comfort. Due to these factors, a COLAV that is 100% reactive is unsuited for this problem due to its unpredictability.

The operational environment of milliAmpere does on the other hand introduce a couple of great advantages. Due to the short trajectory and hence short transit time, it is possible to get a relatively good understanding of what the traffic picture in the canal will look like in the time-scope of the transit. Another advantage of the canal shaped environment is the increased predictability of potential objects, since they tend to be either on their way in or out of the canal. The nature of the mission for the ferry also gives that the ferry will be at dock, which is considered a state free of collisions, at least once every 80 or so seconds. This gives the possibility of simply staying docked every time the traffic picture gives predictions of a problematic transit, awaiting better condition. These factors combined facilitates the use of a deliberate COLAV system as the main COLAV system of the ferry.

The strategy of staying docked awaiting better conditions in stead of laying out on a trajectory of evasive maneuvers rests on the assumption that the threshold for acceptable traffic is above the typical traffic in the canal in order to be an effective method. As long as this condition is met, the waiting strategy will improve passenger comfort, energy consumption, safety and also increase the charging time of the ferry, since it will use induction-charging whenever it is docked.

In the following subsections a suggestion to a deliberative COLAV system for the milliAmpere ferry will be proposed.

3.1 Structure

Figure 4 illustrates a suggestion to an architecture for a COLAV system with some of the vital supporting functions. The suggested structure is based on the three layer software architecture of autonomy [11].

The **Static Environment Map** keeps information about the static or slowly varying environments. This might be provided by ENC, supported by real time radar and Lidar data that catches changes in the static environment caused by either changes in water levels, currents or boats that are periodically docked in the environment. This module is under development in the milliAmpere project.

The **Object Detection Module** provides data on the rapid changes in the environment such as moving objects. The information might come from radar, lidar and camera data either from the ferry or from stations mounted on the dock or along the canal. This module is also under development in the milliAmpere project.

The **Executive Layer** uses information about the trajectory and mission to generate reference signals for the reactive layer. The executive layer generates a heading reference by a aligning the heading of the vessel with the velocity vector reference.

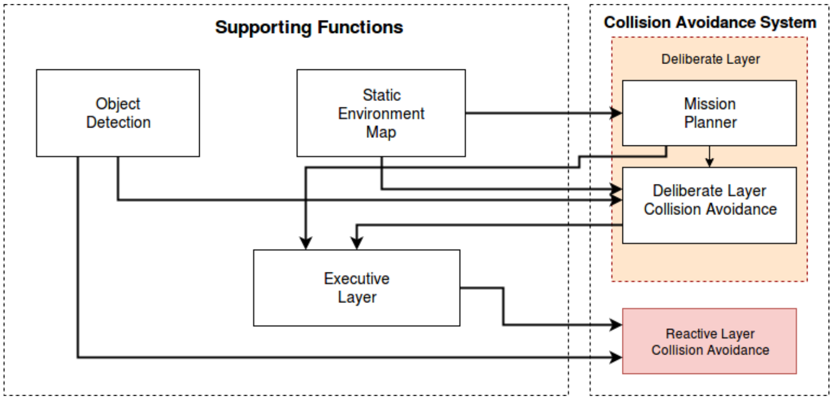

Figure 4: Illustration of structure of a COLAV system with supporting functions [1].

The **Mission Planner** is the top layer of the structure, and part of the deliberate layer. The mission planner handles the transit missions of the ferry. When the ferry is requested to transit from one quay to the other, it is the mission planner that generates the path from the current location to the desired location. Note that the mission planner only gives a path, and not a trajectory, in the way that it provides a set of waypoints in $\{n\}$. The transit request set a time for earliest possible departure, but puts no restrictions on the time of arrival, nor a timestamp for any underway checkpoints. Note that the mission planner only takes into account the information about the static environments, and not any moving objects in the canal.

The **Deliberate Layer** of the COLAV algorithm uses the path from the mission planner, as well as the information from the Object Detection Module about the moving objects in the environment to generate a trajectory that connects the suggested waypoits in a way that avoids collision during the transit. This is done by generating a velocity profile in $\{n\}$ that is fed to the Executive Layer. Note that the Deliberate Layer does not generate a heading reference, this is done by the Executive Layer by aligning the desired heading with the velocity vector. A suggested method for the Deliberate Layer will be presented later in this section.

The **Reactive Layer** of the COLAV tracks the suggested trajectory from the deliberate layer by using a trajectory tracking control system. At the same time it reacts to real time data from the Object Detection Module by making adjustments to the trajectory in order to avoid collision and ensure a certain level of safety.

3.2 Trajectory Planning

In this project the trajectory planning problem is concerned with finding a trajectory in space and time $S(t) \in \boldsymbol{R}^2$ that the ferry can track in order to get from the current position $\boldsymbol{P}_0 = [x_0, y_0]^T = [N_0, E_0]^T$ to a desired position $\boldsymbol{P}_d = [x_d, y_d]^T = [N_d, E_d]^T$. In addition to this, the trajectory should be planned in a way that ensures the ferry not to collide into any static or moving objects.

Here, a method of finding a collision free trajectory is separated into two subproblems, based on [12]. First finding a path that does not collide with any static object, and then finding a velocity profile along that path that ensures no collision with any potential moving objects.

3.2.1 Considerations

Since the trajectory is to take the ferry from one side of the canal to the other, the trajectory should start in a point just outside of one quay, and end in a point just outside of the other quay. This would allow the ferry to get close enough to the quay, so that the docking algorithm can take over. Other than this, the choice of trajectory should take into account the following factors:

- **Predictability:** In order for the passenger to feel safe, the trajectory should be predictable. This will make it easier for the passenger to observe the ferry in the environments and be assured that the crossing is secure and well planed.
- **Safety:** The trajectory should be planned in a way that maximizes the safety-factor of the crossing. This be, for example keeping an adequate distance to stationary object, waiting for high traffic or high velocity vessels to pass before starting the transit, as well as keeping in line with the COLREGs [13] in order to be predictable to other autonomous and non autonomous vessels.
- **Comfort:** The trajectory should be planned in a way that minimizes the acceleration and yaw rate, in order to increase passenger comfort and reduce risk of standing passenger loosing balance as well as motions sickness.
- **Efficiency:** The trajectory should take energy consumption into account and plan a trajectory that is no longer than necessary. There is also possibility to account for environmental factors such as wind and current.
- **Simplicity:** The trajectory should be as simple as possible, but not simpler.

In this particular case, the area in the canal between the two quays is free of any permanent obstacles, and hence a trajectory that resembles a straight line between the two docking stations would simplify the problem into only considering moving objects. A straight line is a very predictable trajectory, makes the crossing as short as possible and is arguably one of the simplest trajectories there is. Limitations associated with committing to a straight line is slowly moving, or temporally static objects located along the line will introduce problems for the planner. Also, in some cases, the trajectory might be substantially longer in duration, or suffers a reduction in one or more of the factors above compared to a trajectory that is allowed to deviate from the straight line.

The problem of finding the path is not addressed further in this project, but will in general be handled by the **Mission Planner** in Figure 4. In the following, we present a way of generating a trajectory that does not collide with any moving objects, given some assumptions about the behaviour of the objects. The trajectory is generated from a straight path, or a path made up from straight line segments, which is to make up the deliberate part of the COLAV system presented in the preceding sections. This method can also be generalized to smooth paths by altering the way the moving objects are transformed into the $time \times path$ domain.

3.2.2 Moving Objects

In the following, it is assumed that the moving objects in the canal close to the transit area is detectable by the sensor systems available to the ferry. It assumes that the objects are detected with a position, and velocity vector, as well as a size estimate. Further it is assume that due to the formation of the canal, moving objects will keep a somewhat steady course and velocity.

3.2.3 Object Representation

In order to make an algorithm that avoids colliding into objects, a representation of the objects is needed. The representation needs to include the area spanned by the object body in a topside view. An easy way of modeling this is to represent every object by a set of points giving the corners of a polygon, where when the sides of the polygon is drawn up, the whole projection of the vessel will be enclosed by the polygon. With this representation the polygon can be considered as a forbidden region that is moving around in the xy plane.

Figure 5 illustrates a way of making the polygons so that they also incorporate COLREGs [13]. By extending the polygon in front of the objects coming from the starboard side of the ferry, and extending to the rear of objects coming from the port side of the ferry the trajectory planning algorithm will favour trajectories that follow COLREGs rule 15.

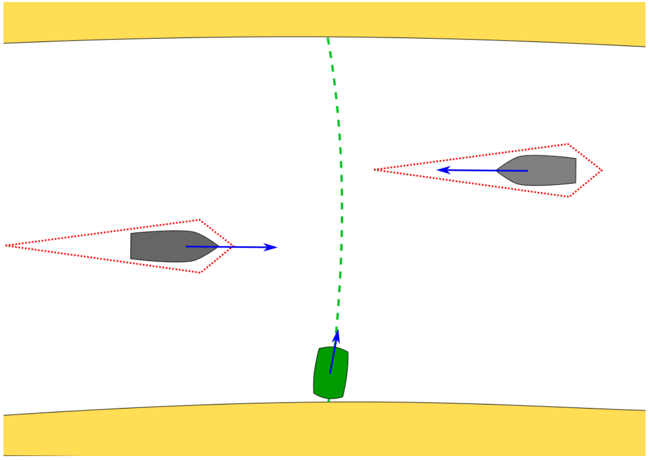

Figure 5: Moving objects in the canal (grey) with a velocity vector (blue) and a forbidden region (red). The ferry (dark green) and the desired path across the canal (light green). The forbidden region extends in front of the vessels approaching from the starboard side, and behind the vessels approaching from the port side in order to facilitate COLREGs.

3.2.4 Path Parameterization

A straight line path P between the points $P_{start} = [x_{start}, y_{start}]^T$ and $P_{end} = [x_{end}, y_{end}]^T$ can be parameterized by

$$P := \frac{x - x_{start}}{a} = \frac{y - y_{start}}{b} \tag{11}$$

$x \in [x_{start}, x_{end}]$, $y \in [y_{start}, y_{end}]$ where the length of the path is

$$l = \sqrt{(x_{end} - x_{start})^2 + (y_{end} - y_{start})^2} \tag{12}$$

By choosing a and b as

$$a = \frac{(x_{end} - x_{start})}{l} \tag{13}$$

$$b = \frac{(y_{end} - y_{start})}{l} \tag{14}$$

where l is the length of the path, the parameterization of the path has the unit *meters*, which proves to be intuitive and beneficial at a later stage.

3.2.5 Object Transformation

With a parameterization of the path, the observed objects can be transformed onto the $path \times time$ domain. For this, it it assumed that the objects will keep a steady course and velocity from the point of detection until it has crossed the path. From this, the time and position of where the object will intersect the path can be estimated. All objects with a size greater than zero will use some time crossing the path, and hence create an area of intersection in the $path \times time$ domain.

Since all objects can be represented by four points corresponding to the four corners of a polygon that is fully enclosing the vessel. The problem of finding the hyper-plane the object projects onto the $path \times time$ space is turned into four equal linear problems that can be solved independently.

The position $[x(t), y(t)]$ of an object in a plane as a function of time can be described by the initial position and velocity-vector at last detection, as well as the time since last detection,

$$x(t) = x_{obj} + v_x(t - t_{obj}) \tag{15}$$

$$y(t) = y_{obj} + v_y(t - t_{obj}) \tag{16}$$

where x_{obj} and y_{obj} is the position of the object at detection at $t = t_{obj}$ By substituting x and y from (11) for $x(t)$ and $y(t)$ in (15)-(16) we get

$$Pa - x_{start} = x_{obj} + v_x(t - t_{obj}) \tag{17}$$

$$Pb - y_{start} = y_{obj} + v_y(t - t_{obj}) \tag{18}$$

The equations can be solved to get a point $[P, t]$ for each corner of the rectangle in the Euclidean space, to get the four corners defining the object in the $path \times time$ space. By doing this for each observed object, a map of all objects projection onto the $path \times time$ space can be made. Figure 14 displays such a projection for the objects listen in Table 5.

3.2.6 Node Representation

With the objects represented in the $path \times time$ space, finding a collision free trajectory can be viewed as finding a graph from the point $[P, t] = [0, 0]$ to the line represented by $P = l$ that does not intersect with the area spanned by the representation of the objects. By making the corners of the objects, as well as the starting point into nodes, and creating vertices between the nodes, this problem can be solved by graph search algorithms. For the the algorithms to be able to find a trajectory to the end of the path, there has to be reachable end-nodes at $P = l$. To provide this, end nodes can be added for each of the objects and start nodes. By setting the time for the end nodes as a function of the desired transit velocity of the ferry according to (19), we facilitate the graph search algorithm to finding trajectories with desired transit velocity. In the equation l is trajectory length, and v_{des} is the desired transit velocity.

$$t_{end\_node} = t_{start\_node} + \frac{l - p_{start\_node}}{v_{des}} \tag{19}$$

3.2.7 Vertices and Weights

For this problem to be solved by a node search algorithm, it needs to be represented as a set of vertices and weights. The vertices are straight line segments connecting two nodes, and represents a potential sub-trajectory. The weights are values connected to each vertex representing the cost of using that sub-trajectory. When the nodes are set, the vertices and weights can be generated. Since the problem is a physical one, not all vertices are feasible. For a vertex to be feasible in this problem, there is three criteria that has to be met.

- The end node is later in time than the start node
- The vertex does not represent a velocity with absolute value above V_{max}
- The vertex does not pass through any of the objects

where $V_{max} > 0$ is the maximum desired velocity for the ferry. The first two criteria are trivial to check. The third criteria is solved by by the following approach: Let $\boldsymbol{V}$ be the set of vertices fulfilling the first two criteria, and $\boldsymbol{O}$ be the set of all vertices representing edges of objects. All vertices in $\boldsymbol{V}$ and $\boldsymbol{O}$ are represented as line P_{vertex} on the format given by equation (20). Then for all vertices in $\boldsymbol{V}$, find the intersection point $(p_{intersect}, t_{intersect})$ with each of the vertices in $\boldsymbol{O}$. If any of the intersection points give correspond to an $p_{vertex} \in [0, 1)$ for both vertices, remove the vertex from $\boldsymbol{V}$.

$$P_{vertex} = \frac{p - p_{start\_node}}{p_{end\_node} - p_{start\_node}} = \frac{t - t_{start\_node}}{t_{end\_node} - t_{start\_node}} \tag{20}$$

Following, the set V can be further reduced. Since we the vertices in $\boldsymbol{V}$ are directed, any node, except for the starting node, that does not have a vertex pointing to it, is an unreachable node. And hence all vertices pointing out from that node can be removed form the set $\boldsymbol{V}$. This is done in an iterative manner until no further reduction of $\boldsymbol{V}$ is possible.

The weight for each vertex can be set based on multiple objectives. For this case the weight is simply set as the length in time for each vertex, namely $W = (t_{end\_node} - t_{start\_node})$, since the objective is to get the ferry along the path in transit velocity.

3.2.8 Graph Search

With the set of vertices and weights, a graph search can be done. For this, the built in matlab function *shortestpath()* is used. The function takes the vertices, weights, start-node and end-node as arguments, and returns a list of nodes that makes up the shortest paths from the start-node to the end-nodes. Firstly the set E of end nodes is constructed, then the shortest path from the start-node to every node in E is calculated. The function returns a set of node lists that

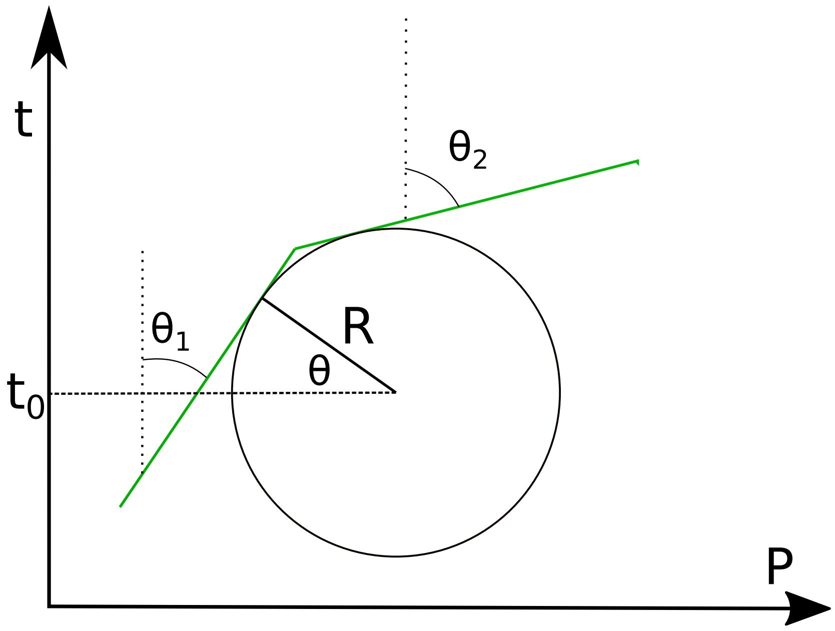

Figure 6: Two sub trajectories in path time space (green), that can be connected by a circle segment with a minimum radius. to assure a saturated acceleration.

contains nodes making up a collision free path form the starting point to the end of the path. If the list comes back empty, there is no collision free path from the start-node to any of the end-nodes. This is done for every start node.

3.2.9 Continuous Velocity Profile

The waypoints make up a trajectory consisting of sub trajectories of constant velocities. For trajectories with more than two waypoints, the trajectory will have a step in velocity between sub-trajectories, corresponding to infinite acceleration. This is unphysical, and will cause problems for the trajectory following control system, and give unwanted deviations from the trajectory at the link between each sub-trajectory. The steps in acceleration can also cause displeasure for the passengers, and is in violation with the list of considerations for trajectory planning. Special care has to be taken when solving this problem, since altering the trajectory might compromise the collision free path. A suggested solution to the problem is inspired by *Dubins Path* , with the line segments connected by circle segments to create a smooth path [14]. Since Dubins Path was created for the two dimensional euclidean space, some considerations has to be taken to use the concepts in the $path \times time$ space. It is desirable to smooth the trajectory in a way that saturates the acceleration to a value a_{max}. The velocity of the trajectory is given

$$v = \frac{\Delta p}{\Delta t} = \frac{dsin(\theta)}{dcos(\theta)} = tan(\theta) \tag{21}$$

where θ is the angle from the time axis to the trajectory tangent. This only holds if the path has the property stated in equations (11)-(14). The acceleration is given as the time derivative of the velocity

$$a = \frac{\dot{\theta}}{cos^2(\theta)} \tag{22}$$

From Figure 6 we can see that the relation between θ and time becomes

$$t = Rsin(\theta) + t_0 \tag{23}$$

By differentiating with respect to time, and solving for $\dot{\theta}$ we get the relation

$$\dot{\theta} = \frac{1}{Rcos(\theta)} \tag{24}$$

By combining (22) and (24), and inserting the the maximum desired acceleration a_{max} for a as well as the biggest angle of θ_1 and θ_2 namely $\theta_{max} = max(\theta_1, \theta_2)$, we get the minimum radius R_{min} for the circle segments connecting two sub-trajectories in order make a trajectory that does not violate the acceleration saturation. The minimum radius becomes

$$R_{min} = \frac{1}{a_{max}cos^3(\theta_{max})} \tag{25}$$

With this radius, the trajectory can be remapped in the area combining two sub trajectories in the *path* × *time* space, to give a continuous velocity profile with an acceleration saturation [14].

3.3 Trajectory Evaluation

With the velocity profile calculated, the trajectories from can be evaluated in order to fint the optimal trajectory. The evaluation is done with regards to

- Transit Time: The difference in transit-time t_{TT} from the shortest transit-time of the trajectories to the transit-time of the trajectory in evaluation, given in seconds.
- Acceleration: The integrated absolute value, a_{sum}, of the body acceleration $a_{abs} = \sqrt{\dot{u}_{ref} + \dot{v}_{ref}}$
- Time to arrival: The difference in arrival-time t_{AT} from the earliest arrival-time of the trajectories to the arrival-time of the trajectory in evaluation, given in seconds.

The score is calculated by the cost function

$$J_{trj} = K_{TT}t_{TT} + K_{acc}a_{sum} + K_{AT}t_{AT} \tag{26}$$

where K_{TT}, K_{AT} and K_{acc} gives the cost of the transit time, arrival time and body acceleration respectively. When all possible trajectories are evaluated, the trajectory of lower cost is chosen, and a transit can begin.

4 Trajectory Tracking

For the ferry to be able to follow the trajectory suggested by the trajectory generation module, it needs to have a control system capable of trajectory tracking. Since the vessel is fully actuated the options for trajectory tracking are many. The criteria for the trajectory tracking are not as comprehensive as the trajectory generation, but the system needs to be robust, stable, and keep the tracking error given by (10) within certain limits in order to ensure a collision free crossing. In this section three approaches to trajectory tracking are presented; a model based reference feed forward with position feed back, a line of sight guidance method with surge reference feed forward and along path distance feedback, and a MPC.

4.1 Thrust Allocation

The thrust allocation system on the ferry is controlling the two azimuth thrusters. The module takes inn a 3DOF force reference ,$\boldsymbol{\tau}_{ref} = [\tau_{u\_ref}, \tau_{v\_ref}, \tau_{r\_ref}]^T$, and calculates the optimal distribution of force between the two actuators according to a cost function. The algorithm uses data from a *Bollard Pull Test*, to estimate the actuator setpoints corresponding to a certain force. The thrust allocation module is described further in the section on simulator design. Since the interface to the thrust allocation system is a 3DOF force reference, the output from the following suggested trajectory tracking control systems will be a 3DOF force vector, $\boldsymbol{\tau} = [\tau_u, \tau_v, \tau_r]^T$, and hence the systems will not consider the dynamics of the thrusters at all.

4.2 Reference Model

The reference model used in the design of the control systems is from a scaled model of C/S Inocean Cat I Drillship. The model is nonlinear 3DOF model in surge, sway and yaw, and is further described in Section 5.2.

4.3 Reference Feed Forward

Since a 3DOF reference is generated, the vessel model can be used to estimate the 3DOF forces needed to follow the reference. By using (7)-(9), the following feed forward law can be generated

$$\boldsymbol{\tau}_{FF} = \boldsymbol{M}\dot{\boldsymbol{\nu}}_{ref} + (\boldsymbol{C}(\boldsymbol{\nu}_{ref}) + \boldsymbol{D}(\boldsymbol{\nu}_{ref}))\boldsymbol{\nu}_{ref} \tag{27}$$

where $\dot{\boldsymbol{\nu}}_{ref}$ is the body acceleration reference, and $\boldsymbol{\nu}_{ref}$ is the body velocity reference.

The law does not take into account the current, wind or waves, and hence a feedback law is needed to eliminate the error caused by disturbance and modeling error. Since the objective is to track a trajectory, a feedback law on the state of the tracking is beneficial, and the feedback on $\boldsymbol{\eta}$ is used to calculate the trajectory tracking error $\boldsymbol{e}$ from (10). With this error, the following feedback control law is generated

$$\boldsymbol{\tau}_{FB} = \boldsymbol{R}(\psi)^T(\boldsymbol{K_p}\boldsymbol{e} + \boldsymbol{K_d}\frac{d\boldsymbol{e}}{dt} + \boldsymbol{K_i}\int_{t_0}^{t}\boldsymbol{e}dt) \tag{28}$$

where $\boldsymbol{R}(\psi)$ is the rotation matrix in (1).

Hence, the control law becomes

$$\boldsymbol{\tau}_{PID} = \boldsymbol{\tau}_{FF} + \boldsymbol{\tau}_{FB} \tag{29}$$

4.4 Line of Sight Controller

Another approach to trajectory tracking is a LOS based steering law with velocity and acceleration feed forward in surge, as well as along track error feedback. The steering law is based on [6].

The method uses the position and the reference position in the path-fixed frame, and hence we need to find a transformation matrix. This is done by first finding the angle of the path α_k relative North, in $\{n\}$, and is found by (30).

$$\alpha_k := atan2(y_{k+1} - y_k, x_{k+1} - x_k) \tag{30}$$

where $p_k = (x_k, y_k)$ and $p_{k+1} = (x_{k+1}, y_{k+1})$ is the previous and next waypoint on the path. Since we here consider a straight path from one quay to the other it can be considered as the start and end point of the trajectory respectively. With this angle, the coordinates of the vessel can be rotated to the path-fixed frame by (31) where the rotation matrix $\boldsymbol{R}_p(\alpha_k)$ is given by (32) and $\boldsymbol{p}(t)$ is the position of the vessel.

$$\boldsymbol{\epsilon}(t) = \boldsymbol{R}_p(\alpha_k)^T(\boldsymbol{p}(t) - \boldsymbol{p}_k) \tag{31}$$

$$\boldsymbol{R}_p(\alpha_k) = \begin{bmatrix} cos(\alpha_k) & -sin(\alpha_k) \\ sin(\alpha_k) & cos(\alpha_k) \end{bmatrix} \tag{32}$$

Here $\boldsymbol{\epsilon}(t) = [s(t), e_c(t)]^T$ where $s(t)$ is the along-track distance, and $e_c(t)$ is the cross-track error. Applying the same rotation to the vector from the path starting point to the current trajectory reference, namely $\boldsymbol{p}_{ref} - \boldsymbol{p}_k$, where $\boldsymbol{p}_{ref} = [N_{ref}, E_{ref}]^T$, we get $\boldsymbol{\epsilon}_{ref}$ with the reference along-track distance, and a reference cross track error that is zero by design. From this we can compute $\tilde{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon}_{ref} - \boldsymbol{\epsilon}$, that gives the along-track error e_a and the cross track error e_c. From this the trajectory tracking objective can be formulated as

$$\lim_{t \to \infty} \tilde{\boldsymbol{\epsilon}}(t) = 0 \tag{33}$$

The objective can be split into two sub-objectives where one is concerned with the cross-track error, and one is concerned with the along-track error. The cross-track error can be handled by the steering law (34) from [6] **Section 10.3.2**. The law is a saturated steering law where $K_p = 1/\Delta$ and Δ is the look-ahead distance, and $K_i > 0$.

$$\chi_r(e_c) = arctan(-K_p e_c - K_i \int_0^t e_c(\tau) d\tau) \tag{34}$$

The desired heading of the vessel can be calculated to be

$$\psi_d = \alpha_k + \chi_r(e_c) \tag{35}$$

and the PID control law (36) can be used to apply the torque N in the 3DOF force $\boldsymbol{\tau} = [X, Y, N]^T = [\tau_X, \tau_Y, \tau_N]^T$

$$N = \tau_N = -K_p\tilde{\psi} - K_d\dot{\tilde{\psi}} - K_i \int_0^t \tilde{\psi}(\tau) d\tau \tag{36}$$

The second objective, concerning the along track error e_a can be solved by a surge velocity and acceleration feed forward along with the feedback on the along track error.

$$X_{FB} = \tau_X = K_{pe_a} e_a + K_{de_a} \frac{de_a}{dt} + \int_0^t e_a(\tau) d\tau \tag{37}$$

$$X_{FF} = [1, 0, 0](\boldsymbol{M}\dot{\boldsymbol{U}}_{ref} + (\boldsymbol{C}(\boldsymbol{\nu}) + \boldsymbol{D}(\boldsymbol{\nu}))\boldsymbol{U}_{ref}) \tag{38}$$

where $\boldsymbol{U}_{ref} = [U_{ref}, 0, 0]^T$ and $U_{ref} = \sqrt{u_{ref}^2 + v_{ref}^2}$.

From this the following 3DOF control input can be formulated

$$\boldsymbol{\tau}_{LOS} = \begin{bmatrix} X_{FB} + X_{FF} \\ 0 \\ N \end{bmatrix} \tag{39}$$

4.5 Model Predictive Controller

A third approach to the trajectory tracking problem is a MPC [15]. MPC is a control strategy where the control action is found by solving a finite horizon open loop optimal control problem. The approach uses the current state of the plant along with a state-space representation of the plant to propose a solution for the next N control inputs, where the control inputs are found by minimizing a quadratic cost function

$$\boldsymbol{J} = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} + \frac{1}{2}\boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u} + \frac{1}{2}\boldsymbol{\Delta}\boldsymbol{u}^T\boldsymbol{R}_{\Delta}\boldsymbol{u} \tag{40}$$

by a quadratic programming algorithm. The cost function represents the objective of the optimization, where $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\boldsymbol{R}_{\boldsymbol{\Delta}}$ sets the cost for the tracking error, control input and change in control input, respectively. For the implementation of an MPC the system needs to be described on a discrete state space form.

4.5.1 State Space Form

By rearranging (7)-(9), the system can be represented in state space form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(\boldsymbol{x}(t))\boldsymbol{x}(t) + \boldsymbol{B}(\boldsymbol{x}(t))\boldsymbol{u}(t) \tag{41}$$

This is done by rearranging (7), and solving for $\dot{\boldsymbol{x}}(t) = \dot{\boldsymbol{\nu}}_r(t)$. This gives the system

$$\dot{\boldsymbol{\nu}}_r(t) = \boldsymbol{M}^{-1}(\boldsymbol{C}(\boldsymbol{\nu}_r(t))+\boldsymbol{D}(\boldsymbol{\nu}_r(t)))\boldsymbol{\nu}_r(t)+\boldsymbol{M}^{-1}(\boldsymbol{\tau}(t)+\boldsymbol{\tau}_{wind}(t)+\boldsymbol{\tau}_{wave}(t)) \tag{42}$$

where the $\boldsymbol{A}$ can be found to be

$$\boldsymbol{A}(\boldsymbol{\nu}(t)) = \boldsymbol{M}^{-1}(\boldsymbol{C}(\boldsymbol{\nu}_r) + \boldsymbol{D}(\boldsymbol{\nu}_r)) \tag{43}$$

and the $\boldsymbol{B}$ matrix becomes

$$\boldsymbol{B}(\boldsymbol{\nu}(t)) = \boldsymbol{M}^{-1} \tag{44}$$

4.5.2 Linearization

The system in (41) can be linearized about a point $\boldsymbol{x}(t_0)$ by substituting the time dependent system matrix $\boldsymbol{A}(\boldsymbol{x}(t))$ with $\boldsymbol{A}_{t_0} = \boldsymbol{A}(\boldsymbol{x}(t_0))$ and $\boldsymbol{B}(\boldsymbol{x}(t))$ with $\boldsymbol{B}_{t_0} = B(\boldsymbol{x}(t_0))$. The linearized system becomes

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_{t_0}\boldsymbol{x}(t) + \boldsymbol{B}_{t_0}u(t) \tag{45}$$

4.5.3 Discretization

The system in (45) can be discretized according to [16].The discretization method assumes that the input $\boldsymbol{u}(t)$ is behaves like a staircase function where the input is piecewise constant, that is $\boldsymbol{u}(t) = \boldsymbol{u}(kT) := \boldsymbol{u}(k)$ for $kT \leq t < (k+1)T$ for $k = 0, 1, 2, ...,$ and uses the general solution of the state space equation [16] to find the discretized solution

$$\boldsymbol{x}[t+1] = \boldsymbol{A}_d \boldsymbol{x}[t] + \boldsymbol{B}_d \boldsymbol{u}[t] \tag{46}$$

with

$$\boldsymbol{A}_d = e^{\boldsymbol{A}T}, \quad \boldsymbol{B}_d = \int_0^T e^{\boldsymbol{A}\tau} d\tau \boldsymbol{B} \tag{47}$$

4.5.4 Optimization Problem

With the linearized system in hand, the optimization problem can be set up. This regards setting up the cost function and the equality constraints for the optimization. In the approach used in this project thesis, the matlab function *mpcqpsolver()* is used to solve the quadratic optimization problem . The solver takes as input the $\boldsymbol{H}$ and $\boldsymbol{F}$ matrix from the cost function written in the form

$$\boldsymbol{J} = \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} + \boldsymbol{F} \boldsymbol{x} \tag{48}$$

as well as the inequality matrices

$$\boldsymbol{A}_{eq} \boldsymbol{x} = \boldsymbol{b}_{eq} \tag{49}$$

In (48) tje $\boldsymbol{H}$ and $\boldsymbol{F}$ matrix represents the cost priorities from $\boldsymbol{Q}$,$\boldsymbol{R}$ and $\boldsymbol{R_\Delta}$. The vector $\boldsymbol{x}$ is the combined states and control input of the system

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}[k+1] \\ \boldsymbol{x}[k+2] \\ \vdots \\ \boldsymbol{x}[k+N] \\ \boldsymbol{u}[k] \\ \boldsymbol{u}[k+1] \\ \vdots \\ \boldsymbol{u}[k+N-1] \end{bmatrix} \tag{50}$$

In the construction of $\boldsymbol{H}$ and $\boldsymbol{F}$ the objective of the trajectory tracking problem must be considered. The objective is to minimize the absolute value of the tracking error $\boldsymbol{x} - \boldsymbol{x}_{ref}$ with

$$\boldsymbol{x}_{ref} = \begin{bmatrix} \boldsymbol{x}[k+1]_{ref} \\ \boldsymbol{x}[k+2]_{ref} \\ \vdots \\ \boldsymbol{x}[k+N]_{ref} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{51}$$

By inserting the tracking error into the quadratic cost function along with a general $\boldsymbol{H}$ we get

$$\boldsymbol{J} = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_{ref})^T \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}_{ref}) \tag{52}$$

$$\boldsymbol{J} = \frac{1}{2}(\boldsymbol{x})^T \boldsymbol{H}(\boldsymbol{x}) + \boldsymbol{x}_{ref}^T \boldsymbol{H}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}_{ref}^T \boldsymbol{H}\boldsymbol{x}_{ref} \tag{53}$$

where the last term in (53) is a constant and can be omitted. This gives

$$\boldsymbol{J} = \frac{1}{2}(\boldsymbol{x})^T \boldsymbol{H}(\boldsymbol{x}) + \boldsymbol{x}_{ref}^T \boldsymbol{H}\boldsymbol{x} \tag{54}$$

which is on the same form as the cost function in (48) with $\boldsymbol{F} = \boldsymbol{x}_{ref}^T \boldsymbol{H}$ From the state and input vector $\boldsymbol{x}$, one can see that the cost matrix needs to be $\boldsymbol{H} = \boldsymbol{H}_1 + \boldsymbol{H}_2$, with $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$ given by (55) and (56) respectively in order for (48) to be equivalent to (40).

$$\boldsymbol{H}_1 = \begin{bmatrix} Q & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & Q & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & Q & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & Q & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & R & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & R & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & R & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & R \end{bmatrix} \tag{55}$$

$$\boldsymbol{H}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_{\boldsymbol{\Delta u}} \end{bmatrix} \tag{56}$$

$$\boldsymbol{R_{\Delta u}} = R_{\Delta u} \begin{bmatrix} 1 & -1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \tag{57}$$

The equality constraint matrices can be calculated by (46) by solving from the time $t = k$, to $t = k+N$ where N defines the prediction horizon. By rearranging the terms on one side we get

$$\boldsymbol{x}[k+n+1] - \boldsymbol{A}_d\boldsymbol{x}[k+n] - \boldsymbol{B}_d\boldsymbol{u}[k+n] = \boldsymbol{0} \tag{58}$$

for $n \in [1, N]$, and

$$\boldsymbol{x}[k+1] - \boldsymbol{A}_d\boldsymbol{x}_0 - \boldsymbol{B}_d\boldsymbol{u}[k] = \boldsymbol{0} \tag{59}$$

for the first step, where $\boldsymbol{x}_0$ is the current state of the system. From this we see that with $\boldsymbol{X}$ as given in (50) the $\boldsymbol{A}_{eq}$ and $\boldsymbol{b}_{eq}$ becomes (60) and (61) respectively.

$$A_{eq} = \begin{bmatrix} \boldsymbol{I} & 0 & 0 & \cdots & 0 & 0 & -\boldsymbol{B} & 0 & 0 & \cdots & 0 & 0 \\ -\boldsymbol{A} & \boldsymbol{I} & 0 & \cdots & 0 & 0 & 0 & -\boldsymbol{B} & 0 & \cdots & 0 & 0 \\ 0 & -\boldsymbol{A} & \boldsymbol{I} & \cdots & 0 & 0 & 0 & 0 & -\boldsymbol{B} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \boldsymbol{I} & 0 & 0 & 0 & 0 & ... & -\boldsymbol{B} & 0 \\ 0 & 0 & 0 & \cdots & -\boldsymbol{A} & \boldsymbol{I} & 0 & 0 & 0 & ... & 0 & -\boldsymbol{B} \end{bmatrix} \tag{60}$$

$$\boldsymbol{b}_{eq} = \begin{bmatrix} \boldsymbol{A}\boldsymbol{x}_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{61}$$

4.5.5 MPC controller

The control algorithm solves the optimization problem for each iteration. This is done by first calculating the linearized state space matrices given by (45) around the current state $\boldsymbol{x}(t_0) = [\boldsymbol{\eta}_0^T, \boldsymbol{\nu}_0]^T$, and then discretize the matrices according to (46) and (47). Subsequently, the equality matrices can be calculated. Further, the reference vector $\boldsymbol{x}_{ref}$ can be constructed from the planned trajectory, and the cost matrix $\boldsymbol{F}$ can be computed. The cost matrix $\boldsymbol{H}$ is constant and can be calculated offline.

When all matrices are calculated, the $\boldsymbol{x}_{opt} = mpcqpsolver(\boldsymbol{H}, \boldsymbol{F}, \boldsymbol{A}_{eq}, \boldsymbol{b}_{eq})$ is called. The $\boldsymbol{x}_{opt}$ state contains a combined state vector on the form (50) with states that minimize the cost function in accordance with the equality constraints. Further the control algorithm extracts the first control input $\boldsymbol{u}[k] = [0, 0, 0, X[k], Y[k], N[k]]^T$ and realizes the 3DOF force vector (62).

$$\boldsymbol{\tau}_{MPC} = [X[k], Y[k], N[k]]^T \tag{62}$$

This is done for for every timestep of the trajectory following control loop.

5 Simulator

Even though the ferry milliAmpere is fully functional and available in the harbour in the centre of Trondheim, it is somewhat time consuming to take it out for sea trials every time a new iteration of the COLAV system needs testing. A system under development rarely works on the fist try either, so the simulator can contribute in removing the obvious bugs in the system In addition to this, testing of COLAV algorithms under development might lead to uncomfortable situations, and hence it is beneficial to be able to test the system in a risk free simulator environment beforehand.

Due to this, a simulator was developed to facilitate the testing of COLAV systems in series with the trajectory following systems on the ferry. The simulator was developed in Matlab/Simulink and is based on a 3DOF horizontal plane maneuvering model as described by Fossen [6]. The simulator interfaces with the systems on the ferry through ROS messages, and uses the ROS interface in *MathWorks - Robotic System Toolbox* [17].

The simulator includes a thruster model for the two azimuth thrusters on the ferry. The thruster-model is based on data from a bollard pull test and will be explained further in the section on thruster model. The navigation node from the ferry is also included in the simulator. This node handles the data from the GPS systems and the IMU and makes this available for the rest of the ROS system.

5.1 Simulator Layout

An overview of the layout for the simulator is displayed in Figure 7. The green blocks represents the ROS interfaces that are either publishing or subscribing to messages from the ROS nodes in the ferry control system of milliAmpere. The green blocs make up the vessel model. The red block simulates the environmental forces.

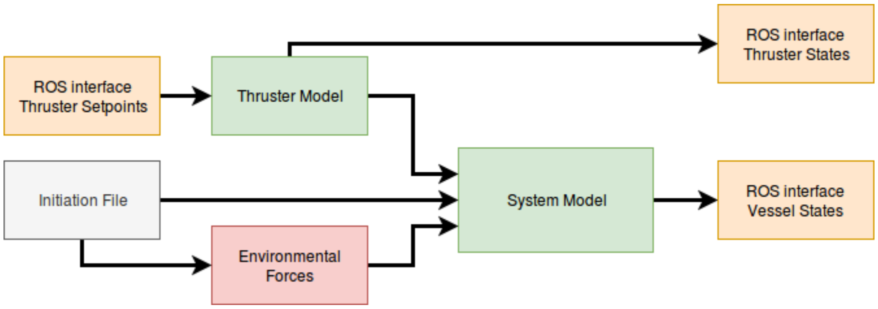
ROS interface Thruster Setpoints
Thruster Model
ROS interface Thruster States
Initiation File
System Model
ROS interface Vessel States
Environmental Forces

Figure 7: Layout showing an overview of the the simulator implemented in Matlab/Simulink with ROS interfaces interfacing with the same ROS-nodes that is running on the milliAmpere OBC

5.2 3 DOF Model

At the time of simulator development there was no model of the ferry, so the simulator is based on a model of the C/S Inocean Cat Drillship [2]. The vessel mode is from a 1:90 scale replica of a supply ship with a length of $L = 2.578m$

The model is a 3DOF vessel model equal to (3)-(9). The system matrices is given in the following, and the model parameters are given by Table 1. The rigid bod and added mass matrix is given by

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & I_z \end{bmatrix} \tag{63}$$

$$\boldsymbol{M}_{A} = \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -N_{\dot{v}} & -N_{\dot{r}} \end{bmatrix} \tag{64}$$

The Coriolis and centripetal matrix is given by

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -m(x_g r + v) \\ 0 & 0 & mu \\ m(x_g r + v) & -mu & 0 \end{bmatrix} \tag{65}$$

$$\boldsymbol{C}_{A}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -c_{A,13}(\boldsymbol{\nu}) \\ 0 & 0 & c_{A,23}(\boldsymbol{\nu}) \\ c_{A,13}(\boldsymbol{\nu}) & -c_{A,23}(\boldsymbol{\nu}) & 0 \end{bmatrix} \tag{66}$$

where the terms in the Coriolis matrix is given by

$$c_{A,13}(\boldsymbol{\nu}) = -Y_{\dot{r}} r - Y_{\dot{v}} v \tag{67}$$

$$c_{A,23}(\boldsymbol{\nu}) = -X_{\dot{u}} u \tag{68}$$

Parameter	**value**	**Parameter**	**value**
X_u	-5.35	N_{rv}	0.08
$X_{\|u\|u}$	0	N_{vr}	0.08
X_{uuu}	-19.6312	Y_{rv}	-0.805
Y_v	-10.16	Y_{vr}	-0.845
$Y_{\|v\|v}$	-0.8647	$X_{\dot{u}}$	-10
Y_{vvv}	-681.1745	$Y_{\dot{v}}$	-105
Y_r	-7.25	$Y_{\dot{r}}$	-0.525
$Y_{\|r\|r}$	-3.45	$N_{\dot{v}}$	-0.157
Y_{rrr}	0	$N_{\dot{r}}$	-3.450
N_v	0	N_{ur}	-0.525
$N_{\|v\|v}$	-0.2088	N_{uv}	95
N_{vvv}	0	Y_{ur}	10
N_r	-14.55	x_g	0.0375
$N_{\|r\|r}$	-9.9597	m	127.92
N_{rrr}	-0.3101	I_z	61.967

Table 1: Model parameters for the 3DOF model used in the simulator [2].

The damping matrix $\boldsymbol{D}(\boldsymbol{\nu})$ is given by (69) where the matrix elements are given by (70)-(74).

$$\boldsymbol{D}(\boldsymbol{\nu}) = \begin{bmatrix} D_{11}(\boldsymbol{\nu}) & 0 & 0 \\ 0 & D_{22}(\boldsymbol{\nu}) & D_{23}(\boldsymbol{\nu}) \\ 0 & D_{32}(\boldsymbol{\nu}) & D_{33}(\boldsymbol{\nu}) \end{bmatrix} \tag{69}$$

$$D_{11}(\boldsymbol{\nu}) = -X_u - X_{|u|u}|u| - X_{uuu}u^2 \tag{70}$$

$$D_{22}(\boldsymbol{\nu}) = -Y_v - Y_{|v|v}|v| - Y_{|r|v}|v| - Y_{vvv}v^2 \tag{71}$$

$$D_{23}(\boldsymbol{\nu}) = -Y_r - Y_{|r|r}|r| - Y_{|v|r}|v| - Y_{rrr}v^2 - Y_{ur}u \tag{72}$$

$$D_{32}(\boldsymbol{\nu}) = -N_v - N_{|v|v}|v| - N_{|r|v}|r| - N_{vvv}v^2 - N_{uv}u \tag{73}$$

$$D_{33}(\boldsymbol{\nu}) = -N_r - N_{|r|r}|v| - N_{|v|r}|v| - N_{rrr}r^2 - N_{ur}u \tag{74}$$

The simulator solves

$$\boldsymbol{\nu}(t) = \int_{t_0}^{t} \boldsymbol{M}^{-1}(\boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu})dt + \boldsymbol{\nu}(t_0) \tag{75}$$

$$\boldsymbol{\eta}(t) = \int_{t_0}^{t} \boldsymbol{R}(\psi)\boldsymbol{\nu}dt + \boldsymbol{\eta}(t_0) \tag{76}$$

where the initial condition $\boldsymbol{\nu}(t_0)$ and $\boldsymbol{\eta}(t_0)$ is set in the initiation file of the simulator, and

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{thruster} + \boldsymbol{\tau}_{external} \tag{77}$$

is the forces on the vessel where $\boldsymbol{\tau}_{thrusters}$ come from $\boldsymbol{\tau}_{PID}$, $\boldsymbol{\tau}_{LOS}$ or $\boldsymbol{\tau}_{MPC}$ depending on the choice of control strategy, and $\boldsymbol{\tau}_{external}$ come from the external forces that will be mentioned in a later subsection.

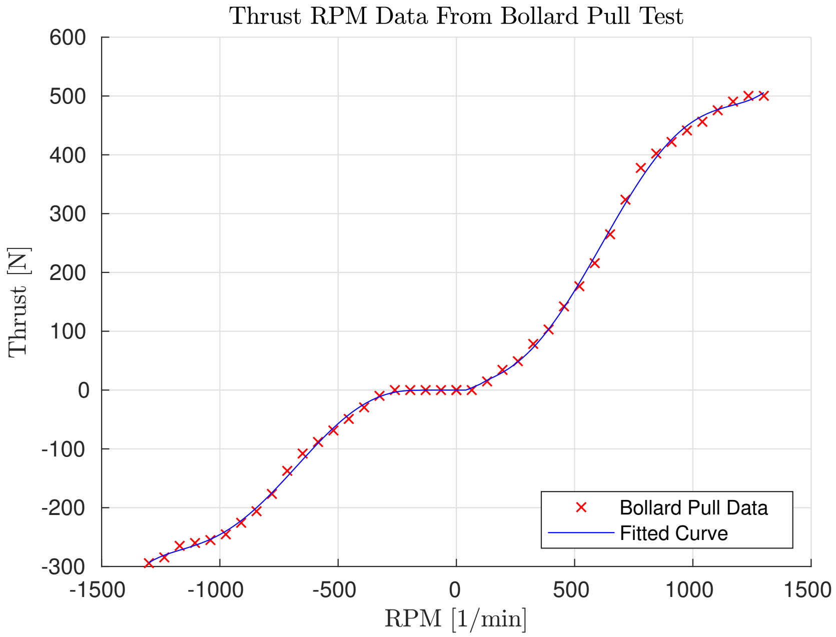

Figure 8: Curve-fitting of the data from the bollard pull test in Table 2.

5.3 Thruster Model

The interface between the thrusters and the rest of the ferry control is limited to four signal values. The thrust allocation system sends a rotational velocity setpoint, $omega_{set} \in [-1000, 1000]$ where the two edge cases correspond to min and max rotational velocity respectively. In addition, the thrust allocation send angle setpoint, corresponding to the desired angle of the thruster relative the x-axis of the vessel. The thruster system reads the setpoints for angle and rotational velocity. The commands are realized by the motor controllers to the best ability, and the actual RPM and thrust angle of the azimuth thruster are published back to the thrust allocation system.

The mapping between the RPM thrust is based on data form a bollard pull test performed on the ferry 06.06.2018 by people in the milliAmpere project. The data can be seen in Table 2. The data was imported to matlab and a 5th degree polynomial was curve-fitted to the thruster data. The fitted curve can be seen in Figure 8. From the polynomial, an evenly distributed data set of thrust and RPM was calculated, for use in a linear look-up table. The fitted curve and the brakepoints can be seen in Figure 9. The dynamics of the azimuth angle is also modeled based on test data from the same test. Data from running the azimuth angle with step inputs are presented in Table 3. Since this is all

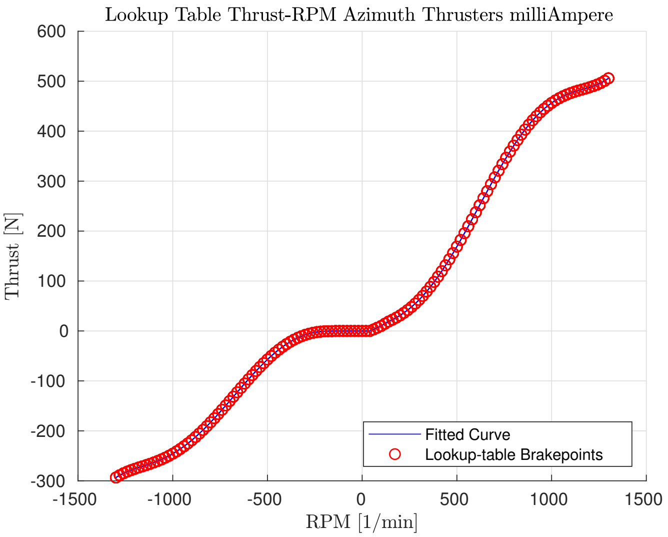

Figure 9: Data for the look-up table for the thruster model in Simulink. The table gives the relation between RPM and thrust as well as the curve fit to the bollard pull data.

Motor Speed [%]	**Positive Thrust** [N]	**Negative Thrust** [N]
5.0	0	0
10.0	29.0	0
15.0	59.0	0
20.0	69.0	0
25.0	88.0	9.8
30.0	122.0	29.0
35.0	166.0	49.0
40.0	200.0	69.0
45.0	222.0	88.0
50.0	277.0	111.0
55.0	322.0	144.0
60.0	399.0	199.0
65.0	411.0	211.0
70.0	433.0	233.0
75.0	466.0	266.0
80.0	477.0	277.0
85.0	499.0	277.0
90.0	500.0	288.0
95.0	500.0	299.0
100.0	500.0	300.0

Table 2: Results from the bollard pull test performed with the milliAmpere ferry 06.06.2018.

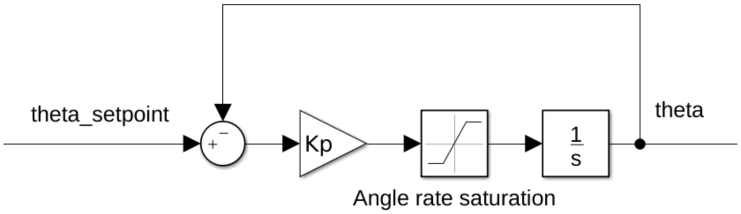

Figure 10: Azimuth angle modeling in Simulink. The angular velocity is set proportional to the error, and saturated to max angular rate. The saturation value is found from the data in Table 3.

Step Start[deg]	**Step End**[deg]	**Time** [s]
0	360	9
0	180	4
0	90	2
360	0	9
270	0	6
180	0	4
90	0	1.9

Table 3: Results from step input test of azimuth angle on the thruster system. The table gives start value and end value of the step input, as well as the corresponding response time of the azimuth thruster.

the data available it is assumed that the thruster rotates at a constant angular velocity until it is close to the setpoint, and then ramps down the velocity until the azimuth angle corresponds with the setpoint. The thruster is modeled with a velocity proportional to the error

$$e_\theta = \theta_{set} - \theta \tag{78}$$

where θ_{set} is the azimuth angle setpoint, and θ is the actual azimuth angle. A saturation is added to match the maximum angular rate of the thrusters which is found from the test data in Table 3. The azimuth angle model from Simulink is displayed in Figure 10.

5.4 Thruster Force Transformation

The two thrusters are modeled independent of each other and generates independent forces F_{front} and F_{rear} at the azimuth angles θ_{front} and θ_{rear} respectively. The vessel model uses 3DOF thruster forces $\boldsymbol{\tau}_{thruster}$, and hence a transformation needs to be done. The position of the thrusters can be seen in Figure 11.

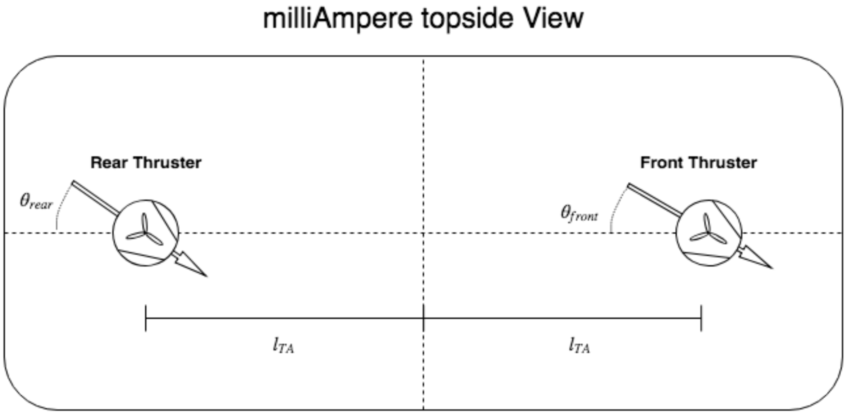

Figure 11: Topside view of the thruster layout on milliAmpere. The ferry is symmetrical, and hence the front and rear thruster arm is the same.

Both thrusters are placed on the front-aft centerline of the vessel, symmetrical about the midship beam. The relationship between the thruster forces and angles and the 3DOF force is

$$\boldsymbol{\tau}_{thrusters} = \begin{bmatrix} cos(\theta_{front}) & cos(\theta_{rear}) \\ sin(\theta_{front}) & sin(\theta_{rear}) \\ -l_{TA}sin(\theta_{front}) & l_{TA}sin(\theta_{rear}) \end{bmatrix} \begin{bmatrix} F_{front} \\ F_{rear} \end{bmatrix} \tag{79}$$

where l_{TA} is the front and rear thruster arm.

5.5 External Forces

The simulator is also augmented with the possibility to add external forces $\boldsymbol{\tau}_{external}$. Both static and dynamical forces can be added. The forces are defined in the $\{n\}$ frame as $\boldsymbol{F}_{external}$, and is rotated to the $\{b\}$ frame by

$$\boldsymbol{\tau}_{external} = \boldsymbol{R}(\eta)^T \boldsymbol{F}_{external} \tag{80}$$

where $\boldsymbol{R}(\eta)$ is given by (1).

5.6 ROS Interface

The simulator interfaces with the control system though ROS messages. Blocks from the *Robotic System Toolbox* support importation of custom messages, and make the adaptation seamless. The actuator setpoints are published by the thruster allocation node, and is subscribed to by a block in the simulator in the same way the actuator controllers do. The simulator also has a sensor system module, where it publishes sensor data on the same topics and message format as the navigation node in the ferry control system. This makes for easy switching between testing the control system in the simulator and the experimental platform.

6 Simulation Results and Discussion

6.1 Trajectory Planning

The trajectory planning algorithm is tested by three traffic situations with increasing traffic. All three situations are shown in Figure 12. In the figure, the planned path is marked in green, the canal banks in dashed blue, and the initial position of objects, with a vector showing the heading of the objects in red. The path of the ferry starts on the lower bank in the point $P_{start} = [x, y]^T[10, 10]^T$ and ends on the higher bank in the point $P_{end} = [x, y]^T = [100, 30]^T$. The information about the objects for Situation 1, Situation 2 and Situation 3 is given in Table 4, Table 5 and Table 6 respectively. For each of the situations introduces in Figure 12 , the trajectory planning algorithm has tried to plan a trajectory. Figure 13-15 shows the collection of vertices that fulfilled the three conditions from Section 3.2.7, and hence qualifies as potential sub-trajectories. The figures also contains the projection of the moving objects onto the $path \times time$ domain.

From the results in figures 13-15 it is evident that the trajectory planner is able to find a trajectory that is collision free according to the data available, given the mentioned assumptions about the moving objects. An increasing traffic picture increases the size of the problem by adding more nodes and vertices, but does not make the problem any more complicated, and hence the same method can be used, independent of the traffic picture.

Object	**Position** $(x_0, y_0)[m]$	**Heading** $[rad]$	**Velocity** $[m/s]$	**Length** [m]	**Width** [m]
Obj_{11}	$[24, -6]$	$\pi/2.1$	1.0	4	2
Obj_{12}	$[70, 70]$	$-\pi/1.9$	0.9	4	2

Table 4: The moving objects in Situation 1. The table contains the object data that is available to the trajectory planning.

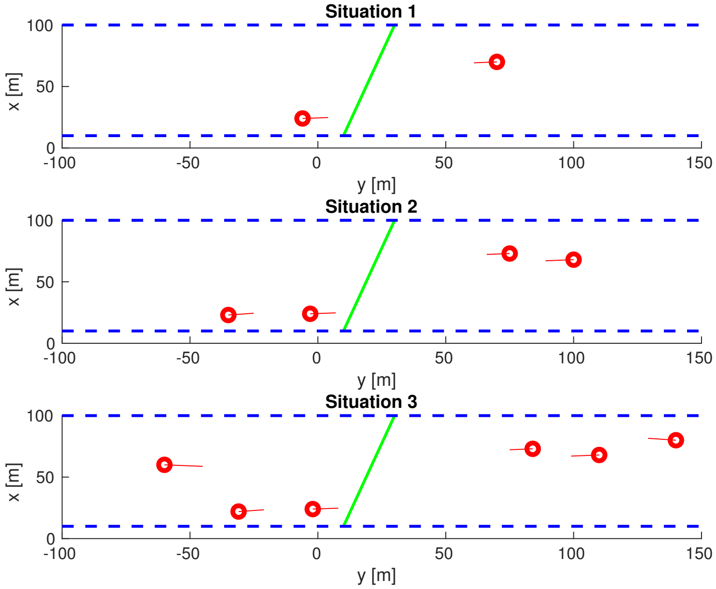

Figure 12: Situation overview of the three test situations. Blue line defies the canal banks, green line is the planned path and red dots are the detected objects with a velocity vector scaled up by a factor of 10 for visibility. The path starts on the lower bank and ends on the higher bank.

Object	**Position** $(x_0, y_0)[m]$	**Heading** $[rad]$	**Velocity** $[m/s]$	**Length** [m]	**Width** [m]
Obj_{21}	$[24, -3]$	$\pi/2.1$	1.0	4	2
Obj_{22}	$[23, -35]$	$\pi/2.2$	1.0	6	3
Obj_{23}	$[68, 100]$	$-\pi/1.9$	1.1	6	3
Obj_{24}	$[73, 75]$	$-\pi/1.9$	0.9	4	2

Table 5: The moving objects in Situation 2. The table contains the object data that is available to the trajectory planning.

Object	**Position** $(x_0, y_0)[m]$	**Heading** $[rad]$	**Velocity** $[m/s]$	**Length** [m]	**Width** [m]
Obj_{31}	$[24, -2]$	$\pi/2.1$	1.0	4	2
Obj_{32}	$[22, -31]$	$\pi/2.2$	1.0	6	3
Obj_{33}	$[68, 110]$	$-\pi/1.9$	1.1	6	3
Obj_{34}	$[73, 84]$	$-\pi/1.9$	0.9	4	2
Obj_{35}	$[60, -60]$	$\pi/1.9$	1.5	4	2
Obj_{36}	$[80, 140]$	$-\pi/2.2$	1.1	6	3

Table 6: The moving objects in Situation 3. The table contains the object data that is available to the trajectory planning.

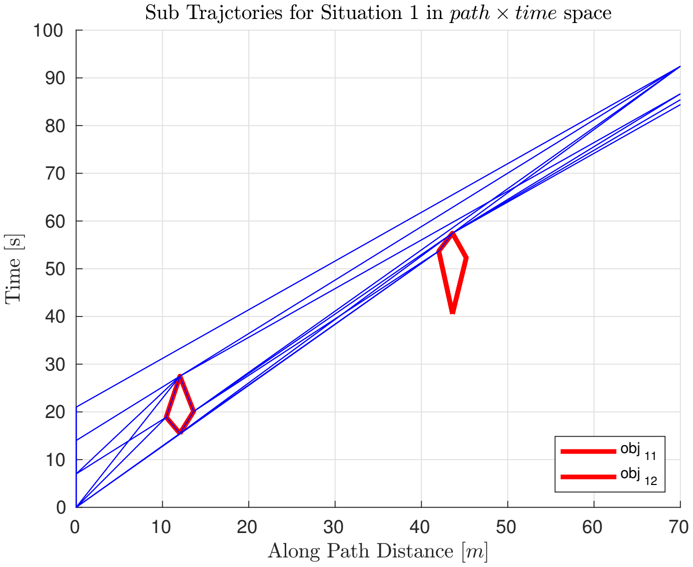

Figure 13: Possible sub-trajectories for the trajectory planner in Situation 1 plotted in blue. The objects from Table 4 represented at forbidden regions in red.

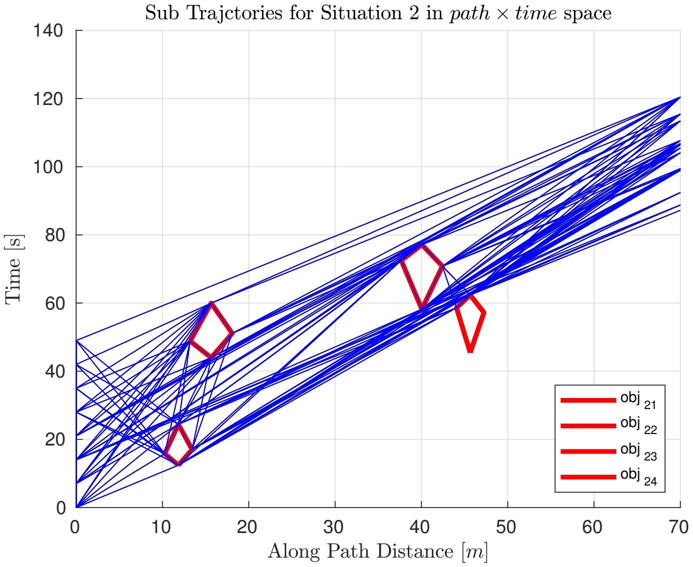

Figure 14: Possible sub-trajectories for the trajectory planner in Situation 2 plotted in blue. The objects from Table 5 represented at forbidden regions in red.

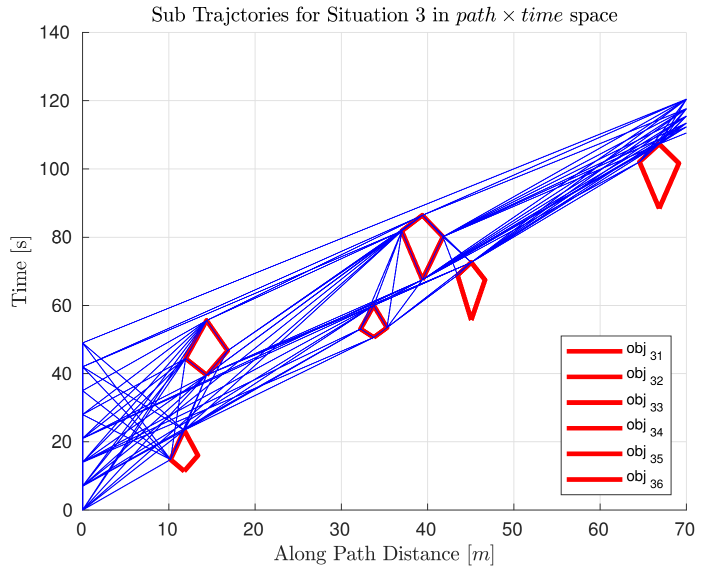

Figure 15: Possible sub-trajectories for the trajectory planner in Situation 3 plotted in blue. The objects from Table 6 represented at forbidden regions in red.

Figure 16-18 shows the optimal trajectory for each time of start for each situation, along with the suggested trajectory for the transit in dashed green line. Table 7, 8 and 9 shows the calculated cost for each trajectory in situation 1, 2 and 3 respectively. The cost is calculated according to (26) with gains $K_{TT} = 1$, $K_{AT} = 1$ and $K_{acc} = 1$.

When the algorithm finds potential trajectories, it finds the shortest trajectory from a start node to the an node in time, and does therefore often suggest trajectories that zigzag between the objects to get to the other side. It is therefore beneficial to add enough start noes for the algorithm to always find a trajectory with a constant velocity profile that transits after all the objects have passed. This becomes clear by Figure 18, where the optimal trajectory awaits until all vessel in the canal have passed before starting the transit. If the last two start nodes had not been includes, one can see from cost in Table 9 that trj_{34} would be the optimal, and hence the ferry would perform a transit with 19.8 seconds longer transit time, that passes in front of two vessels in during the transit, and arrives only 8.1 seconds earlier.

The method of always adding start nodes until an undisturbed transit is available has some limitations, since it assumes that all objects in the canal are observable, which demands sensor systems able to see several hundred meters in each direction, which might not be realistic. Without this assumption fulfilled, the trajectories waiting for vessels to pass, will introduce high uncertainty,

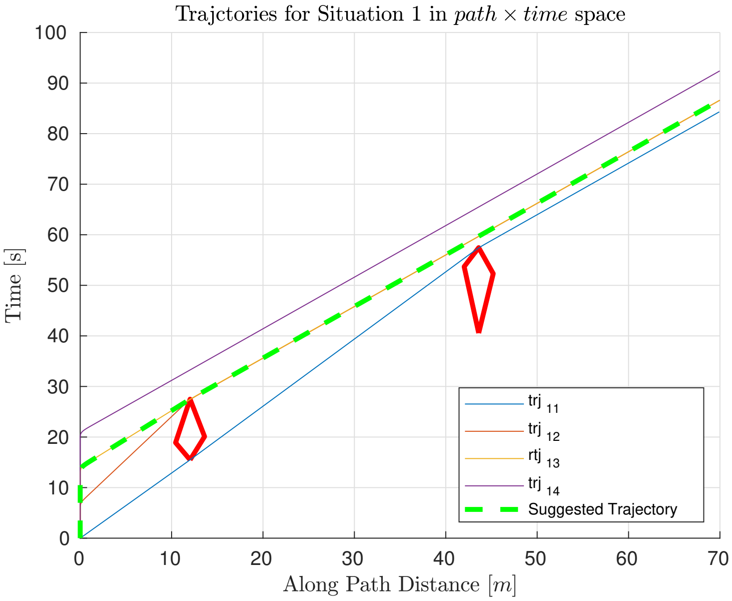

Figure 16: Optimal trajectory for each time of start in blue. Suggested transit trajectory in green. Forbidden regions around the objects in Table 4. Cost for each trajectory is shown in Table 7.

due to lack of information about what the traffic picture will look like in the future. If the optimal trajectory suggest to postpone the transit start, a re-planning of the trajectory should be done close to the transit start, but this might led to a further postponement of transit start if new moving objects are detected. With a high traffic picture this could led to long waiting times before transit, in stead of taking what appears to be a sub-optimal trajectory with a shorter time to transit start.

Name	Transit Cost	Acceleration Cost	Time To Arrival Cost	Total Cost
trj_{11}	12.5	2.869	0	15.37
trj_{12}	8.0	4.497	2.3	14.8
trj_{13}	1.1	1.015	2.3	4.415
trj_{14}	0	0	8.1	8.1

Table 7: Calculated cost for the optimal trajectory in each starting point of the trajectories in Figure 16. Cost is calculated according to (26), with gains $K_{TT} = 1$, $K_{AT} = 1$ and $K_{acc} = 1$.

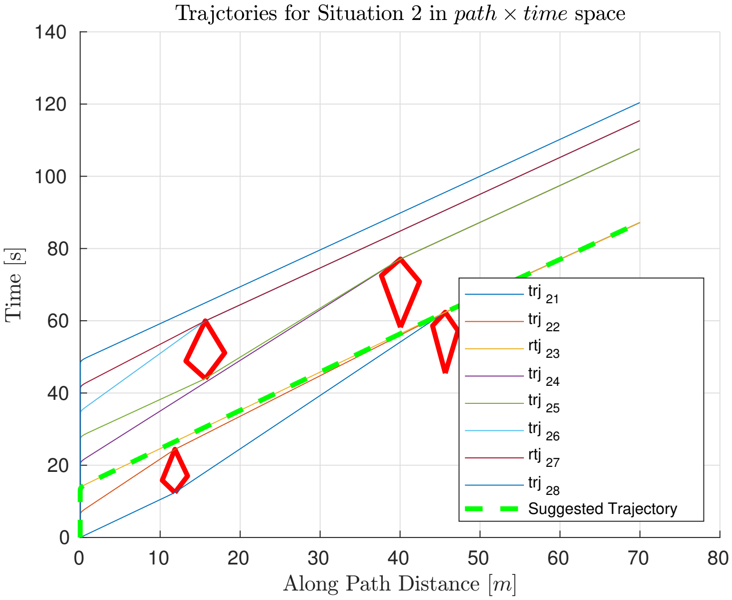

Figure 17: Optimal trajectory for each time of start in blue. Suggested transit trajectory in green. Forbidden regions around the objects in Table 5. Cost for each trajectory is shown in Table 8.

Name	Transit Cost	Acceleration Cost	Time To Arrival Cost	Total Cost
trj_{21}	15.4	6.621	0	22.02
trj_{22}	8.7	3.436	0	12.14
trj_{23}	1.8	0.4144	0	2.214
trj_{24}	15.1	3.046	20.4	38.55
trj_{25}	8.2	5.558	20.4	34.16
trj_{26}	8.8	4.045	28.2	41.05
trj_{27}	1.9	1.257	28.2	31.36
trj_{28}	0	0	33.2	33.2

Table 8: Calculated cost for the optimal trajectory in each starting point of the trajectories in Figure 17.Cost is calculated according to (26), with gains $K_{TT} = 1$, $K_{AT} = 1$ and $K_{acc} = 1$.

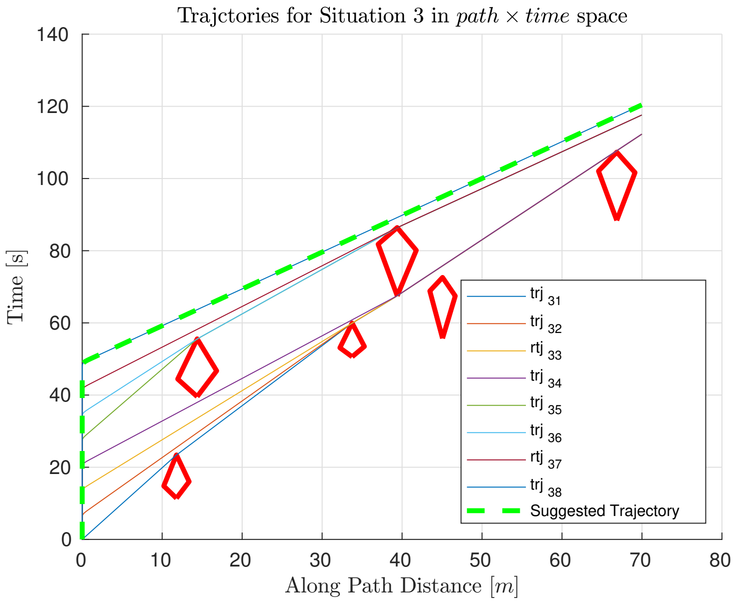

Figure 18: Optimal trajectory for each time of start in blue. Suggested transit trajectory in green. Forbidden regions around the objects in Table 6. Cost for each trajectory is shown in Table 9.

Name	**Transit Cost**	**Acceleration Cost**	**Time To Arrival Cost**	**Total Cost**
trj_{31}	40.5	3.498	0	44.0
trj_{32}	33.7	1.963	0	35.66
trj_{33}	26.8	0.8525	0	27.65
trj_{34}	19.8	1.88	0	21.68
trj_{35}	18.0	5.244	5.3	28.54
trj_{36}	13.1	4.214	5.3	22.61
trj_{37}	4.1	1.063	5.3	10.46
trj_{38}	0	0	8.1	8.1

Table 9: Calculated cost for the optimal trajectory in each starting point of the trajectories in Figure 18.Cost is calculated according to (26), with gains $K_{TT} = 1$, $K_{AT} = 1$ and $K_{acc} = 1$.

6.2 Trajectory Following

The trajectory following algorithms are all tested on the same trajectory, a straight path from one quay to the other, with a velocity profile that ensures no collisions. The position, velocity and acceleration profile in $\{n\}$ is shown in Figure 19. Figure 20 displays the external forces acting on the vessel, given in the $\{n\}$ frame. The tracking error for the three trajectory following methods is shown in Figure 10, while the control output from the control systems are displayed in Figure 22. Figure 23 shows the body fixed linear and angular accelerations for the vessel during the transit, and Figure 24 shows the heading and the yaw rate. Metrics for comparing the trajectory tracking methods in terms of passenger comfort are shown in Table 10. The table gives values for the integral of the absolute value of the accelerations and yaw rate in $\{b\}$ for the whole transit. Table 11 gives the same values for the tracking error in $\{n\}$ for the data presented in Figure 21. The table also gives the integrated sum of all control inputs.

From the data presented in Figure 21 and Table 11 is is clear that the MPC approach to trajectory tracking is far superior to the two other. The MPC has an integrated tracking error of an order of magnitude less, with approximately the same control input. Table 10 gives that the MPC also scores best in terms of passenger comfort. The MPC handles the environmental disturbances well, and manages to keep the heading of the vessel steady, and hence the yaw rate at a minimum. The vessel model in the MPC is the same as the one used to make the simulator, which gives it an unrealistic advantage. This is also the case the feed forward in both the other control strategies, and hence experimental testing of all three methods is desirable.

The reference feed forward with feedback on $\boldsymbol{\eta}$, named **PID** in the tables and figures, and the LOS guidance has similar performance both in trajectory tracking and passenger comfort. The PID is more precise, and scores better on passenger comfort. Since the control law adjusts the heading of the vessel in order to

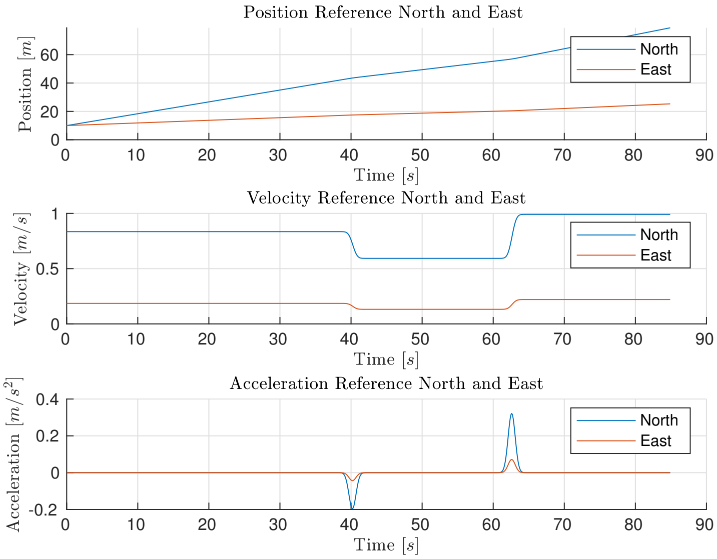

Figure 19: Reference trajectory for the transit. Position, velocity and acceleration in $\{n\}$.

Controller	$\dot{u}$	$\dot{v}$	r	$\dot{r}$
PID	0.7168	0.1125	0.03731	0.00834
LOS guidence	0.9295	0.3266	0.5956	0.1267
MPC	0.6864	0.01428	0.001221	0.004297

Table 10: Metrics for comparing the passenger comfort of each control strategy. Values are integrated absolute value of the state in the header.

minimize the tracking error, it is reasonable that the controller will introduce yawing under the influence of external disturbances. Effect from the external disturbances in Figure 20 on the heading and yaw rate can be seen in Figure 24. This effect could be reduces by in stead controlling the velocity vector of the vessel while keeping the heading at a stable or slowly varying reference. This would also make the LOS utilize the capabilities of the fully actuates system, in stead of acting like an under actuated system, as can be seen from Figure 22, where the Y output from the LOS guidance is zero.

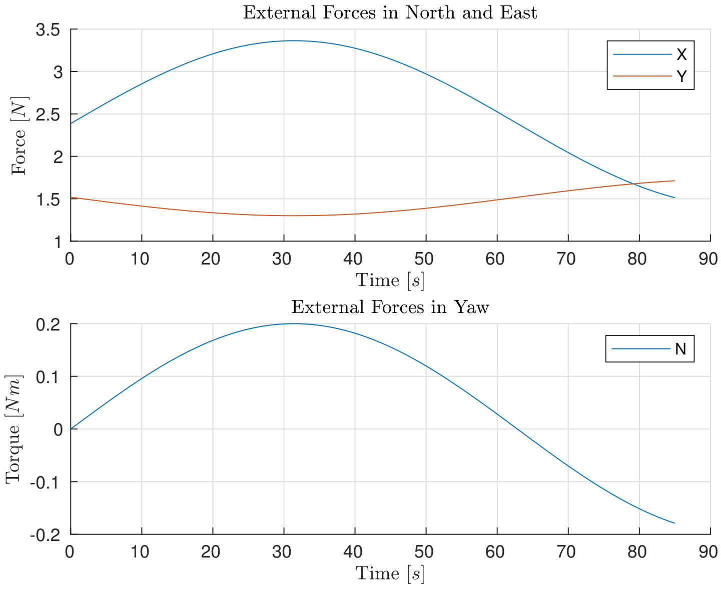

Figure 20: External forces in $\{n\}$, on the vessel during transit .

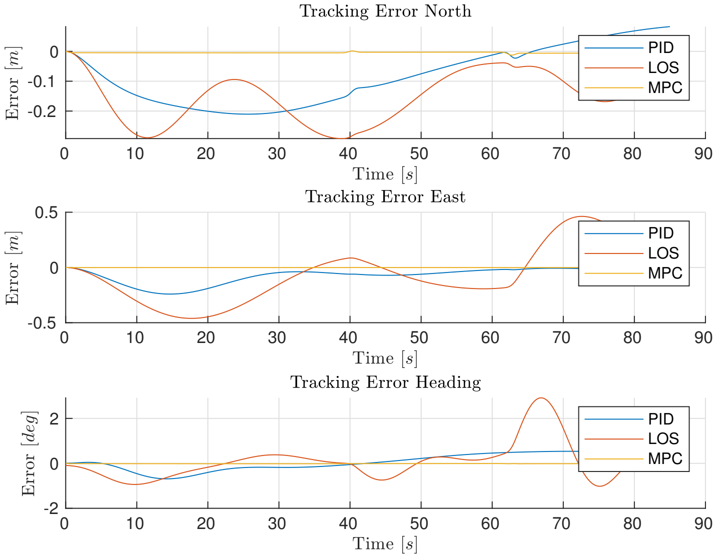

Figure 21: Tracking error in $\{n\}$ for the trajectory following systems.

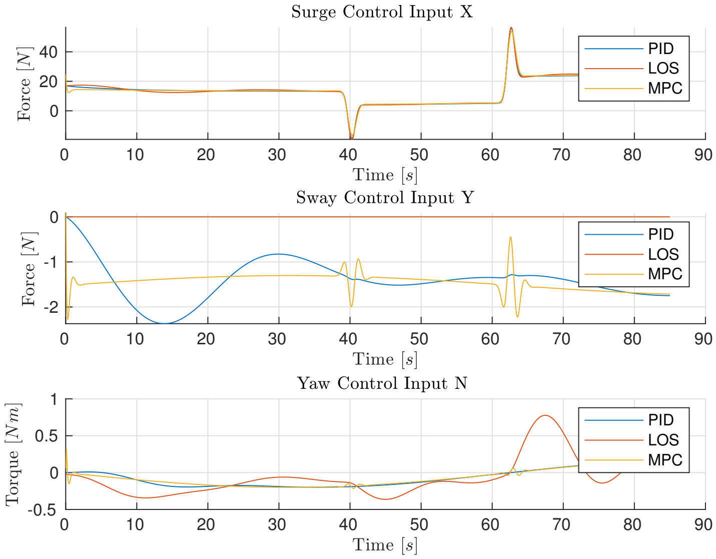

Figure 22: Control input in $\{b\}$, from the three trajectory following systems.

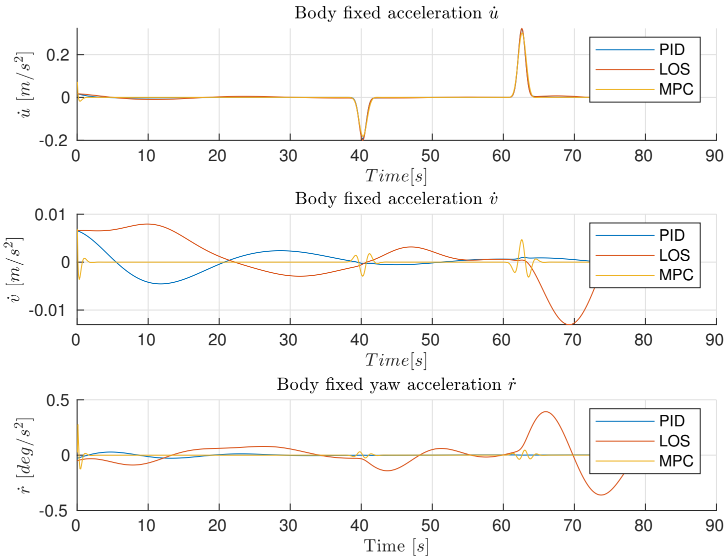

Figure 23: Body fixed acceleration for the ferry in transit by the three different trajectory tracking systems.

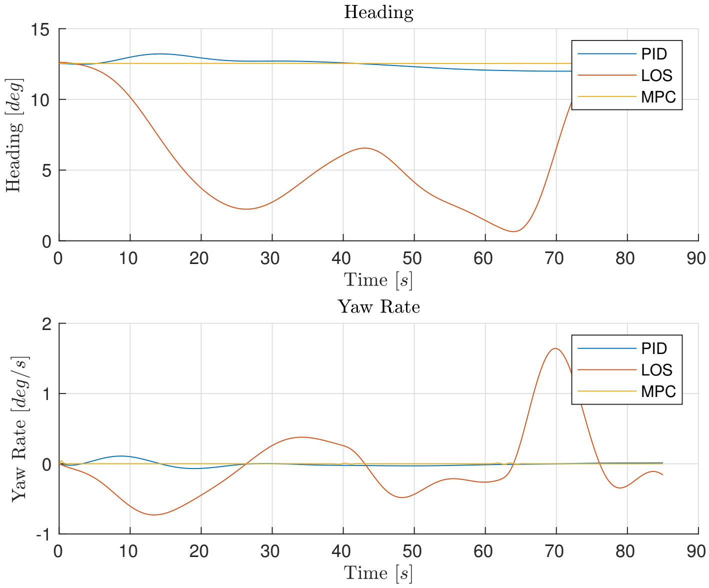

Figure 24: Heading and yaw rate for the ferry in transit by the three different trajectory tracking systems.

Controller	Error North	Errro East	Error Heading	Control Input
Feed Forward PID	8.856	6.098	0.48	1259.0
LOS guidance	12.91	18.12	0.7987	1258.0
MPC	0.3923	0.08381	0.01997	1260.0

Table 11: Metrics for comparing the trajectory tracking capabilities. Values are integrated absolute value of the state in the header.

7 Conclusions and Future Work

Some concluding remarks about the work presented in this project thesis and possible future work.

7.1 Conclusions

In this project, a deliberate trajectory planning system with COLAV has been implemented and tested in simulations, along with three trajectory following control strategies.

The COLAV system performed satisfactory. It is intuitive in the way it plans the trajectory, which gives it predictability, as well as makes it easy to evaluate and improve. The object representation makes it simple to compensate for uncertainty or add a factor of safety.

The COLAV system does not handle uncertainty very well, and is dependent on observable environments. Therefore it is of benefit for for the system to be augmented with a reactive COLAV layer.

The MPC came out as the superior trajectory tracking method, but is based on a model with unrealistically high fidelity. Both the PID and the LOS approach perform acceptable, but would benefit from being augmented with observers and feedforward of external disturbances. The LOS guidance also need a change of control objective in order to increase the passenger comfort.

7.2 Future Work

In continuation of this work, the COLAV systems will be augmented with a reactive layer, giving the possibility to adapt to erroneous assumptions in the traffic picture used in the deliberate trajectory planning. Following, the two systems combined will be tested and validated on the milliAmpere ferry. For the deliberate COLAV, the assumptions about the objects will be further investigated. The object model can be augmented with some stochasticity on heading and velocity in order to increase the safety factor in the planned trajectory. The cost function for the trajectories will also be reviewed and expanded to include a term for the risk involved with the trajectory.

The simulator will be updated to include a more correct vessel model, as soon as the work on system parameter identification is completed. There will be added a model for wind disturbance, in order to facilitate testing of more advanced features of the trajectory following system. A new and improved object detection module will be developed, with some stochasticity on the objects in the canal, so that the reactive COLAV can be tested in more realistic situations.

The trajectory following systems will also be tested and validated on the ferry. A choice of control strategy, and development of a robust trajectory following system will be conducted. It will be augmented with wind and current observers, as well as model parameter estimators for system parameters that are rapidly changing, such as the mass of the ferry as passengers are boarding and unboarding.

References

[1] B.-O. Eriksen and M. Breivik, "MPC-based Mid-level Collision Avoidance for ASVs using Nonlinear Programming," in *IEEE Conference on Control Technology and Applications*, Hawaii, USA, 08 2017.

[2] J. Bjørnø, "Thruster-Assisted Position Mooring of C/S Inocean Cat I Drillship." Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2016.

[3] Wärtsilä. (2018) Wärtsilä achieves notable advances in automated shipping with latest successful tests. [Online]. Available: https://www.wartsila.com/media/news/28-11-2018-wartsila-achieves-notable-advances-in-automated-shipping-\with-latest-successful-tests-2332144

[4] T. Stensvold, "Verdens første helt autonome fergeseilas gjennomført - teknologien er 100 prosent klar," *Teknisk Ukeblad*, Dec 2018. [Online]. Available: https://www.tu.no/artikler/verdens-forste-helt-autonome-fergeseilas-gjennomfort-teknologien-er-100-\prosent-klar/452610

[5] K. Maritime. (2018) Autonomous ship project, key facts about YARA Birkeland. [Online]. Available: https://www.km.kongsberg.com/ks/web/nokbg0240.nsf/AllWeb/4B8113B707A50A4FC125811D00407045?OpenDocument

[6] T. I.Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control.* John Wiley & Sons, 2011.

[7] ROS.org. (2018, November) About ros. [Online]. Available: http://www.ros.org/about-ros/

[8] B.-O. Eriksen, M. Breivik, K. Pettersen, and M. Wiig, "A Modified Dynamic Window Algorithm for Horizontal Collision Avoidance for AUVs," in *IEEE Multi-Conference on Systems and Control (MSC)*, Benos Aires, Argentina, 09 2016.

[9] I. B. Hagen, D. K. M. Kufoalor, E. F. Brekke, and T. A. Johansen, "Mpc-based collision avoidance strategy for existing marine vessel guidance systems," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7618–7623, 2018.

[10] T. Johansen, T. Perez, and A. Cristofaro, "Ship Collision Avoidance and COLREGS Compliance Using Simulation-Based Control Behavior Selection With Predictive Hazard Assessment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 1–16, 05 2016.

[11] B. Siciliano and O. Khatib, *Springer Handbook of Robotics.* Berlin, Heidelberg: Springer-Verlag, 2007.

[12] K. Kant and S. W. Zucker, “Toward Efficient Trajectory Planning: The Path-Velocity Decomposition,” *The International Journal of Robotics Research 1986 5:72*, vol. 5, pp. 72–89, 1986.

[13] “COLREGS - International Regulations for Preventing Collisions at Sea,” *Articles of the Convention on the International Regulations for Preventing Collisions at Sea*, 1972.

[14] S. Kim, P. Silson, A. Tsourdos, and M. Shanmugavel, “Dubins path planning of multiple unmanned airborne vehicles for communication relay,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 225, pp. 12–25, 01 2011.

[15] H. Zheng, R. R. Negenborn, and G. Lodewijks, “Trajectory tracking of autonomous vessels using model predictive control,” *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 8812 – 8818, 2014, 19th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016430041

[16] C.-T. Chen, *Linear Systems Theory and Design.* Oxford University Press, 2013.

[17] MathWorks, “Robotic System Toolbox ,” 2018. [Online]. Available: https://se.mathworks.com/products/robotics.html