Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Emil Hjelseth Thyri

A Path-Velocity Decomposition Approach to Collision Avoidance for Autonomous Passenger Ferries

Master's thesis in Cybernetics and Robotics
Supervisor: Morten Breivik
June 2019

NTNU
Norwegian University of Science and Technology

Emil Hjelseth Thyri

A Path-Velocity Decomposition Approach to Collision Avoidance for Autonomous Passenger Ferries

Master's thesis in Cybernetics and Robotics
Supervisor: Morten Breivik
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Abstract

When developing fully autonomous surface vessels such as passenger ferries, two critical systems need to be in place for it to be autonomous, and not merely automated. Firstly, it needs to have a situational awareness system that is able to capture all the features of the situation that are relevant for the mission of the ferry, and describe the features in a way that is usable. Secondly, it needs to have a mission planning system that uses the understanding of the situation along with the mission goal to make a feasible plan. The mission planner also needs to be able to adjust the plan underway, if changes in the perception of the situation were to occur. Such a mission-planning system for an autonomous surface vessel is typically called a collision avoidance (COLAV) system.

This Master's thesis presents an approach to a complete COLAV system for autonomous passenger ferries. The COLAV system has a three-layered structure with a deliberate, reactive and executive layer. The deliberate layer is a global trajectory planner that uses path-velocity decomposition, with a predefined path, or set of paths, that is collision-free with regard to any static objects. This reduces the task of avoiding moving objects down to a velocity planning problem. The problem is solved by firstly transforming an object representation onto the path-time space as nodes, and subsequently solving the velocity planning as a minimum cost node search problem. The reactive layer tracks the trajectory reference and continuously validates the feasibility of the current global plan. It also monitors changes in the object situation based on estimated time to closest approach and distance of closest approach and invokes a replan in the deliberate layer when necessary. The executive layer acts as a relay between the deliberate and reactive layer and handles mission-request and invocations.

Two versions of the velocity-planning algorithm in the deliberate layer are implemented. The first uses a single predefined path as input to the velocity planning problem, while the second uses a set of predefined parallel paths, where the algorithm is allowed to switch between the paths freely during the transit.

A simulator is developed based on model parameters from a prototype of an autonomous passenger ferry called milliAmpere. The simulator is used to test the COLAV system with up to four moving objects with different behaviours. The two versions of the COLAV system are tested and compared to an implementation of the existing velocity obstacle algorithm.

Through simulations, both the new velocity-planning algorithm prove to be effective. They are able to plan global trajectories with ease, adapt to rapid changes in the environment and perform transits with lower maneuvering efforts than the velocity obstacle approach.

The proposed COLAV system is also validated through sea trials, where both algorithms again performed satisfactory. The single-path approach turns out to be the most convincing algorithm in terms of passenger comfort and safety.

Sammendrag

Under utviklingen av autonome systemer, som for eksempel autonome passasjerferger, er det to kritiske sub-systemer som må være på plass for at systemet ikke bare skal være automatisert. Det første er et situasjonsforståelses-system som kan detektere og beskrive alle aspekter ved omgivelsene som er relevant for oppdraget som skal utføres. Det andre er et system som planlegger utførelsen av oppdraget basert på situasjonen i omgivelsene og oppdragets natur. Oppdragsplanleggeren må også kunne tilpasse planen underveis dersom det skjer endringer i situasjonsforståelses-systemets beskrivelse av omgivelsene. For autonome overflatefartøyer kalles et slikt oppdragsplanleggings-system gjerne et kollisjonsungåelses-system (COLAV-system).

I denne masteroppgaven presenteres et forslag til et komplett COLAV-system for autonome passasjerferger. Systemet har en tredelt struktur med en planleggende del, en organiserende del og en reaktivt del. Den planleggende delen består av en global ruteplanlegger. Den deler oppgaven inn i baneplanlegging og hastighetsplanlegging, og benytter seg av en eller flere forhåndsdefinerte baner som ikke kolliderer med statiske hindringer i omgivelsene. Dette reduserer problemet til å planlegge en hastighetsprofil langs den forhåndsdefinerte banen som sørger for at fergen ikke kolliderer med andre fartøyer i omgivelsene. Problemet løses ved at alle fartøyer representeres som noder i et bane-tid rom, for så å løse det som et minste-kostnad-nodesøk problem. Den reaktive delen av systemet følger ruten som er planlagt, og samtidig driver en kontinuerlig vurdering av planens holdbarhet, basert på oppdatert informasjon om fartøyene i omgivelsene. Om vurderingen viser at planen ikke holder, igangsettes en ny planlegging i den planleggende delen. Den organiserende delen håndterer kommunikasjonen mellom det planleggende og den reaktive delen i tillegg til å håndtere oppdragsforespørseler.

To versjoner av COLAV-systemet er implementert. Den første bruker en forhåndsdefinert bane som input til hastighetsplanleggingsproblemet, den andre bruker et sett med parallelle forhåndsdefinerte baner der algoritmen får lov til å skifte mellom banene underveis i overfarten.

En simulator er utviklet basert på modellparametere fra en prototype av en autonom passasjerferge som heter milliAmpere. Simulatoren brukes til testing av COLAV-systemet med opptil fire bevegelige objekter med forskjellig oppførsel. De to versjonene av COLAV-systemet er testet og sammenlignet med en versjon av Velocity Obstacle algoritmen.

Begge systemene viser god oppførsel i simulatoren. De klarer å planlegge globale ruter uten problemer, og tilpasser seg endringer i omgivelsene underveis. De utfører også overfarten med mindre manøvrering enn Velocity Obstacle algoritmen.

Systemet er også validert gjennom sjøprøver der begge metodene igjen gir tilfredsstillende resultater. Systemet med bare én bane viser seg å være det mest overbevisende systemet når det gjelder passasjerkomfort og sikkerhet.

Preface

This thesis is written as the product of the finalization of my M.Sc degree in Cybernetics and Robotics at the Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU). In the work presented here, I have had the pleasure of diving into the field of autonomous vessels and collision avoidance. It has been an overwhelming, interesting and inspiring task, where the potential for results is very tangible and might contribute to solving some of the problems we face today. This has been a great motivation.

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. I would also like to thank Revolve NTNU for everything I learned and experienced during my year in the organization. Without it, I would not be half as capable as I am today. Lastly, I would like to thank my friends for making these five years in Trondheim quite enjoyable.

The goal of this project was to develop a COLAV system for autonomous passenger ferries operating in an environment characterized by high traffic and confined space. The COLAV system was developed for a prototype with limited functionality concerning situational awareness.

- During the semester, my supervisor has contributed with guidance through a bi-weekly follow-up meeting where the progress of the thesis and other related topics were discussed.
- The development was mostly conducted in Matlab/Simulink, where a simulator was implemented to facilitate the testing. The vessel and thruster model parameters were determined by Anders Pedersen in his project and Master's thesis work during the fall of 2018 and spring of 2019.
- For the comparison with the velocity obstacle method, a complete code-base was borrowed from fellow student Anette Uttisrud, from the work she conducted in her project thesis. Only minor changes to the code were necessary to make it work and interface with the dummy object detection and reference filter.
- It is the availability of the milliAmpere platform that has allowed me to perform the sea trials. The platform is owned by NTNU and has been developed since 2016 by several people at the university.
- During the experiments, Brage Sæther contributed as a vessel operator when the autonomous pipeline was not running, and as a sparring partner when the autonomous pipeline performed unpredictably. Brage is also the person that has developed the navigation and motion control system that the COLAV system runs on top of.

Emil Hjelseth Thyri
Trondheim, June 17, 2019

Table of Contents

List of Tables

List of Figures

Abbreviations

COG	Course Over Ground
COS	Change Of Speed
DCO	Distance to Closest Object
DCPA	Distance to Closest Point of Approach
DOF	Degree Of Freedom
DP	Dynamic Positioning
DW	Dynamic Window
ECEF	Earth Centered Earth Fixed
ENC	Electronical Nautical Chart
GNC	Guidance Navigation and Control
GNSS	Global Navigation Satellite System
Lidar	Light Imaging Detection and Ranging
LOS	Line Of Sight
MP-VP	Multiple Path Velocity Planner
MPN	Mild Penalty Nodes
NED	North East Down
OBC	On Board Computer
OS	Operating System
PPP	Path Planning Problem

ROC	Region Of Collision
ROS	Robot Operating System
ROT	Rate Of Turn
RRT	Rapidly Exploring Random Tree
RTK	Real-Time Kinematic
SOG	Speed Over Ground
SP-VP	Single Path Velocity Planner
SPA	Sense-Plan-Act
SPN	Strict Penalty Nodes
TCPA	Time to Closest Point of Approach
UAV	Unmanned Aerial Vehicle
VO	Velocity Obstacle
VPP	Velocity Planning Problem

Chapter 1

Introduction

In this chapter, the motivation for the work done in this Master's thesis 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

Over the last two decades, the evolution of autonomous systems has shown revolutionary tendencies. The development is going faster and faster, much caused by more and more resources being pumped into the field as the market potential is charted. In the automotive industry, this is especially noticeable with several of the big car manufacturers delivering cars with some level of autonomy, and are promising more to come in the next years (FutureAgenda, 2019). Goods transportation is also seeing potential in the autonomous technology, both on land, air and sea, with the autonomous containerships, haul trucks, platooning and UAVs delivering packages in cities (Kongsberg Maritime, 2018); (RioTinto, 2018); (Scania, 2019); (Amazon, 2016). The heavy urbanization and growth of cities over the last decades have posed problems related to green transportation systems, to cope with the increased demand for transportation while not polluting the city environment. Autonomous electric buses is an option under development and testing in several cities over the world. The waterway is an underutilized option in many cities, where water-shuttles can take load off the roads. This has previously been an expensive option with high operator cost, but now, as the technology is emerging, unmanned electric autonomous ferries are becoming an option in cities, as well as coastal infrastructure.

Electric autonomous passenger ferries can offer an adaptable, environmentally friendly and cost-effective option to the city infrastructure. The ferries can be unmanned, and thereby reduce the operating cost. An electric propulsion system does not pollute the city air like traditional diesel ferries. The ferries demand little "on land" infrastructure, and can therefore be very adaptable depending on the season or changing need in the transport demand.

Figure 1.1: Top left: Yara Birkeland, an autonomous container-vessel under development by Kongsberg Maritime, courtesy of Kongsberg Maritime. Top right: Autonomous hauling-trucks, courtesy of Christian Spronge Photography. Bottom left: Four autonomous trucks platooning, courtesy of Scania. Bottom right: Autonomous package delivering drones, courtesy of Amazon.

milliAmpere is a prototype of an autonomous passenger ferry. The ferry is three by five meters and can be seen in **Fig. 1.2**. The ferry serves as a development platform for all systems that are needed on an autonomous passenger ferry. On the milliAmpere project, there is currently development on everything from power management in the propulsion systems, new sensor systems, to situational awareness and collision avoidance systems. The project is funded by NTNU, where bachelor and master students, PhD candidates and professors work alongside each other in the developing and testing process. The design process of a full-scale ferry based on the milliAmpere prototype can be seen in **Fig. 1.3**. The vision for the project is to produce a fully functional on-demand electric autonomous passenger ferry for urban environments. Urban environments call for complex traffic picture. If the ferry is to perform fully autonomous transits, it needs to plan its motion-pattern in a way that avoids collision with any static and dynamic objects in the environments, hence it needs a COLAV system. The COLAV system, along with the situational awareness system, are two of the systems that separate a non-autonomous ferry from an autonomous one. What seems trivial for trained captains is no trivial task for a computer. Observing a situation, predicting how the environments change, and acting based on it quickly becomes complex with an environment that is nowhere near fully observable. However, if autonomous passenger ferries are to be a reality, this is a problem that needs to be solved.

1.2 Problem Description

The problem that is addressed in this Master's thesis is the trajectory planning for an autonomous vessel operating in known environments, where the motions are confined by a predefined area or path. The trajectory is to be planned in a way that ensures a collision-

Figure 1.2: The prototype milliAmpere during sea trials. The ferry functions as a platform for development and testing of sensor systems, situational awareness algorithms, COLAV systems and control systems needed for fully autonomous transits. Courtesy of Nicholas Dalhaug.

free transit from the current position to the destination. The system needs to react to changes in the situation underway, and always deliver a collision-free trajectory. It is assumed that the predefined area or path is free of any static object, and the trajectory planning therefore only needs to consider the moving objects in the environment. The trajectory planner further needs to be encapsulated in a complete COLAV system that interfaces with the situational awareness systems at one end, and the low-level motion control systems in the other end. The following objectives are proposed for this Master's thesis

- Develop a complete COLAV system for an autonomous passenger ferry, with all functionality that is needed for autonomous transits.
- Implement interfaces to the situational awareness modules.
- Implement interfaces to the trajectory following system.
- Develop a simulator that can emulate moving objects in the environments.
- Verify the performance of the COLAV system through simulations.
- Prepare the system for, and perform full-scale testing of the COLAV system.

1.3 Contributions

The contributions of this Master's thesis are

Figure 1.3: Illustration of the full-scale autonomous ferry that is under development at NTNU. Courtesy of Petter Mustvedt, Institute of design, NTNU.

- A complete COLAV system is developed and tested. The system plans a trajectory from start to finish that is collision-free, and continuously validates and updates the trajectory during the transit. The system is designed with respect to the environment the ferry is to operate in. In addition, it is simple, predictable and does not introduce a high computational cost to the OBC of the ferry.

- A simulator that interfaces with the existing control and sensor systems on the ferry is developed. The simulator facilitates 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 sensor systems and functionality.

- A dummy object detection module is developed. The module emulates moving objects in the environments and has functionality for different object behaviour, such as aggressive, passive and rules-compliant. The module facilitates testing of the COLAV system in both ”normal” and critical situations in both the simulator and in full-scale testing, without imposing great risk to the platform and the operators.

- A reference generator with a dual reference filter is developed to serve as an interface between the COLAV system and the trajectory following systems. The filters are

designed with respect to passenger comfort, as well as response in critical situations. The filters ensure that the reference from the COLAV system is feasible, and sustain the passenger comfort whenever possible.

- The COLAV system is benchmarked against a velocity obstacle-based COLAV system through simulations.
- The complete pipeline with dummy object detection, COLAV system, reference filters and the control systems on the milliAmpere ferry is tested in full-scale sea trials.

1.4 Previous Work

Today there are at least a couple of actors in the process of development and testing of autonomous ferries. Wärtsilä was testing their automated transit and docking system in November 2018 and performed a three-legged transit including docking and departure between each leg. The transit was performed fully automated, but without any COLAV functionality, so if a situation were to emerge, a stand-by operator would take over Wärtsilä (2018). Only a short week later Rolls-Royce performed a fully autonomous transit with COLAV capabilities in Finland. The transit encountered a set of (artificial) scenarios, where the situational awareness systems registered the situations, and the COLAV system acted on the observations. Only weeks before the finalization of this report, Wärtsilä, as the first in history, put a self-driving ferry into commercial operation (NRK, 2019).

On collision avoidance algorithms for marine vessels, there has been done quite some research over the last couple of decades. The collision avoidance methods can be roughly separated into two categories, reactive and deliberate. Among the reactive/local methods there is for example the Dynamic Window (DW) algorithm, and the Velocity Obstacle (VO) algorithm (Eriksen et al., 2016);(Kuwata et al., 2014). Both methods use assumptions about moving objects to make a prediction of collision-free headings and velocities and pick a velocity-heading set according to a cost function. The reactive algorithms typically depend on real-time sensor data, and not a global understanding, in combinations with an objective to determine the control-action. This makes them reactive to the current situation, and they often come at a low computational cost. The short-term span, on the other hand, can cause them to favour local minima, and if care is not taken, get stuck or diverge from the global objective.

The deliberate methods use a priori information about the environments, i.e. a continuously updated map on the environments, including both static and dynamic objects, to find a global (optimal) solution. In the modified A* algorithm presented in (Campbell de Oliveira et al., 2013), the dynamic objects are represented by making unavailable regions in locations where the COLREGs will be violated. Other deliberate methods are the Rapidly Exploring Random Tree (RRT) algorithm introduced in (M. Lavalle and Kuffner, 2000), and MPC-based methods like the Branching Course MPC in (Eriksen and Breivik, 2017) and the simulation-based MPC in (Hagen et al., 2018). In the latter of the methods, the MPC objective is to compute modifications to the desired course and speed in order to produce a COLREGs compliant trajectory. The method assumes constant obstacle behaviour. The deliberate algorithms often come with high computational complexity, and therefore

put demands on the computer system that houses it, and restricts replanning frequency and response time.

In later years, we have seen the emergence of hybrid architectures, where a reactive and a deliberate method run in parallel, at different rates. This gives the benefit of long-term planning, where the chances of reaching the global goal are high, along with the reactive quality of a high-frequency short term planner (Loe, 2008);(Eriksen and Breivik, 2018).

What makes the case for autonomous passenger ferries special is that the area of operation is known in advance, and will typically be confined by static obstructions in the environments. This restricts the avoidance maneuvers of the ferry, with respect to course-changing. In comparison, the methods presented in (Eriksen et al., 2016), (Kuwata et al., 2014) and (Hagen et al., 2018), favours changes in course, and deviates from the initial path with significant cross-track error. The high traffic in urban waterways is also a feature that is unique for this case, and multi-target collision avoidance is a necessity. The combination of confined space and high traffic scenarios is, to the best of the author's knowledge, not much considered in the literature so far.

1.5 Outline

The rest of this report will be about solving the problems described in **Section 1.2**. Chapter 2 gives an introduction to some fundamental theory as well as some key terms and concepts. In Chapter 3, the suggested COLAV system is presented in detail followed by simulation results and a description of the simulator in Chapter 4. Chapter 5 describes the experimental platform along with the experimental data. Lastly, in Chapter 6, the report is concluded along with suggestions for future work.

Chapter 2

Theoretical Background

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

2.1 Vessel Modelling

2.1.1 Kinematics

In navigation and motion control of marine crafts, a set of reference frames 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 centre of the earth, and it rotates around the earth 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.

- **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 points towards the true north, the y-axis points to the east, and the z-axis 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 $\{b\}$ and 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-axis 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 the use of $\{n\}$ in combination with $\{b\}$ to describe the system. Since we

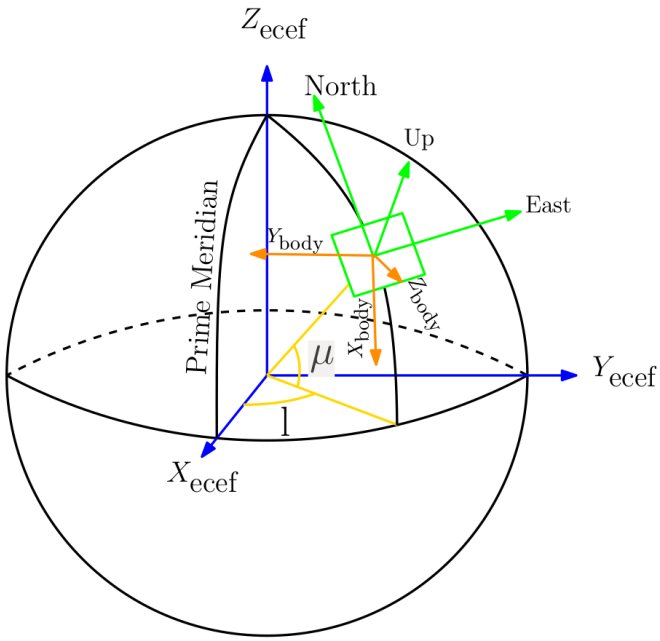


Figure 2.1: Reference Frames, ECEF frame in blue, NED frame in green and BODY frame in orange.

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-axis

$$\boldsymbol{J}_{\Theta}(\eta) = R_{z,\psi} = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

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

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu}, \tag{2.2}$$

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 (Fossen, 2011). The manoeuvring model is based on the rigid body kinetics

$$\boldsymbol{M}_{RB}\dot{\nu} + \boldsymbol{C}_{RB}(\nu)\nu = \boldsymbol{\tau}_{RB}, \tag{2.3}$$

$$\boldsymbol{\tau}_{RB} = \boldsymbol{\tau}_{hyd} + \tau_{hs} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{waves} + \boldsymbol{\tau}, \tag{2.4}$$

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

$$\boldsymbol{\tau}_{hyd} = -\boldsymbol{M}_A\dot{\boldsymbol{\nu}}_r - \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r - \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r, \tag{2.5}$$

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 (2.3)-(2.5) we get the maneuvering equation

$$\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{M}_A\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r\boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu} = \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{wave}. \tag{2.6}$$

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 $\dot{\boldsymbol{v}}_c = \mathbf{0}$ Fossen (2011). By doing this the system can be written on the form

$$\boldsymbol{M}\dot{\boldsymbol{\nu}_r} + \boldsymbol{C}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r = \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{wave}, \tag{2.7}$$

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A, \tag{2.8}$$

$$\boldsymbol{C}(\boldsymbol{\nu}_r) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu}_r). \tag{2.9}$$

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 (Fossen, 2011). 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

$$\boldsymbol{e}(t) := \begin{bmatrix} N(t) - N_d(t) \\ E(t) - E_d(t) \\ \psi(t) - \psi_d(t) \end{bmatrix}. \tag{2.10}$$

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 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 a software developed for building robot applications. ROS is not an operating system, but rather a collection of software frameworks for robot software development (ROS.org, 2018). 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 the intuitive and standardized message-based interface that ROS uses. It also facilitates the use of a variety of programming languages in combination. ROS works well for both low-level hardware drives, as well as advanced algorithms.

Chapter 3

Collision Avoidance System

This chapter presents an approach to a complete COLAV system. Key functions of the system are described in the following sections. The method is based on the concept of path velocity decomposition first described in Kant and Zucker (1986) and later revised in Fraichard and Laugier (1993). Path velocity decomposition separates the trajectory planning problem into a Path Planning Problem (PPP) and a Velocity Planning Problem (VPP). The path planning only considers static obstacles and the velocity planning finds a velocity-profile for the planned path that gives a collision-free trajectory with respect to any moving objects. In this thesis, only the VPP is considered, since the static environments are familiar and therefore a path can easily be predetermined.

In more detail, the velocity planning problem is solved by transforming the moving objects onto the two-dimensional space spanned by the path and time, hereby referred to as $path \times time$ space. This is done by assuming constant velocity and heading for all moving objects, and performing a linear transformation on the object representation to transform it onto the $path \times time$ space. In the two-dimensional space, the velocity planning can be solved by a node-search algorithm. The feasibility of this approach for the case considered in this thesis was looked into by the author in (Thyri, 2018). In the thesis, up to six objects were transformed onto a $100m$ long path, and it was investigated whether a node-search algorithm was a viable way of finding a velocity profile. The method proved efficient and will, therefore, found the basis for the COLAV system developed in this Master's thesis.

This thesis presents two systems based on the same method. One method finds a trajectory along a single path, while the other searches for a trajectory along multiple-paths. The methods are named single-path velocity planner (SP-VP), and multiple-path velocity planner (MP-VP).

3.1 System Architecture

In this section, the choice of system architecture is reasoned, and an introduction to the different system modules is given. In the development of autonomous systems, architecture is one of the major factors in the system capacity and behaviour. The system architecture

is the composition of the autonomous system, hardware and software, between the sensors and the actuators (Kortenkamp and Simmons, 2007). In the very first autonomous systems, this was done in hardware, but as the micro-controllers and computers emerged with ever-increasing computational capacity, this problem evolved to be more of a software problem. Up to the mid-1980s, the Sense-Plan-Act (SPA) method was the dominant view. The SPA system is a hierarchical system composed of a sensing system interpreting the sensor-data into a world model, a planning system that combines the mission goal with the world mode to generate a plan, and an acting system that generates and execute actions according to the plan. **Fig. 3.1a** illustrates the SPA-architecture. The hierarchy of the system makes the flow of information unidirectional, which makes it easy to implement on a single computational unit by running the three steps in sequence. As the missions grew in complexity, so did the mission planning and the amount of sensor data. This, along with available hardware, puts restrictions on the planning frequency, which eventually became a major drawback of the SPA structure. It eventually reached its maximum potential. From the mid-1980s quite a lot happened in this field, and by 1990 the three-layer Architecture was born (Gat, 1997). The three-layer architecture is a hybrid structure, where the sensing, planning and acting can run independently if the other, and the flow of information can be bi-directional between all modules. **Fig. 3.1b** give an illustration of such a system (Lussier et al., 2019). The hybrid structure allows for running subsystems at different frequencies, which enables advanced planning algorithms to perform long-term mission planning, while the sensing and acting can perform high frequency tasks to comply with a short-term objective.

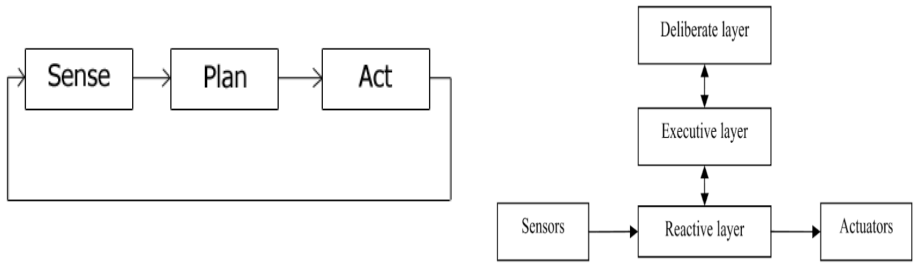


Figure 3.1: **(a)** The hierarchical Sense-Plan-Act architecture. Courtesy of Wikipedia. **(b)** Hybrid structure. Courtesy of Alexander Inzartsev.

In the three-layer architecture, the deliberate layer performs long-term planning based on mission objectives, environmental information as well as vessel models. In the case considered here, the plan will be in the form of a set of waypoints or a trajectory that stretches a period ahead from the current time and state. The executive layer functions as a supervisor, receiving the plans from the deliberate layer and sequencing it before feeding it on to the reactive layer. It also receives information based on real-time sensor data from the reactive layer and sends requests and invocations on to the deliberate layer. The reactive layer realizes to the best of ability the plans from the executive layer. In this case, it will

by large be a trajectory following task, which includes computing the actuator setpoints based on a reference as well as real-time sensor data. In addition to this, the reactive layer monitors sensor data and continuously validates the feasibility of the current plan.

Due to the generic and adaptable qualities of the three-layer architecture, it is the structure of choice for the COLAV system presented in this thesis. The system overview will be presented in further detail in the following sections.

3.1.1 Module Overview

Fig. 3.2 gives an illustration of a COLAV system with the three-layer architecture. The figure contains the three layers as well as a set of surrounding support modules that interfaces with the COLAV system.

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 change 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. In the work presents in this thesis it is assumed that the Object Detection module delivers extended object tracking data on all objects in proximity to the ferry (Brekke et al., 2018).

The **Executive Layer** is an organizing layer. It receives missions from the mission-planner in terms of transit requests, as well as a path or a set of paths. The supervisor thereafter invokes the deliberate planner and receives the plan as a set of waypoints (with timestamps) in return. The plan is sequences into reference signals by the reference generator and streamed to the trajectory following control system. The long-term plan from the deliberate layer is also relayed onto the reactive layer where it is continuously evaluated. If it fails, the supervisor in the executive layer invokes the deliberate planner again.

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 requests a transit from the supervisor in the executive layer. The mission-planner also provides the path from the current location to the desired location. Note that the mission planner only gives a path, or a set of paths, and not a trajectory, in the way that it provides a set of waypoints in $\{n\}$ independent of time. The transit request sets a time for earliest possible departure but puts no restrictions on the time of arrival, nor a timestamp for any underway checkpoints. The suggested path or set of paths is collision-free only with respect to static obstacles.

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 waypoints in a way that avoids collision during the transit. This is done by generating a position and velocity profile in $\{n\}$ that is fed to the executive layer. A suggested method for the deliberate layer will be presented later in this section.

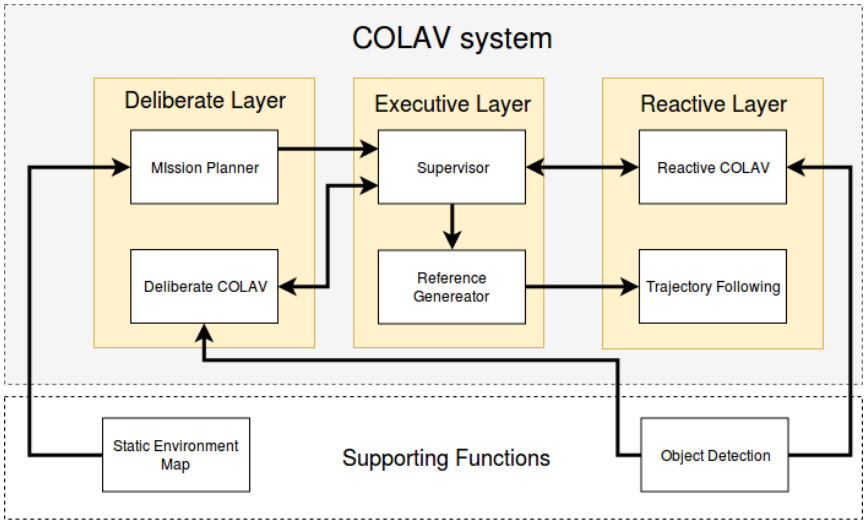


Figure 3.2: Illustration of a COLAV system with the three-layer architecture and supporting functions.

The **Reactive Layer** receives the plan in terms of instructs and references from the executive layer. In this system, the reactive layer will have two main tasks. One is to track the trajectory reference from the executive layer, by means of a trajectory tracking control system. The second main task is to monitor sensor data from the object detection module and evaluate if the current plan still is feasible, and if a replan would be of benefit due to sudden changes in the information on the dynamic objects. In this thesis, the second task will be the one in focus, while the task of trajectory tracking has been addressed previously by the author in (Thyri, 2018).

3.2 Single-Path COLAV System

In this section, the SP-VP is presented. The section starts by describing how the objects are represented in both $\{n\}$ and the $path \times time$ space. Following, the design of the nodes and vertices in $path \times time$ is explained along with how potential trajectories are found and evaluated. In the last subsections, the feasibility and smoothing of the trajectory are addressed.

3.2.1 Object Representation

The moving objects in the environments are detected by the object detection module. In this thesis, it is assumed that the module is capable of extended object tracking (Brekke et al., 2018). In the paper, the shape of the object is fitted to an ellipsoidal contour with the length and width as the semi-minor and semi-major axis respectively. Along with size, the position, heading and velocity are estimated where the heading is parallel with the semi-

major axis. In order to ease the computational complexity, a Region Of Collision (ROC) determined by four corners in $\{n\}$ is placed around the vessel in a way that encapsulates the whole contour of the ellipse and is further used as the new object representation. The four corners are calculated by

$$co_{front} = [N_{obj} + k_{f\_roc} l_{obj} \cos(\psi_{obj}), E_{obj} + k_{f\_roc} l_{obj} \sin(\psi_{obj})], \tag{3.1}$$

$$co_{starboard} = [N_{obj} + k_{s\_roc} w_{obj} \cos(\psi_{obj} + \frac{\pi}{2}), E_{obj} + k_{s\_roc} w_{obj} \sin(\psi_{obj} + \frac{\pi}{2})], \tag{3.2}$$

$$co_{rear} = [N_{obj} + k_{r\_roc} l_{obj} \cos(\psi_{obj} + \pi), E_{obj} + k_{r\_roc} l_{obj} \sin(\psi_{obj} + \pi)], \tag{3.3}$$

$$co_{port} = [N_{obj} + k_{s\_roc} w_{obj} \cos(\psi_{obj} + \frac{3\pi}{2}), E_{obj} + k_{s\_roc} w_{obj} \sin(\psi_{obj} + \frac{3\pi}{2})], \tag{3.4}$$

where l_{obj} and w_{obj} is the length and width of the object, $[N_{obj}, E_{obj}]$ is the North-East position of the object, and k_{f_roc}, k_{r_roc} and k_{s_roc} is the gain for the forward length, rear length and width respectively. The gains can be adjusted to compensate for uncertainty and desired safety factor.

For the node search algorithm to be able to find a trajectory that does not collide with the objects, a set of nodes, denoted $\boldsymbol{O}$ needs to be added in proximity to the object. The nodes will allow the node search algorithm to use one of the nodes if the trajectory is to pass close by a moving object and thereby ensure a certain distance to the object, based on how the nodes are constructed.

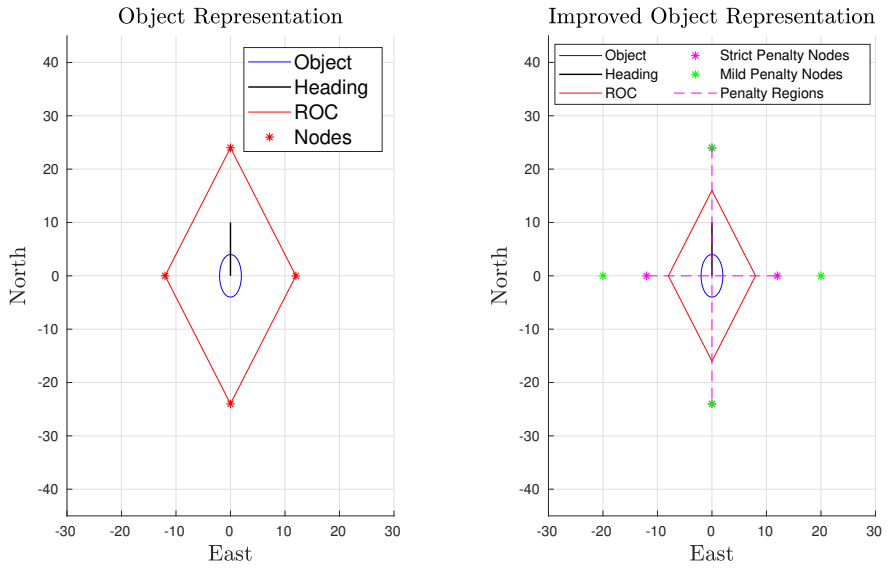
One way of positioning the nodes is to place the nodes at the corners of the ROC. **Fig. 3.3a** illustrates what this approach looks like for an object at the position $[N_{obj}, E_{obj}] = [0, 0]$ and $\psi_{obj} = 0$. The figure displays the object ellipse, heading as well as the ROC and the nodes. This approach was initially tested, and the results from it can be seen in **Section 4.3.1**. The method proved to be prone to disturbance and non-constant object behaviour due to zero margin between the nodes and the ROC. This lead to the improved object representation in **Fig. 3.3b**.

The improved object representation has a few extra features. For one, there are two sets of nodes, namely the *Strict Penalty Nodes*, SPN, and the *Mild Penalty Nodes*, MPN, as pink and green asterisk in **Fig. 3.3b** respectively. In addition, the *Penalty Region* is introduced as a line between two and two SPNs. The function of all the elements in the object representation will be clear in the following.

3.2.2 Transformation to Path-Time Space

For the VPP to be solved as a node search problem, the object representations are transformed onto the $path \times time$ space. In the transformation, it is assumed that all moving objects keep constant COG and SOG. With the assumptions along with a parameterization of the predefined path, the object representations can be transformed onto the $path \times time$ space.

Since both the nodes and the four corners of the ROC can be described by the coordinates of a point in $\{n\}$, all the object information can be transformed by the same method. Therefore the method is described for a general path and point in $\{n\}$.



(a) **(b)**

Figure 3.3: Old and new object representation. **(a)** Initial object representation with the nodes on the corners of the forbidden regions. **(b)** Improved object representation, with the nodes placed outside the forbidden regions.

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

$$P := \frac{N - N_{start}}{a} = \frac{E - E_{start}}{b}, \tag{3.5}$$

$N \in [N_{start}, N_{end}]$, $E \in [E_{start}, E_{end}]$ where the length of the path is

$$l = \sqrt{(N_{end} - N_{start})^2 + (E_{end} - E_{start})^2}. \tag{3.6}$$

By choosing a and b as

$$a = \frac{(N_{end} - N_{start})}{l}, \tag{3.7}$$

$$b = \frac{(E_{end} - E_{start})}{l}, \tag{3.8}$$

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.

The position $[N(t), E(t)]$ of a point in a plane as a function of time can be described by

$$N(t) = N_{obj} + U_{obj}\cos(\psi_{obj})(t - t_{obj}), \tag{3.9}$$

$$E(t) = E_{obj} + U_{obj}\sin(\psi_{obj})(t - t_{obj}), \tag{3.10}$$

where $[N_{obj}, E_{obj}, \psi_{obj}]^T$ and U_{obj} is the pose and velocity at at time t_{obj} respectively. By substituting N and E from (3.5) for $N(t)$ and $E(t)$ in (3.9)-(3.10) we get

$$Pa - N_{start} = N_{obj} + U_{obj}\cos(\psi_{obj})(t - t_{obj}), \tag{3.11}$$

$$Pb - E_{start} = E_{obj} + U_{obj}\sin(\psi_{obj})(t - t_{obj}). \tag{3.12}$$

The equations can be solved to get a point $[P, t]$ in the $path \times time$ space for every point in $\{n\}$. **Fig. 3.4a** displays such a projection of four moving objects.

3.2.3 Add Start and End Nodes

For the node search to find a path from the current position of the vessel to the desired destination, nodes need to be added at those locations. A set of start nodes $\boldsymbol{S}$ is added to the total set of nodes. One start node is added at the current position of the ferry, namely $[P, t] = [0, t_0]$ where t_0 is the current time. If the ferry is docked, awaiting to start a transit, a number of start nodes with $P = 0$ and time greater than t_0 can also be added to $\boldsymbol{S}$, this will give the trajectory planner the possibility to wait at the current position for some time before initiating the transit. This can prove beneficial for example if the traffic along the transit path is especially active. The effects of this are previously studied by the author and presented in (Thyri, 2018).

With the start nodes added, a set of end nodes, $\boldsymbol{E}$, is added. In order to facilitate the trajectory planner choosing trajectories at transit velocity, end nodes can be added in such a way that there always will be an end node in transit velocity from each of the nodes in $\boldsymbol{S}$ and $\boldsymbol{O}$. This is done by setting the P coordinate of each end node to the path-length, l, and calculate the t value according to

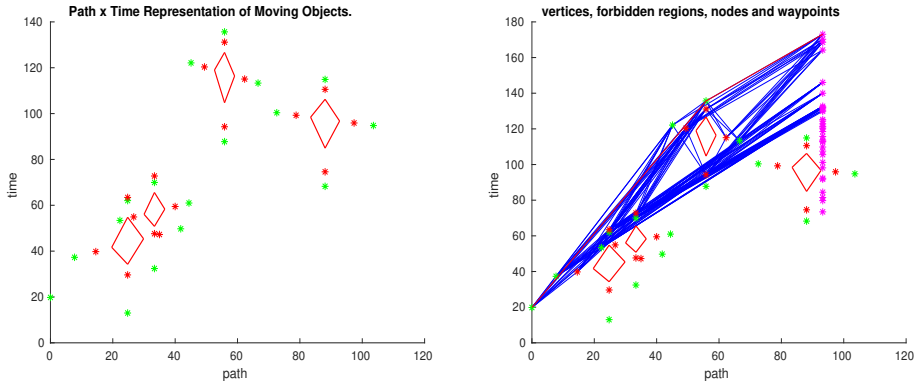
$$t_{end\_node} = t_{node} + \frac{l - P_{node}}{U_{des}}, \tag{3.13}$$

for all nodes in $\boldsymbol{O}$ and $\boldsymbol{S}$, where U_{des} is the desired transit velocity and $[P_{node}, t_{node}]$ are the coordinates of the node.

3.2.4 Vertex Generation

With the set of all nodes $\boldsymbol{N}$ consisting of the subsets $\boldsymbol{S}$, $\boldsymbol{O}$ and $\boldsymbol{E}$, the set of feasible vertices combining the nodes can be added. A vertex is a directional straight line that connects two nodes, namely *Node_one* and *Node_two*, where it starts in *Node_one*. The set of vertices make a tree that connects the start node, at the current position of the vessel, to the end nodes, at the destination of the transit. **Fig. 3.4b** shows such a net for a situation with four objects. Since a vertex represent a physical aspect, four criteria has to be fulfilled for it to be considered feasible.

- *Node_two* must be later or equal in time than *Node_one*
- *Node_one* must be later or equal in time than the current node
- The velocity required to travel along the vertex is not higher than U_{max}



(a) **(b)**

Figure 3.4: $path \times time$ space for the deliberate planner in a situation with four moving objects in the environment. **(a)** Transformation of the object representation with nodes from **Fig. 3.3b**. **(b)** Object-nodes, ROC, start node, end nodes and vertices.

- The vertex does not pass through any Regions of Collision

where U_{max} is the maximum velocity for the ferry.

The first three criteria are trivial to check, the fourth criteria can be checked by the following approach. Let $\boldsymbol{V}$ be the set of all vertices that fulfill the first three criteria, and let $\boldsymbol{F}_{ROC}$ be the set of edges that make up the Region of Collision for all moving objects. Each vertex can be formulated as a line by

$$V_i = \frac{p - p_s}{p_e - p_s} = \frac{t - t_s}{t_e - t_s}, \tag{3.14}$$

with $V_i \in (0, 1)$ where $[p_s, t_s]$ are the coordinates of *Node_one*, $[p_e, t_e]$ are the coordinates of *Node_two* and i is the index of the vertex in $\boldsymbol{V}$. The ROC can, in the same manner be described by

$$R_j = \frac{p - p_1}{p_2 - p_1} = \frac{t - t_1}{t_2 - t_1}, \tag{3.15}$$

with $L_i \in (0, 1)$ where j is the index of the ROC in $\boldsymbol{F}_{ROC}$, and $[p_1, t_1]$ and $[p_2, t_2]$ are the start and end coordinates of the line in $path \times time$ space. Equations (3.14) and (3.15) can be written as

$$p = V_i(p_e - p_s) + p_s, \tag{3.16}$$

$$t = V_i(t_e - t_s) + t_s, \tag{3.17}$$

$$p = R_j(p_2 - p_1) + p_1, \tag{3.18}$$

$$t = R_j(t_2 - t_1) + t_1. \tag{3.19}$$

By combining (3.16) and (3.18) to eliminate p

$$V_i(p_e - p_s) + p_s = R_j(p_2 - p_1) + p_1, \tag{3.20}$$

and (3.17) and (3.19) to eliminate t

$$V_i(t_e - t_s) + t_s = R_j(t_2 - t_1) + t_1, \tag{3.21}$$

the two new new equations can be solved to find V_i and R_j by either solving

$$R_j = \frac{t_1 - \frac{p_1 - p_s}{p_e - p_s}(t_e - t_s)}{\frac{p_2 - p_1}{p_e - p_s}(t_e - t_s) - (t_2 - t_1)}, \tag{3.22}$$

and inserting R_j into (3.21) to find V_i, if $p_e - p_s \neq 0$, or solving

$$R_j = \frac{p_1 - \frac{t_1 - t_s}{t_e - t_s}(p_e - p_s)}{\frac{t_2 - t_1}{t_e - t_s}(p_e - p_s) - (p_2 - p_1)}, \tag{3.23}$$

and inserting R_j into (3.20) to find V_i if $p_e - p_s = 0$ and $t_e - t_s \neq 0$. Both $p_e - p_s = 0$ and $t_e - t_s = 0$ can not be true at the same time, since it gives a vertex without size in $path \times time$ space.

For every V_i in $\boldsymbol{V}$: solve the equations for every R_j in $\boldsymbol{F}_{ROC}$. If any of the combinations give $V_i \in (0, 1)$ and $R_j \in (0, 1)$, the vertex intersects a ROC, and is removed from $\boldsymbol{V}$.

3.2.5 Vertex Cost Function

The vertex cost represents the cost related to using that vertex as a sub-trajectory in the trajectory. The cost should be calculated in a way that represents the objectives for the trajectory. The objectives for the trajectory is that it should be safe, and ideally not pass through any penalty regions. Also, since this approach does not give a way to penalize difference in velocity between two vertices in the trajectory, a cost representing how far the vertex velocity is from the desired velocity is included. In addition, the cost of the node is included, along with a cost on the duration of the vertex. The cost is calculated according to equations (3.24)-(3.28)

$$cost_{vel} = |U_{vertex} - U_{des}|K_{vel}, \tag{3.24}$$

$$cost_{risk} = risk_{bool}K_{risk}, \tag{3.25}$$

$$cost_{node} = cost_{node\_two}, \tag{3.26}$$

$$cost_{time} = \min(\frac{K_{time}}{|U_{vertex}|}, time\_cost_{max}), \tag{3.27}$$

$$cost = cost_{vel} + cost_{risk} + cost_{node} + cost_{time}, \tag{3.28}$$

where K_{vel}, K_{risk} and K_{time} are tunable gains, $risk_{bool}$ is a boolean value that is true if the vertex passes through a high risk region as described in **Section 3.2.1**.

3.2.6 Search for Best Vertices

The node search algorithm used to solve the minimum cost path algorithm is the *Dijkstra's Minimum Cost Path Algorithm*. Initially, the Matlab function *shortestpath()* was used, but since it does not support code generation, a modified version of the algorithm in Kirk (2015) is used. The inputs for the algorithm are the nodes, vertices, weight, start nodes and end nodes. The algorithm outputs the nodes making up the minimum cost path from all the start nodes to all the end nodes.

3.2.7 Generate Trajectory from Nodes

The nodes in the $path \times time$ space has to be translated to a format that the trajectory following control system can make use of. Therefore the nodes are transformed to waypoints in the local $\{n\}$ frame along with the timestamp for that waypoint. The north and east coordinates of the waypoint are found by inserting the P coordinate of the node into (3.5), the timestamp of the waypoints is simply the t coordinate of the node. The set of waypoints can further be used to generate a pose, velocity and acceleration reference as a function of time for the control system to follow. Due to the format of the node search problem and the vertices, the resulting trajectory from the waypoints will have a step in velocity between two vertices, and therefore, infinite acceleration. Since infinite acceleration is not yet possible for autonomous passenger ferries, some filtration needs to be applied to the reference signals for it to be feasible.

Reference Filter

In the process of filtration, special care has to be taken in order not to violate the collision-free qualities of the initial trajectory, since any filtration of the original reference will give an error between the hard reference, and the feasible reference. In addition to the physical capabilities of the ferry, the comfort of the passengers becomes a factor in the design of the filter for the trajectory references. Studies have been done on determining what is considered comfortable acceleration and jerk in the longitudinal motion (Hoberock, 1977). The article gives a review and a conclusion on 11 studies on longitudinal acceleration for in-ground mass transport systems, mainly with passengers standing unsupported or supported by holding in overhead straps or wall-mounted stanchions. The article concludes that a longitudinal acceleration of $1.1 - 1.5 m/s^2$ and a jerk of $3 m/s^3$ is about the limit for what is considered sufficient passenger comfort. Since the physical experience of standing on a passenger ferry, operating in calm waters, is comparable to standing on a subway train, the suggested values from the article will be used as design parameters in this thesis.

Since a limited jerk and hence a continuous acceleration profile is desirable, a third-order reference filter is needed to filter the steps in the velocity reference. An approach to this is given in Fossen (2011), where a first-order low-pass filter is cascaded with a mass-damper-spring system.

$$\dddot{\boldsymbol{\eta}}_d + (2\boldsymbol{\Delta} + \boldsymbol{I})\boldsymbol{\Omega}\ddot{\boldsymbol{\eta}}_d + (2\boldsymbol{\Delta} + \boldsymbol{I})\boldsymbol{\Omega}^2\dot{\boldsymbol{\eta}}_d + \boldsymbol{\Omega}^3\boldsymbol{\eta}_d = \boldsymbol{\Omega}^3\boldsymbol{r}^n, \tag{3.29}$$

where $\boldsymbol{r}^n$ is the pose reference and $\boldsymbol{\eta}_d$, $\dot{\boldsymbol{\eta}}_d$, $\ddot{\boldsymbol{\eta}}_d$ and $\dddot{\boldsymbol{\eta}}_d$ are the desired pose, velocity,

acceleration and jerk in $\{n\}$. The reference model can be shown to satisfy

$$\lim_{x\to\infty} \boldsymbol{\eta}_d(t) = \boldsymbol{r}^n, \tag{3.30}$$

for a constant $\boldsymbol{r}^n$. This is not the case in our case, where the reference pose is changing when throughout the transit. Since the vertices have a constant velocity, the reference will have a piecewise constant rate, and an acceleration of zero. This means that by augmenting the model with either an integrating effect or a velocity reference in addition to the pose reference, the desired tracking of a dynamic pose reference can be achieved

$$\dddot{\boldsymbol{\eta}}_d + (2\boldsymbol{\Delta} + \boldsymbol{I})\boldsymbol{\Omega}\ddot{\boldsymbol{\eta}}_d + (2\boldsymbol{\Delta} + \boldsymbol{I})\boldsymbol{\Omega}^2\dot{\boldsymbol{\eta}}_d + \boldsymbol{\Omega}^3\boldsymbol{\eta}_d = \boldsymbol{\Omega}^{\mathbf{3}}\boldsymbol{r}^n + (2\boldsymbol{\Delta} + \boldsymbol{I})\boldsymbol{\Omega}^2\dot{\boldsymbol{r}}^n. \tag{3.31}$$

To ensure that the jerk and acceleration do not violate the limits that are comfortable for a standing passenger, saturation can be applied in the reference filter. The physical capabilities of the ferry also need to be taken into account in the calculation of the reference.

The velocity limitations of the ferry can be found by performing some step input tests on the vessel. The implementation of the reference filter will be placed in the executive layer of the COLAV system and needs to have a discrete form. This can be achieved by the following approach, where a fixed timestep Δ_t between each reference calculation is used. The timestep Δ_t is dependent on the frequency the reference filter is running at on the OBC and will be $0.1s$ in the simulations and testing performed in this thesis.

Firstly the jerk reference is calculated from the filter in (3.31) to be

$$\dddot{\boldsymbol{\eta}}_d = \boldsymbol{\Omega}^{\mathbf{3}}(\boldsymbol{r}^n - \boldsymbol{\eta}) + (2\boldsymbol{\Delta} + \boldsymbol{I})\boldsymbol{\Omega}^2(\dot{\boldsymbol{r}}^n - \dot{\boldsymbol{\eta}}) - (2\boldsymbol{\Delta} + \boldsymbol{I})\boldsymbol{\Omega}\ddot{\boldsymbol{\eta}}, \tag{3.32}$$

where $\boldsymbol{\eta}$ is the current pose of the ferry. Following the jerk can be saturated by the saturating function

$$sat(x) = \begin{cases} sgn(x)x_{max} & \text{if } \|x\| \geq x_{max} \\ x & \text{else} \end{cases} \tag{3.33}$$

with $x_{max} = 3m/s^3$, to comply with the passenger preference. Subsequently the acceleration, velocity and pose reference cal be calculated to be

$$\ddot{\boldsymbol{\eta}}_d = \ddot{\boldsymbol{\eta}} + \dddot{\boldsymbol{\eta}}_d\Delta_t, \tag{3.34}$$

$$\dot{\boldsymbol{\eta}}_d = \dot{\boldsymbol{\eta}} + \ddot{\boldsymbol{\eta}}_d\Delta_t, \tag{3.35}$$

$$\boldsymbol{\eta}_d = \boldsymbol{\eta} + \dot{\boldsymbol{\eta}}_d\Delta_t, \tag{3.36}$$

with a saturation between each step, where the acceleration is saturated to $1.1m/s^2$ to comply with passenger comfort. The velocity can also be saturated to the desired limit below the maximum limit of the vessel. A downside of a reference filter designed for passenger comfort is that it reduces the response of the ferry. In most cases, this is unproblematic but could be problematic if a critical situation was to appear, i.e. an object moving so unpredictably that it will lead to a collision if the ferry does not respond fast enough. In such situations, the passenger comfort is no longer of highest priority, and the ferry should use full effect in order to avoid colliding. Therefore it is beneficial to have a second reference filter that only takes into account the capabilities of the ferry. For this, a

second-order reference filter, allowing infinite jerk, and hence steps in acceleration can be used.

$$\ddot{\boldsymbol{\eta}}_d + \boldsymbol{\Delta\Omega}\dot{\boldsymbol{\eta}}_d + \boldsymbol{\Omega}^2\boldsymbol{\eta}_d = \boldsymbol{\Omega^2 r}^n + \boldsymbol{\Delta\Omega}\dot{\boldsymbol{r}}^n. \tag{3.37}$$

Where the velocity and position reference is calculated according to (3.35) and (3.36) respectively. Also in this case a saturation can be applied to the acceleration and velocity reference to comply with the vessels limitations, and ensure that the filter reference does not "run away" from the vessel, but the saturation limits should be as close to the physical limitations as possible to ensure maximum response. Since drag is acting on the hull of the vessel every time is has a relative velocity to the water surrounding it, the maximum acceleration will be dependent on the vessel state $\boldsymbol{\nu}$ as well as any potential currents in the water surrounding the vessel. In this thesis, the effects of the currents will be neglected in order to limit the scope, despite it being absolutely relevant in many of the environments the vessel is to operate in, such as canals and narrow sounds. The maximum acceleration of the vessel can be found by inserting the maximum thruster force $\boldsymbol{\tau}_{max}$ into (2.7), and solving the equation for $\dot{\boldsymbol{\nu}}$ to get

$$\dot{\boldsymbol{\nu}}_{max} = \boldsymbol{M}^{-1}(\boldsymbol{\tau}_{max} - \boldsymbol{C}(\boldsymbol{\nu_r})\boldsymbol{\nu}_r - \boldsymbol{D}(\boldsymbol{\nu_r})\boldsymbol{\nu}_r). \tag{3.38}$$

By using a logical function that can apply automatic switching between the two reference filters according to the situation, passenger comfort and safety does not have to be a compromise. With a feasible trajectory reference calculated, it can be fed on to the trajectory-following control system in order to perform a collision-free transit from the current location to the destination. The reference is passed forward by a ROS-interface between the reference filter in the executive layer and the motion-control system of the vessel at a rate of $10Hz$.

3.3 Multiple-Path Augmentation

During the evaluation of the SP-VP described in **Section 3.2**, it became evident that the strict limitation of a single path between the starting point and the destination of the transit might be more of a limitation than an aid to a simple and predictable COLAV system. The problems are mainly related to moving objects that have a small velocity component orthogonal to the path, as well as unexpected object behaviour that calls for evasive manoeuvres. Therefore an augmentation of the original system was made, in order to compare the single path method to a method that is not locked to one path in $\{n\}$ (Fraichard and Laugier, 1993). Since the Multiple Path approach is based on the same principles as the original method, reuse and modification of the original code became the method of choice in the development. The scope of this augmentation was greatly underestimated, and in retrospect, the author wishes a method with more new code and less reuse was chosen. Nevertheless, the multiple-path COLAV system was developed, and, after some time, tested and compared to the Single Path Approach. In the following subsections, the additions that had to be made to the SP-VP system in order to get the MP-VP will be described.

3.3.1 Multiple Paths

The main difference between the methods is the increased set of paths. Note that, also for this method, the paths are predefined, and the assumption that all paths are free of collision with any static objects stands. The method can be used on an arbitrary number of paths larger than zero. A set of five parallel paths with equal separation have been used in this thesis, an illustration of the paths can be seen in **Fig. 3.5** as blue lines. The set of paths are defined as a set of start and endpoints in $\{n\}$, $P_{pn\_start} = [N_{pn\_start}, E_{pn\_start}]$, $P_{pn\_end} = [N_{pn\_end}, E_{pn\_end}]$, where n is the path number. The paths are numbered from left to right in the direction of travel, therefore the upper path in the figure is path no 1, the lower path is path no 5, and the centerpath is path no 3. The predefined paths are separated with a distance of l_{sep}. From the predefined paths, a set of possible branching paths from the current location of the vessel to the destination along the predefined paths is calculated. In the calculations, it is not assumed that the vessel is on any of the paths. Therefore, the calculations start by finding the shortest way onto one of the predefined paths. This is done by first finding the cross path error to all the paths

$$\epsilon_{n\_cp} = l_{eta} \sin(\theta_{path} - \theta_{l\_eta}), \tag{3.39}$$

where θ_{path} is the course of the path and

$$, \theta_{l\_eta} = atan2((E - E_{n\_start}), (N - N_{n\_start})) \tag{3.40}$$

is the course of

$$l_{eta} = \sqrt{(N - N_{n\_start})^2 + (E - E_{n\_start})^2}, \tag{3.41}$$

where N and E is the north and east position of the vessel, respectively. The current path is then selected to be the path with the shortest absolute cross-track error. From the current path, a set of branching subpaths are made to the other paths in the set of predefined paths. The branching subpaths are defined by the parallel path separation l_{sep} as well as the branching angle α_{branch}, that is the change in course the vessel has to make to branch out from the current path. The branching subpath starts from the point $P_{sn\_start} = [N_{sn\_start}, E_{sn\_start}]$, where s is the index of the path it is branching to, with

$$N_{s\_start} = N + \frac{\epsilon_{n\_cp}}{sin(\alpha_{branch})} cos(\theta_{path} - sgn(\epsilon_{n\_cp})\alpha_{branch}), \tag{3.42}$$

$$N_{s\_start} = N + \frac{\epsilon_{n\_cp}}{sin(\alpha_{branch})} sin(\theta_{path} - sgn(\epsilon_{n\_cp})\alpha_{branch}), \tag{3.43}$$

and ends in the point $P_{sn\_end} = [N_{s\_end}, E_{s\_end}]$

$$N_{s\_end} = N_{s\_start} + \frac{l_{sep}|k|}{sin(\alpha_{branch})} cos(\theta_{path} + sgn(k)\alpha_{branch}), \tag{3.44}$$

$$E_{s\_end} = E_{s\_start} + \frac{l_{sep}|k|}{sin(\alpha_{branch})} sin(\theta_{path} + sgn(k)\alpha_{branch}), \tag{3.45}$$

with $k = n_{end} - n_{current}$ where $n_{current}$ is the number of the current path, and n_{end} is the number of the predefined path the branching subpath is branching to. This ensures

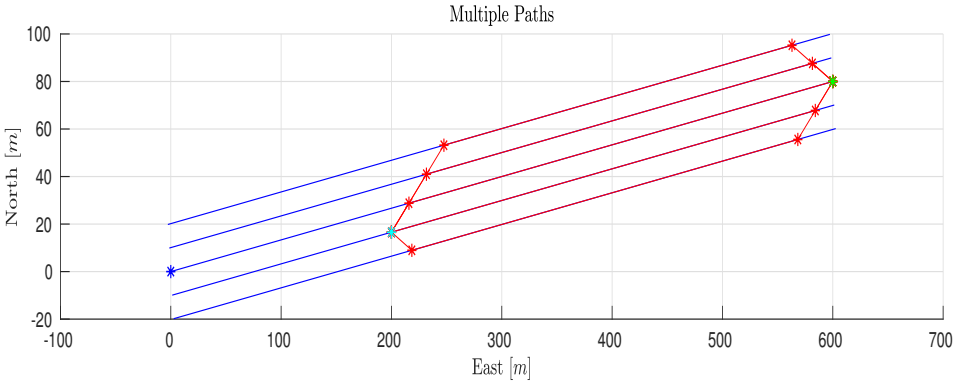


Figure 3.5: Predefined paths in blue, and set of possible branching paths in red, with branching, merging and parallel subpaths. The current location of the vessel as green asterisk. Start and end point for transit in blue and green asterisk respectively.

that the vessel will travel at the angle α_{branch} onto the current path if it has a cross track error greater than zero, and branch out from the current path at an angle α_{branch}. Hence α_{branch} can be used to design the behaviour of the ferry to make large enough changes in course that it is clear to other objects, as well as keeping courses that keep the ferry moving towards the destination.

Once the branch out subpaths are calculated, the branch back subpaths needs to be calculated in order to ensure that the set of paths all end in the desired destination, marked as a green asterisk in **Fig. 3.5**. The branch-back subpaths all end in the same point, $P_{t\_end} = [N_{t\_end}, E_{t\_end}]$, on the centerpath from the start points to the desired destination. The branch back subpaths also use the angle α_{branch}. The starting points $P_{tn\_start} = [N_{tn\_start}, E_{tn\_start}]$ becomes

$$N_{tn\_start} = N_{t\_end} + \frac{l_{sep}|j|}{sin(\alpha_{branch})} cos(\theta_{path} - sgn(j)\alpha_{branch} + \pi), \tag{3.46}$$

$$E_{tn\_start} = E_{t\_end} + \frac{l_{sep}||}{sin(\alpha_{branch})} sin(\theta_{path} - sgn(j)\alpha_{branch} + \pi), \tag{3.47}$$

with $j = n - n_{center}$ where n_{center} is the index of the centerpath of the predefined paths, and n is the path where the branch back subpath is starting from.

With the branch-out and branch-back subpaths defined, the parallel subpath, the part of the branching path that is parallel to the centrepath, is defined by the end point of the branch-out subpath P_{sn_end} and the start point of the branch-back subpath P_{tn_start}.

The set of branching paths consisting of branch-out, parallel and branch-back subpath for a transit starting in $[0, 0]$ and ending in $[80, 600]$ can be seen in **Fig. 3.5**. In the figure, the vessel is on predefined path number 4. The position of the vessel is marked by a green asterisk. The red lines give the possible branching paths from the current position to the destination of the transit.

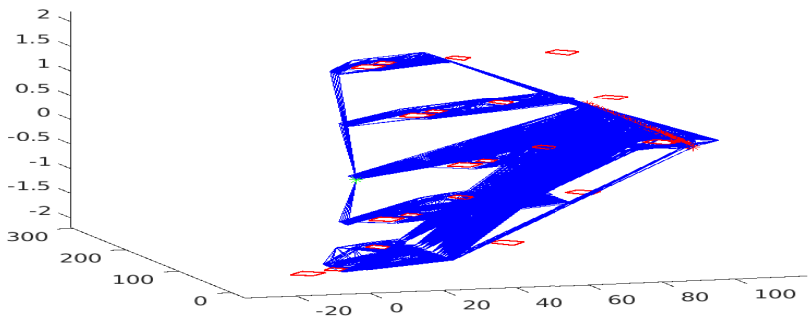


Figure 3.6: Illustration of the vertices and ROC in $path \times time$ space for five parallel paths and four moving objects.

3.3.2 Transformation to Path-Time Space

As the paths are created, the objects can be transformed onto the $path \times time$ space. The method for this is the same as for a single path, but since we now have multiple paths this will give a set of $path \times time$ spaces. Since the minimum cost node search problem needs to have vertices combining the paths, a third dimension is added to the node search problem, where the new dimension represents the predefined path index as a layer in the $path \times time$ space, and the merging and branching subpaths make the path for the vertices between the layers. **Fig. 3.6** displays such a representation in $path \times time$ space for a transit with four moving objects. In the figure, the five parallel subpaths make up the path dimension in each flat layer, and the merging and branching subpaths make up the surfaces containing the vertices combining the layers.

3.3.3 Start Nodes and Vertices

The time coordinate for the start nodes in each layer depends on the desired transit velocity, the separation between the paths l_{sep} and the cross-track error to the current path. If the ferry is a distance greater than zero away from the position of a start node, $node_i$, it will take the ferry a time t_i greater than zero to reach the node. This time has to be added to the current time t_0 to get start nodes with feasible time coordinates. For the current layer the distance to travel is dependent on the cross-track error and the angle of attack, and becomes

$$t_c = \frac{|\epsilon_{n\_cp}|}{sin(\alpha_{branch})v_{des}}, \tag{3.48}$$

where ϵ_{n_cp} is the cross track error of the current path. The added time for the starting nodes in the layers that are not the current layer becomes

$$t_i = t_c + \frac{l_{sep} * |c - i|}{sin(\alpha_{branch})v_{des}}, \tag{3.49}$$

where c is the index of the current layer. The coordinates of the start node in each layer therefore become

$$node_{i\_start} = [0, t_i], \tag{3.50}$$

where i is the index of the layer. The path coordinate is zero since the start nodes are placed at the beginning of each parallel subpath.

Additional start nodes can be added in order to facilitate branching at different velocities than v_{des}. This can be done by simply adding a Δ_t to the time coordinate of the start nodes. In **Fig. 3.6**, each layer except for the current layer have 5 start nodes with a separation $\Delta_t = 4[s]$.

Branch Vertices

The branch vertices are the vertices from the start nodes in the current layer to the start nodes in the rest of the layers. It is the branch vertices that make the connection between the layers. The vertices are created in much the same manner as in the single path method. Of the four criteria listed in **Section 3.2.4**, the first three are true by design for the branch vertices, the fourth one must be tried. This is done by firstly transposing the object representations in $\{n\}$ onto the $path \times time$ space for the branch-out subpath, and subsequently checking if the vertices intersect any of the forbidden regions. The cost for the vertices are calculated from the same cost function as in (3.28), but with an additional term, $cost_{branch} \geq 0$ that adds a cost related to the branching maneuver.

Parallel Path Vertices

For the parallel subpath, the method is the same as for the single path method. The object nodes are found by transforming the object representation onto the path. End nodes are added at the end of the path, with time-coordinates based on transit-velocity from all the start and object nodes and the vertices are made as described in **Section 3.2.4**.

Merge Vertices

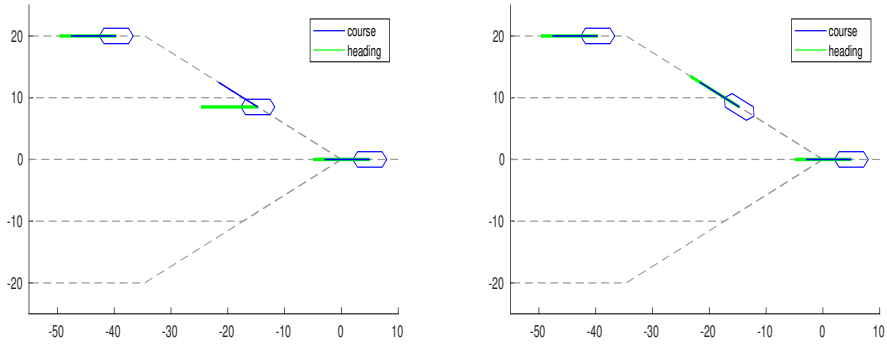
The merge vertices combine the end nodes in each layer to the end nodes in the centre layer. The procedure is identical to the branching vertices. The objects are transformed onto the branch-back subpath, and the vertex from each end node in the branched layer to each end node in the centre layer is tried on the four criteria in **Section 3.2.4**. If it passes the criteria, a cost is calculated according to (3.28). There is no additional cost related to the merging maneuver.

3.3.4 Find Minimum Cost Path

Once all the vertices and weights are calculated, the lowest cost path from the current position to all the end nodes in the centre layer can be found by the same minimum cost path algorithm as for the single path. **Fig. 3.6** displays the complete set of vertices for the problem. The rest of the algorithm is the same as for the single path. The path is calculated from the minimum cost nodes, and the reference filter is applied.

3.3.5 Heading and Course

A consideration to make when creating the trajectory from the nodes for the multiple-path approach is what to do about the vessel heading in the branching subpaths. In theory, there is an infinite set of options, but arguably only two reasonable ones. One is to keep the vessel heading and course reference aligned, and thereby perform a yaw movement of $\pm\alpha_{branch}$, at the start and end of each branching subpath. The second option is to keep the heading constant and aligned with the parallel paths, during the whole crossing, and hence perform a crab motion in the branching and merging subpaths. **Fig. 3.7** gives an illustration of the two methods for determining the vessel heading.



(a) **(b)**

Figure 3.7: Illustration of the two ways of handling the heading when branching. **(a)** Keep a constant heading, and perform a crab motion in the branching maneuvers. **(b)** Align heading and velocity, and perform a yaw rotation at the start and end of each branching maneuver.

The method of aligned course and heading reference is similar to what one would expect from a ”traditional” vessel. This is because most vessels are under-actuated in the sway motion, and therefore not capable of crab motions. This behaviour is also what other actors in the environments expect to observe when a vessel changes its course, and therefore, it increases predictability and ”readability”. The COLREGs also dictate some guidelines for this in Rule 8, *Actions to avoid collision* COLREGs (1995). The rules say that ”Any alternation of course and/or speed to avoid collision shall, if the circumstances of the case admit, be large enough to be readily apparent to another vessel observing visually or by radar”, and ”If there is sufficient sea room, alteration of course alone may be the most effective action”.

The method of keeping a constant heading independent of course limits the yaw motions during the transit, which might increase passenger comfort. In addition, on the shorter transits, the excessive yawing might mislead both passengers and other actors with respect to the intention of the ferry in terms of destination, and therefore induce uncertainty among the passengers.

The method should, therefore, be picked according to the operating environments for the ferry. If the ferry is crossing an open sound or a small fjord, an alignment of course

and heading might be the best choice of actions, since this will increase the readability for other vessels especially if the crossing is so long that it is not apparent where the ferry is heading. In short crossings, on the other hand, the method of constant heading might be preferable, since the constant heading will signal the long-term intention of the vessel while the course changes signal the short-term actions. In any situation, the branch out angle α_{branch} needs to be large enough to be readily apparent.

3.4 Reactive Object Monitoring

The reactive part of the COLAV system is a submodule that monitors the moving objects in the environment and continuously evaluates the feasibility of the current plan and situation. If the reactive object monitoring detects an infeasible plan or a drastic change in the understanding of the situation, a replan in the deliberate layer is triggered. The method for this is roughly twofold, one part checks if the current trajectory still is collision-free based on the current object data. The other part keeps track of the object information and which of the observable objects that need to be taken into account in the planning.

3.4.1 Trajectory Feasibility

In the test for trajectory feasibility, it is the trajectory waypoints that is used. The method tests each leg of the trajectory independently by firstly calculating a path of the leg according to (3.5), where P_{start} is the waypoint at the start of the leg, and P_{end} is the waypoint at the end of the leg. Subsequently, the regions of collision for all objects are translated onto the path and give a $path \times time$ space for that subtrajectory. After that, a vertex can be made between the two waypoints, where the time-coordinate is given by the time of the waypoint. The path-coordinate is zero for the first node, and l for the second node where l is given by (3.6). With the vertex of the subtrajectory and the regions of collision in the same $path \times time$ space, the vertex can be evaluated according to point four in **Section 3.2.4**. This is done for every subtrajectory, and if one of them fails the test, a replan in the deliberate layer is triggered.

3.4.2 Object Monitor

The object monitor performs two tasks. One is to evaluate which of the observable objects that need to be considered in the trajectory planning. This ensures that unnecessary objects are kept out of the node search problem, and thereby reduces the complexity and the runtime. This be, for example, a situation with a stationary fishing vessel that is not intersecting the path, or other vessels moving parallel to the path. The other task is to trigger a replan in the deliberate layer when a new object has to be considered, or when an old object no longer has to be considered. This to ensure that potential situations are handled with as good time margin as possible and that the trajectory is best suited for the current situation.

The second task is done simply by triggering a replan every time an object is added or removed from the list of objects to be considered.

The first task bases its decision making on the DCPA and the TCPA. The DCPA is the estimated shortest distance between an object and the ferry at any point in time. The distance is between the ferry and an object is

$$dist(t) = \sqrt{((N(t) - N_o(t))^2 + (E(t) - E_o(t))^2)}, \tag{3.51}$$

and hence the DCPA becomes

$$DCPA = min(dist(t)), \tag{3.52}$$

where $N(t)$ and $E(t)$ is the north and east coordinate of the ferry in $\{n\}$

$$N(t) = N_0 + U\cos(\psi)(t - t_0), \tag{3.53}$$
$$E(t) = E_0 + U\sin(\psi)(t - t_0), \tag{3.54}$$

where N_0 and E_0 is the position of the ferry at time t_0 and U is the velocity of the ferry. $N_o(t)$ and $E_o(t)$ is the north and east position of the object in $\{n\}$

$$N_o(t) = N_{0\_obj} + U_{0\_obj}\cos(\psi_{0\_obj})(t - t_{0\_obj}), \tag{3.55}$$
$$E_o(t) = E_{0\_obj} + U_{0\_obj}\sin(\psi_{0\_obj})(t - t_{0\_obj}), \tag{3.56}$$

where N_{0_obj} and E_{0_obj} is the position of the object at time t_{0_obj}, and $U_{0\_obj} > 0$ and ψ_{0_obj} is the object SOG and COG respectively. Since the transit is happening within a limited scope of time and path, it is not of interest what the DCPA is outside of this scope. Therefore an estimate of the time at destination is needed. The estimate does not need to be precise, and a little overshoot is better than undershooting. The estimate can be as simple as

$$t_{destination} = \frac{\sqrt{(N - N_{destination})^2 + (E - E_{destination})^2}}{U_{des} * k_{vel}}, \tag{3.57}$$

where $N_{destionation}$ and $E_{destionation}$ is the north and east coordinate of the destination and U_{des} is the desired transit velocity. $k_{vel} \in (0, 1]$ is a factor that reduces the risk of undershooting in the estimate. And hence the DCPA becomes

$$DCPA = min(dist(t)) \quad t \in [t_0, t_{destination}], \tag{3.58}$$

where t_0 is the current time.

The DCPA is found by first finding the TCPA. The TCPA is the time where (3.51) has its minimum value and can be found at the time where the derivative of $dist(t)$ is zero. Since it is assumed that all objects are moving at a constant SOG and COG in the prediction of object movement, the DCPA will have one global minimum and will go towards infinity for $t \to \pm\infty$ unless both $\psi = \psi_{0\_obj}$ and $U = U_{0\_obj}$. For the latter case, the two vessels will have a constant DCPA for all time. For simplicity, the derivative of the square of the $dist(t)$ is found, since both will have a minimum at the same time.

$$dist(t)^2 = (N(t) - N_o(t))^2 + (E(t) - E_o(t))^2), \tag{3.59}$$
$$= (\Delta_N)^2 + (\Delta_E)^2, \tag{3.60}$$

with

$$\Delta_N = N_0 - N_{0\_obj} - U\cos(\psi)t_0 + U_{0\_obj}\cos(\psi_{0\_obj})t_{0\_obj} + (U\cos(\psi) + U\cos(\psi 0\_obj))t, \tag{3.61}$$

$$\Delta_E = E_0 - E_{0\_obj} - U\sin(\psi)t_0 + U_{0\_obj}\sin(\psi_{0\_obj})t_{0\_obj} + (U\sin(\psi) + U\sin(\psi 0\_obj))t, \tag{3.62}$$

which can be written on the form

$$\Delta_N = A + Bt, \tag{3.63}$$

$$\Delta_E = C + Dt, \tag{3.64}$$

with

$$A = N_0 - N_{0\_obj} - U\cos(\psi)t_0 + U_{0\_obj}\cos(\psi_{0\_obj})t_{0\_obj}, \tag{3.65}$$

$$B = U\cos(\psi) + U\cos(\psi_{0\_obj}) \tag{3.66}$$

$$C = E_0 - E_{0\_obj} - U\sin(\psi)t_0 + U_{0\_obj}\sin(\psi_{0\_obj})t_{0\_obj}, \tag{3.67}$$

$$D = U\sin(\psi) + U\sin(\psi_{0\_obj}). \tag{3.68}$$

This gives the derivative of the square DCPA

$$\frac{d}{dt}dist(t)^2 = \frac{d}{dt}(A + Bt)^2 + (C + Dt)^2) \tag{3.69}$$

$$= \frac{d}{dt}A^2 + 2ABt + B^2t^2 + C^2 + 2CDt + D^2t^2 \tag{3.70}$$

$$= 2(AB + CD) + 2(B^2 + D^2)t \tag{3.71}$$

The TCPA is found by solving for $\frac{d}{dt}dist(t)^2 = 0$, giving

$$t = TCPA = -\frac{AB + CD}{B^2 + D^2} \tag{3.72}$$

Inserting this result into (3.63)-(3.64), and further inserting that into (3.60) gives

$$DCPA^2 = \frac{(AD - CB)^2}{B^2 + D^2}, \tag{3.73}$$

and finally, applying the square root to get

$$DCPA = \sqrt{\frac{(AD - CB)^2}{B^2 + D^2}}, \tag{3.74}$$

with A, B, C and D from (3.65)-(3.68). Note that if $t \notin [t_0, t_{destination}]$ whichever limit is being violated is to be used instead.

With the DCPA, the logic for determining what objects to consider can be formulated. Intuitively, it is desirable to consider objects that will be in proximity to the ferry during the transit, so a boolean value

$$bool_{DCPA} = (DCPA <= l_{DCPA}), \tag{3.75}$$

can be used to categorize the objects as ”objects to be considered” or ”objects not to be considered”. Here l_{DCPA} is a threshold value, in meters, for what is acceptable DCPA. This value is a tuning parameter and does not have to be the same for all objects, it can, for example, be a function of the object size.

The DCPA and TCPA calculations presented here assume that both the ferry and the moving object keep a constant COG and COG. These assumptions will not hold in most cases, but for this purpose, it will be sufficient. In (Sang et al., 2016), a method for improved DCPA and TCPA estimations is presented. The method uses the Rate Of Turn (ROT) and Change Of Speed (COS) to improve the predictions of the object trajectories before calculating the DCPA and TCPA. In the article, the ROT and COS values come from an AIS system, which is not realistic for the scenario considered in this thesis, since AIS is not standard equipment for small vessels. An advanced object detection and tracking system could potentially provide an estimate for the needed values. Another approach is to use knowledge about static environments to predict object behaviour. With a priori information on sea-markings, harbours and commercial traffic in the area, a statistical model for the object trajectory can be used in the predictions. Another simple way to improve the DCPA and TCPA predictions is to use the planned trajectory, and not the current velocity and heading from the ferry, but for the sake of limiting the scope, this is left for future work.

Chapter 4

Simulation Results

In this chapter, the simulator, simulation scenarios and simulation results are presented. Firstly a system overview of the ferry and GNC system in the simulator is presented, followed by some in-depth descriptions on the features of the simulator. Subsequently follows a description of some supporting modules to the simulator that emulate COLAV-critical modules on the ferry that is currently under development, and therefore not ready for use. In **Section 4.2** the simulation scenarios are described. In the last four sections of this chapter, the simulation results are presented and discussed. Firstly results from the SP-VP and MP-VP system is presented in **Section 4.3** and **Section 4.4** respectively. In these sections, the basic function of the algorithms is commented on. In **Section 4.5**, the performance of the two systems are evaluated and compared to each other as well as an implementation of a Velocity Obstacle COLAV algorithm. The chapter is rounded of by **Section 4.6**, with a summary of the discussions.

4.1 Simulator

Despite having a full-scale functional prototype of an electric autonomous passenger ferry available for testing purposes, it might not be the method of choice for developing and testing a COLAV system. Apart from the obvious reason (the risk of colliding), the time consumption is a high motivation for doing testing in a simulation-environment. In addition to this, the milliAmpere does not yet have a sufficient situational awareness system to deliver the required extended object tracking-data required by the COLAV system presented in this report. Therefore a simulator was developed. The simulator development is a continuation of the work done in (Thyri, 2018), where the thruster model, ROS-framework and a vessel model based on a model of the C/S Inocean Cat Drillship from (Bjørnø, 2016) was implemented. In this thesis, the simulator is augmented with a coupled vessel model of the milliAmpere platform and an emulated situational awareness in the form of an object-detection system. The new vessel model is included in the report, while the thruster-modeling is included in **Appendix A** for completeness.

Since there is a number of people working on the milliAmpere project, the simulator

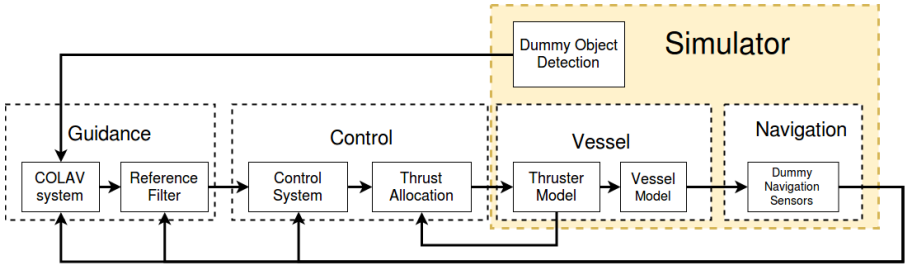


Figure 4.1: Overview of the vessel and GNC system during simulations. The yellow box is the simulator, the other parts are part of the actual ferry GNC system, and therefore the same for simulations and sea trials.

is developed to be generic and adaptable, so that anyone can understand it and fit it to their needs. In order to facilitate full pipeline testing before going out for sea trials, all the ROS-interfaces in the simulator is made the same as on the milliAmpere platform. The simulator is made so that it can run as a stand-alone ROS node, and therefore be run on the OBC of the ferry while it is at the dock as a penultimate test before sea trials. The simulator can also run in Matlab/Simulink and is therefore easy to use for everyone involved in the project. The simulator has also been implemented in python code. This was done to resolve some timing-issues experienced when running the simulator in Simulink while running other parts of the system independent from Matlab/Simulink. This problem occurred because there is no way (to the best of the author's knowledge) to synchronize the simulation time in Simulink with wall-time on a Linux-based OS.

In the following, the layout of the simulator is presented, the interfaces with the rest of the systems on the ferry are pointed out, and some of the features of the simulator are described in detail.

4.1.1 Simulator Layout

An overview of the system-layout for the ferry and simulator is displayed in **Fig. 4.1**. The yellow block contains all systems that are part of the simulator. These are systems that have a physical aspect, like the thrusters, sensors and the vessel itself, as well as the object detection, which simply does not exist yet, and needs to be emulated by the dummy object detection. In the remainder of this paragraph, a short description of each sub-module is given, while some of the sub-modules are described in more detail in the following paragraphs.

- **Dummy Object Detection:** Emulates multiple objects in the environments and supports a set of object behaviours. The module provides extended object tracking data to the COLAV system and is described in further detail in **Section 4.1.5**.
- **COLAV system:** This system is described in detail already. It receives object data and outputs a trajectory reference to the reference filter.
- **Reference Filter:** The reference filter takes the reference from the COLAV system

and makes a feasible pose, velocity and acceleration reference for the control system to follow.

- **Control System:** Is a model reference adaptive controller. The system inputs a pose, velocity and acceleration reference and outputs a 3 DOF force request to the thrust allocation. The system is described in detail in (Sæther, 2019).

- **Thrust Allocation:** inputs a 3 DOF force vector from the control system, and outputs RPM and azimuth angle setpoints to each of the two thrusters on the ferry. The thrust allocation problem is solved as a nonlinear scalar optimization-problem. The thrust allocation was developed the summer of 2018, by Tobias Torben.

- **Thruster Model:** A model of the two azimuth thrusters on the ferry. It inputs a rotational velocity setpoint and azimuth angle setpoint and outputs a 3 DOF force, as well as the actual rotational velocity and azimuth angle. The thruster model is described in detail in **Section 4.1.3**.

- **Vessel Model:** A 3 DOF model of the milliAmpere platform. Described in detail in **Section 4.1.2**.

- **Dummy Navigation Sensors:** Keeps track of the vessel states and outputs states through a ROS-interface on the same format as the navigation-system on the milliAmpere platform.

4.1.2 3 DOF Vessel Model

The vessel model used in the simulator is a 3 DOF model like the one presented in (2.6). The model parameters are determined by Anders Pedersen through the work done in his Master's thesis during the spring of 2019 (Pedersen, 2019). The parameters are estimated using linear regression on data from steady-state coupled and uncoupled motions with the milliAmpere platform, along with simulator-based methods on transient data.

The inertia matrix

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_{A}, \tag{4.1}$$

with

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & I_z \end{bmatrix}, \tag{4.2}$$

and

$$\boldsymbol{M}_{A} = \begin{bmatrix} -X_{\dot{u}} & -X_{\dot{v}} & -X_{\dot{r}} \\ -Y_{\dot{u}} & -Y_{\dot{v}} & -Y_{\dot{r}} \\ -N_{\dot{u}} & -N_{\dot{v}} & -N_{\dot{r}} \end{bmatrix}. \tag{4.3}$$

For the milliAmpere ferry, the CO coincides with the center of gravity. Since all sensor values are transformed to the CO, x_g is zero in (4.2) and (4.6). Due to challenges related to separating the added mass terms from the rigid body terms in the inertia matrix in the

experimental data, the sum of the terms are in stead estimated. The inertia matrix therefore becomes

$$\boldsymbol{M}_A = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}, \tag{4.4}$$

with $m_{11} = m - X_u$, $m_{22} = m - Y_v$ and $m_{33} = I_z - N_r$. The parameters are given in **Table 4.1**.

The centripetal matrix

$$\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu}), \tag{4.5}$$

with

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -m(x_g r + v) \\ 0 & 0 & mu \\ m(x_g r + v) & -mu & 0 \end{bmatrix}, \tag{4.6}$$

and

$$\boldsymbol{C}_A(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & Y_{\dot{r}} r + Y_{\dot{v}} v \\ 0 & 0 & -X_{\dot{u}} u \\ -Y_{\dot{r}} r - Y_{\dot{v}} v & X_{\dot{u}} u & 0 \end{bmatrix}, \tag{4.7}$$

also has all cross terms assumed zero, along with $x_g = 0$. The centripetal matrix becomes

$$\boldsymbol{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & c_{13} \\ 0 & 0 & c_{23} \\ c_{31} & c_{32} & 0 \end{bmatrix}, \tag{4.8}$$

with $c_{13} = -m_{22} v$, $c_{23} = m_{11} u$, $c_{31} = -c_{13}$ and $c_{32} = -c_{23} v$. The damping matrix

$$\boldsymbol{D}(\boldsymbol{\nu}) = \boldsymbol{D}_t + \boldsymbol{D}_n(\boldsymbol{\nu}), \tag{4.9}$$

with the linear damping

$$\boldsymbol{D}_t = \begin{bmatrix} -X_u & -X_v & -X_r \\ -Y_u & -Y_v & -Y_r \\ -N_u & -N_v & -N_r \end{bmatrix}, \tag{4.10}$$

and the non-linear damping

$$\boldsymbol{D}(\boldsymbol{\nu}) = \begin{bmatrix} D_{11}(\boldsymbol{\nu}) & 0 & 0 \\ 0 & D_{22}(\boldsymbol{\nu}) & D_{23}(\boldsymbol{\nu}) \\ 0 & D_{33}(\boldsymbol{\nu}) & D_{32}(\boldsymbol{\nu}) \end{bmatrix}, \tag{4.11}$$

with

$$D_{11}(\boldsymbol{\nu}) = -X_u - X_{|u|u}|u| - X_{uuu} u^2, \tag{4.12}$$
$$D_{22}(\boldsymbol{\nu}) = -Y_v - Y_{|v|v}|v| - Y_{vvv} v^2, \tag{4.13}$$
$$D_{23}(\boldsymbol{\nu}) = -Y_v - Y_{|v|v}|v| - Y_{vvv} v^2, \tag{4.14}$$
$$D_{32}(\boldsymbol{\nu}) = -N_{|v|v}|v| - N_{|r|v}|r| \tag{4.15}$$
$$D_{33}(\boldsymbol{\nu}) = -N_r - N_{|r|r}|r| - N_{rrr} r^2. \tag{4.16}$$

Parameter	Value	Unit
m_{11}	2131.80	kg
m_{12}	1.00	kg
m_{13}	141.02	kgm
m_{21}	-15.87	kg
m_{22}	2231.89	kg
m_{23}	-1244.35	kgm
m_{31}	-423.76	kgm
m_{32}	-397.64	kgm
m_{33}	4351.56	kgm^2
X_u	-68.676	$\frac{kg}{s}$
$X_{\|u\|u}$	-50.08	$\frac{kg}{m}$
X_{uuu}	-14.93	$\frac{kgs}{m^2}$
X_v	-25.20	$\frac{kg}{s}$
X_r	-145.30	$\frac{kgm}{s}$
Y_u	90.15	$\frac{kg}{s}$

Parameter	Value	Unit
Y_v	-8.69	$\frac{kg}{s}$
$Y_{\|v\|v}$	-189.08	$\frac{kg}{m}$
Y_{vvv}	-0.00613	$\frac{kgs}{s^2}$
$Y_{\|r\|v}$	–3086.95	kg
Y_r	-24.09	$\frac{kgm}{s}$
$Y_{\|v\|r}$	-338.32	kg
$Y_{\|r\|r}$	1372.06	kgm^2
N_u	-38.00	$\frac{kgm}{s}$
N_v	-97.26	$\frac{kgm}{s}$
$N_{\|v\|v}$	-18.85	kg
$N_{\|r\|v}$	5552.23	kgm
N_r	-230.19	$\frac{kgm^2}{s}$
$N_{\|r\|r}$	-0.0063031	kgm^2
N_{rrr}	-0.0006723	$kgms$
$N_{\|v\|r}$	-5888.89	kgm

Table 4.1: Estimated model parameters for the milliAmpere ferry (Pedersen, 2019).

The model parameters are given in **Table 4.1**.

The simulator solves

$$\boldsymbol{\nu}(t) = \int_{t_0}^{t} \boldsymbol{M}^{-1}(\boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu})dt + \boldsymbol{\nu}(t_0), \tag{4.17}$$

and

$$\boldsymbol{\eta}(t) = \int_{t_0}^{t} \boldsymbol{R}(\psi)\boldsymbol{\nu}dt + \boldsymbol{\eta}(t_0), \tag{4.18}$$

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

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{thruster} + \boldsymbol{\tau}_{external}, \tag{4.19}$$

is the forces on the vessel where $\boldsymbol{\tau}_{thrusters}$ come from the thruster model (A.2), and

$$\boldsymbol{\tau}_{external} = \boldsymbol{R}(\eta)^T \boldsymbol{F}_{external}, \tag{4.20}$$

is the external forces acting on the vessel, where $\boldsymbol{F}_{external}$ can be defined in the simulator initialization file, and $\boldsymbol{R}(\eta)$ is given by (2.1).

4.1.3 Thruster Model

The thruster system on milliAmpere is a set of two azimuth thrusters capable of 360 degrees rotation. The thrusters are positioned along the centerline of the vessels front-aft axis, and symmetrical about the port-starboard axis. The motor drivers have a ROS interface where the input is a rotational velocity setpoint ω_{set} and an azimuth-angle setpoint θ_{set}, and the output is the actual rotational velocity and azimuth-angle, that is fed into the

feedback-loop of the thrust-allocation. The thruster model was developed by the author and was first presented in (Thyri, 2018). The modeling is also included in **Appendix A**.

4.1.4 Reference Filter

The reference filter is not part of the simulator but is a part of the GNC pipeline. The filters were introduced and explained in detail in **Section 3.2.7**.

Filter Saturation Limits

In order to determine the the filter saturation limits in acceleration as well as velocity, a step input test was performed on the vessel model described in the preceding paragraphs. The thrust input has two steps, at $t = 5s$ a step from zero to maximum thrust in the body x-direction, namely $\boldsymbol{\tau} = [1000, 0, 0]^T$. At $t = 40s$ the direction of the thrust changes to $\boldsymbol{\tau} = [-1000, 0, 0]^T$. The velocity and acceleration response can be seen in **Fig. 4.2**. The step input test yielded velocity limits of $[U_{max}, U_{min}] = [2.930, -2.930]m/s$ and acceleration limits of $[a_{max}, a_{min}] = [0.466, -0.932]m/s^2$. From this, one can see that the

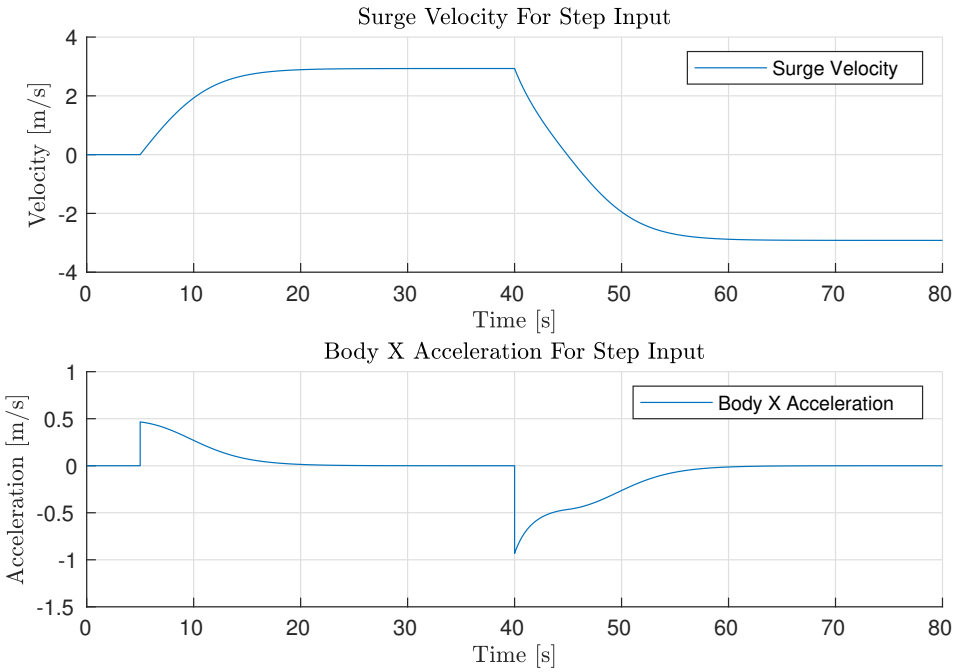


Figure 4.2: Vessel response on step input in thrust.

acceleration capabilities of the vessel are within the limits of passenger comfort in all situations, and both the comfort filter and response filter needs to be saturated in acceleration according to (3.38) to ensure a feasible reference.

Determining Filter Error

The testing of the reference filters was done by firstly defining a reference with steps in velocity, similar to the references the COLAV system provides and feeding that reference through the reference filters. Subsequently, the reference is fed into a trajectory tracking control system that provides a 3 DOF thrust that is used as input to the simulator equation in (4.17). The simulations were run without external forces, and the reference was constructed with only surge velocity, travelling with a heading $\psi = 0$.

The reference is constructed to represent realistic situations for the ferry. It is a 140 second transit along a straight line, travelling north, with $\psi = 0$. The velocity profile for the transit can be seen in **Fig. 4.3** as the *Hard Velocity Reference*. The first part of the reference represents typical transit situations, where small step changes in the velocity are requested. The last part of the reference at $45s$ represents a critical situation where the velocity steps from $1.5m/s$ to $-1.5m/s$. **Fig. 4.3** also shows the velocity reference from both the Comfort Filter and the Response Filter.

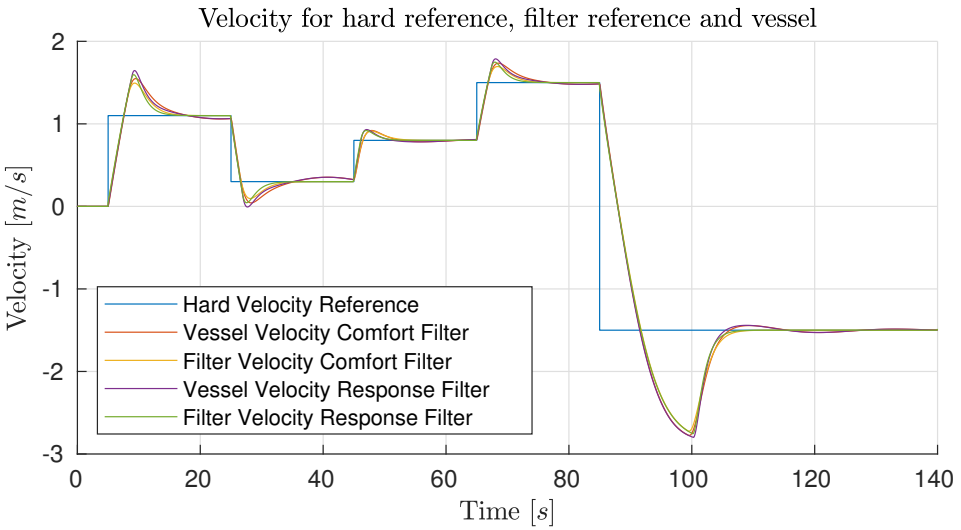


Figure 4.3: Velocity profiles for the *Hard Reference* with steps in the velocity, the velocity reference from both reference filters, as well as the vessel velocity in both cases.

Fig. 4.4 shows the tracking error introduced by the filter

$$\boldsymbol{e}_{filter} = \boldsymbol{\eta}_d - \boldsymbol{r}^n, \tag{4.21}$$

as well as the total tracking error

$$\boldsymbol{e}_{total} = \boldsymbol{\eta} - \boldsymbol{r}^n, \tag{4.22}$$

where $\boldsymbol{r}_n$ is the hard reference. **Fig. 4.5** shows the absolute value of the difference in error

$$\boldsymbol{e}_{diff} = abs(\boldsymbol{e}_{comfort\_filter} - \boldsymbol{e}_{response\_filter}), \tag{4.23}$$

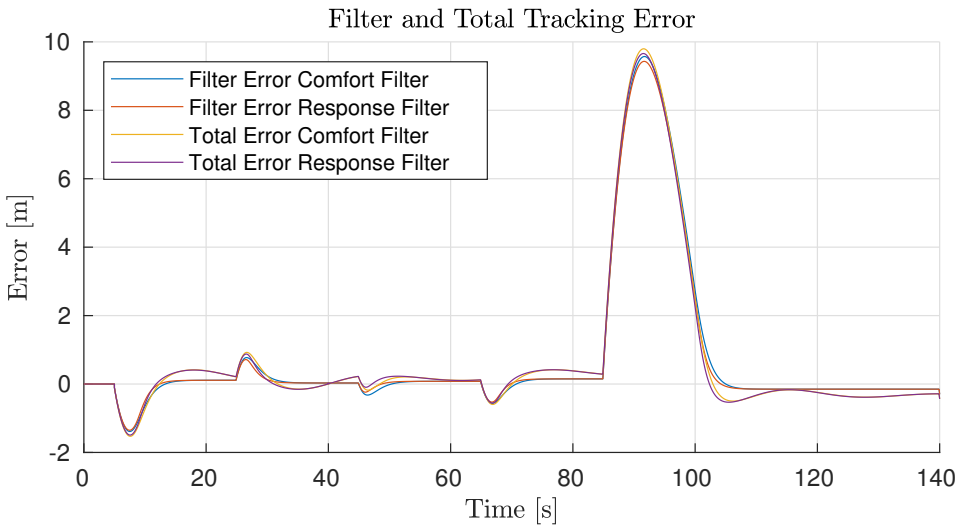


Figure 4.4: Filter error and total error for both the Comfort Filter and Response Filter with saturation matching the physical limitations of milliAmpere.

where $\boldsymbol{e}_{comfort\_filter}$ and and $\boldsymbol{e}_{response\_filter}$ is the error in (4.21) for the comfort filter, and response filter respectively.

From **Fig. 4.3-4.5** it is apparent that the difference in performance of the two filters is small. This is due to the low maximum thrust of the thruster system on the vessel milliAmpere. The acceleration limits of the vessel saturate both filters, and hence only the saturation in jerk for the comfort filter differs them.

Fig. 4.6 displays the same data as **Fig. 4.4**, for a vessel identical to the milliAmpere ferry, but with a thruster system capable of $6000N$ of thrust. From the figure, it is apparent that the comfort filter reduces the total tracking capabilities of the vessel-filter system compared to the response filter by a significant amount in the critical situation. The increased filter tracking error is about $3m$ higher with the comfort filter, while in the normal-operation-part of the reference, the performance is more comparable, with a difference of about $0.5m$. With a logic unit able to detect a critical situation, for example by looking at the steps in reference velocity, or cases where the velocity is negative, switching between the two reference filters can be done automatically. This can give the benefit of maximum passenger comfort as well as high response and increased safety.

By determining the tracking performance of the system consisting of the vessel-filter-system, with the two filters along with a switching unit, the maximum tracking error can be added to the dimensions determining the forbidden regions surrounding the moving objects. By doing this it can be assured that COLAV system plans a trajectory that ensures no collisions, given the assumptions on observable objects as well as reasonable object behaviour.

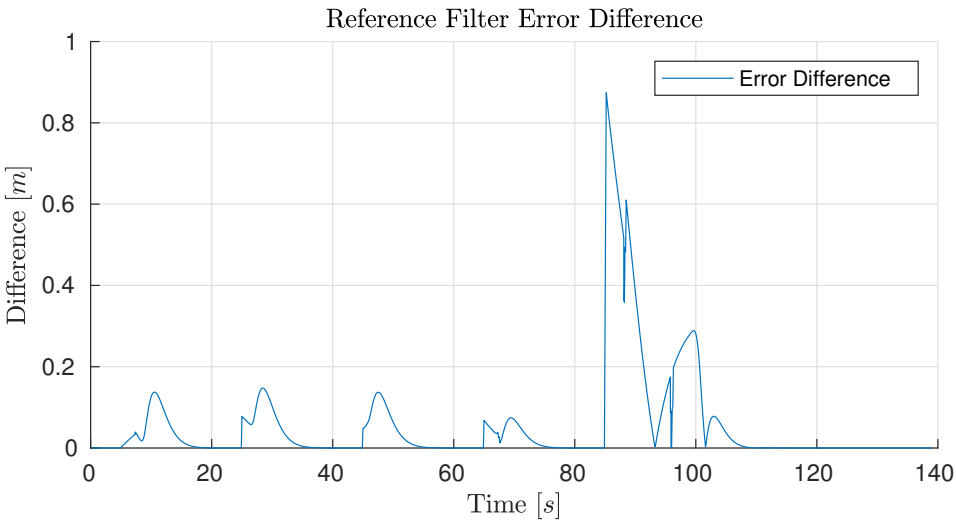


Figure 4.5: Filter error and total error for both the Comfort Filter and Response Filter with saturation matching the physical limitations of milliAmpere.

Structure Field	Parameter	Unit
η_{obj_i}	$(N_{obj\_i}, E_{obj\_i}, \psi_{obj\_i})$	$([m], [m], [rad])$
$Velocity_{obj\_i}$	U_{obj_i}	$[m/s]$
$Size_{obj\_i}$	(l_{obj_i}, b_{obj_i})	$([m], [m])$

Table 4.2: Information about object *obj_i* that is available through the *get_object_data()* interface.

4.1.5 Dummy Object Detection

An object detection module has been developed to facilitate testing and simulation of the COLAV system. The module emulates the interface of an object detection system that is capable of extended object tracking. The module has the interface *get_object_data()* that returns an array of data structures, one for each object. The data structure contains information about the object position, heading, velocity and size. **Table 4.2** shows the parameters that are stored in the structure of Object i. The module has been augmented with a set of ”behaviours” that is to resemble human operator behaviour. The behaviours are designed to emulate recreational vessels driven by non-professionals in a confined space. The movements of an object only adapt to the ferry, and not any other objects. The behaviours are described in detail in the following paragraphs. The Object detection module fetches information about the object from an initialization file upon simulation start-up and integrates the η_{obj_i} and U_{obj_i} according to the defined behaviour and the state of the object. The object detection module can also add noise on the ”estimate” of both object heading and object velocity. It is also possible to adjust the region of observation, by setting a parameter $r_{observation}$, where only objects inside a circle with centre in the ferry coordinates (N, E) will be provided information on through the *get_object_data()* interface.

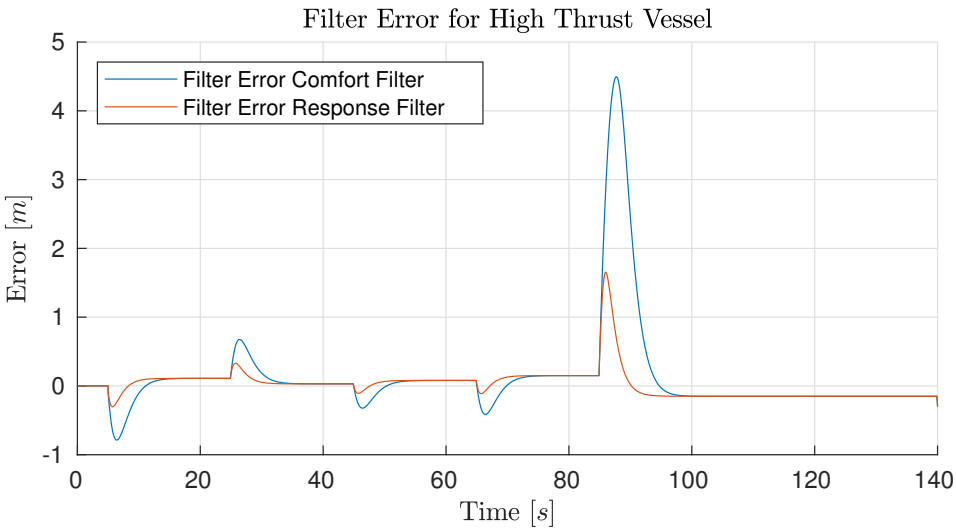


Figure 4.6: Filter error for comfort filter and response filter designed for a vessel identical to miliAmpere, but with a thruster system capable of $6000N$ of thrust.

Behaviour 1 - Constant object behaviour

With this behaviour, the objects move at a constant speed and constant heading. They do not react to the behaviour of the ferry, nor any aspect of the environments. This is in line with what is assumed about the objects in the projection of the objects onto the $path \times time$ space.

Behaviour 2 - Slow Down

The behaviour of the moving objects are defined by their distance to the ferry. Two circles with center in current position of the ferry, $(x, y) = (N, E)$, and radius $r_{v\_startramp}$ and $r_{v\_stopramp}$ where $r_{v\_startramp} \geq r_{v\_stopramp}$ define the behaviour. If the object is outside $r_{v\_startramp}$ it keeps a constant velocity and heading like in scenario 1. If the object enter $r_{v\_stopramp}$ it will reduce its velocity v_{obj_i} as a linear function of the distance to the ferry according to

$$v_{obj\_i} = min(v_{0\_obj\_i}, v_{0\_obj\_i} \frac{r_{obj\_i} - r_{v\_stopramp}}{r_{v\_startramp} - r_{v\_stopramp}}), \tag{4.24}$$

until it hits zero velocity at $r_{v\_stopramp}$. Here $v_{0\_obj\_i}$ is the initial velocity of object i, and

$$r_{obj\_i} = \sqrt{(N - N_{obj\_i})^2 + (E - E_{obj\_i})^2}, \tag{4.25}$$

gives the distance between the position of the ferry and the position of the object obj_i. The $min()$ function in (4.24) prevents the objects from "backing up", and instead keeps the objects stationary when inside the circle given by $r_{stopramp}$.

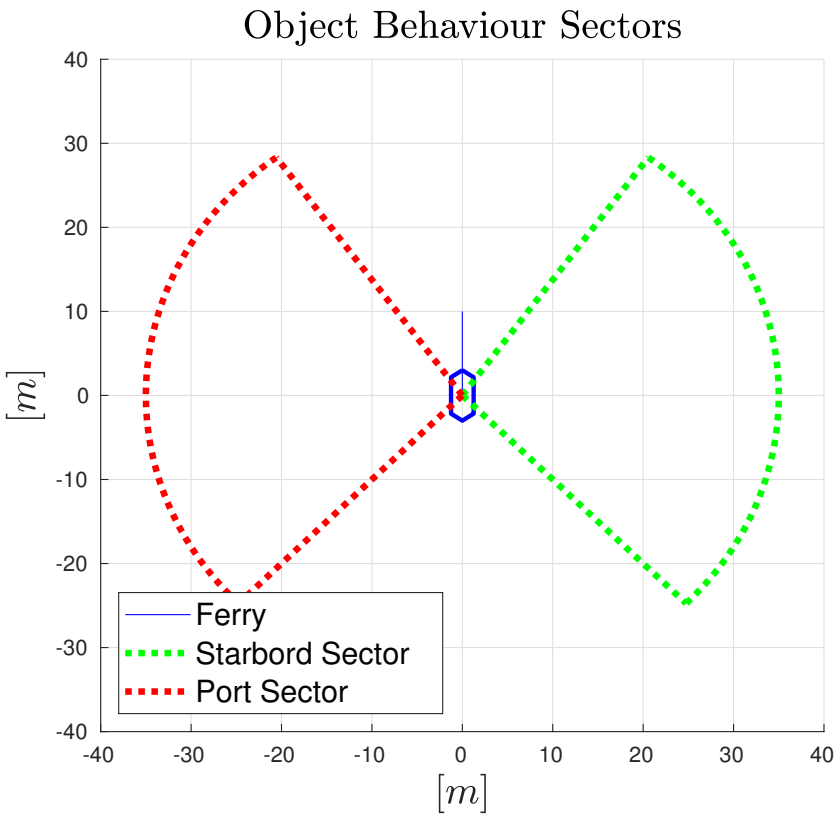


Figure 4.7: The two approaching sectors that defined the object behaviour.

Behaviour 3 - Pass In Front of Ferry

With Behaviour 3, the behaviour of the objects is based on the distance to the object, the objects angel of approach to the ferry and object heading. The behaviour is to be triggered when the object is approaching the ferry in one of the sectors in **Fig. 4.7**, and is traveling towards the ferry, with a relative heading $\psi_{rel\_obj\_i} \in [\pi/5, 6\pi/8]$ or $\psi_{rel\_obj\_i} \in [-6\pi/8, -\pi/5]$ depending whether it is in the port or starboard sector respectively. The angle of approach is given by

$$\alpha_{approach} = atan2((E_{obj\_i} - E), (N_{obj\_i} - N)), \tag{4.26}$$

where atan2 is the 2-argument arctangens function. The relative heading of the object is

$$\psi_{rel\_obj\_i} = \psi - \psi_{obj\_i}. \tag{4.27}$$

The object keeps constant velocity and heading as long as it is clear of the given regions with the corresponding relative headings. If the object enters the region and fulfills the heading requirements it will increase its velocity by a factor k_{inc_vel} and adjust its heading

to aim for a point with a distance l_{fore} straight in front of the ferry. The velocity of the object is then given by

$$U_{obj\_i} = k_{inc\_vel} U_{0\_obj\_i}, \tag{4.28}$$

and the object heading

$$\psi_{obj\_i} = atan2((E_{fore} - E_{obj\_i}), (N_{fore} - N_{obj\_i})), \tag{4.29}$$

with

$$(N_{fore}, E_{fore}) = (N + l_{fore} cos(\psi), E + l_{fore} sin(\psi)). \tag{4.30}$$

This will keep up until the object exits the region. Upon exit, it will go back to the initial heading in a smooth transition given by

$$\psi^{t+1}_{obj\_i} = \alpha_\psi \psi^t_{obj\_i} + (1 - \alpha_\psi), \psi_{0\_obj\_i} \tag{4.31}$$

$$v^{t+1}_{obj\_i} = \alpha_{vel} v^t_{obj\_i} + (1 - \alpha_{vel}) v_{0\_obj\_i}, \tag{4.32}$$

where $\alpha_\psi \in [0, 1]$ and $\alpha_{vel} \in [0, 1]$ are smoothing factors, and $\psi^t_{obj\_i}$ and $v^t_{obj\_i}$ are the heading and velocity of object i in the timestep t respectively.

Behaviour 4 - Pass Behind Ferry

Behaviour 4 is similar to Behaviour 3. The behaviour is triggered in the same manner, but the object will instead aim for a point that lies a distance l_{aft} straight behind the ferry. The object velocity is given by (4.28), while the object heading becomes

$$\psi_{obj\_i} = atan2((E_{aft} - E_{obj\_i}), (N_{aft} - N_{obj\_i})), \tag{4.33}$$

with

$$(N_{fore}, E_{fore}) = (N + l_{aft} cos(\psi), E + l_{aft} sin(\psi)). \tag{4.34}$$

Once the behaviour is over, the smooth transition back to $v_{0\_obj\_i}$ and $\psi_{0\_obj\_i}$ is the same as for Behaviour 3.

Behaviour 5 - Follow COLREG

With Behaviour 5, the objects are acting according to COLREGs Rule 8 and Rule 15. If the object is approaching the vessel from the port side, it will try to pass behind the vessel, and if it approaches from the starboard side it will try to pass in front of the vessel since it has the right of way. The behaviour is triggered in the same way as in Behaviour 3. The velocity and heading of the object are calculated in the same way as in Behaviour 3 and Behaviour 4, depending on whether it is the give-way or stand-on vessel.

4.2 Scenario Overview

The results presented in this thesis is from a set of simulated scenarios. The scenarios are combinations of two different crossings, namely Crossing 1 and Crossing 2, and the five different object behaviours described in **Section 4.1.5**. For simplicity, the scenarios are given numbers as names, where the number have no other purpose than relieving the author from coming up with more original names. **Table 4.3** contains an overview of the simulated scenarios that are included in this thesis. The table contains the scenario name, the subsection that contains the simulation data, the crossing number, the object behaviour and COLAV algorithm that is used.

Most of the simulations are run with four moving objects. This is a high traffic picture, especially when one considers the short transit-length of about $100m$. The motivation for this is to stress test the COLAV system, and the objects are initialized in such a way that the ferry is to get "caught" in a situation mid-way, and not wait at the start of the transit for the objects to pass. The single path planner was tested in Thyri (2018), with up to 6 moving objects, and has no problems finding a feasible trajectory. Therefore a situation with one or two objects and infinite region of observation might be too trivial for the planner, and not produce any interesting results. On Crossing 2, only two objects are used, since the scenarios are looking into the effect of region of observation. It is worth mentioning that in Thyri (2018) the transit is only planned once, and not run, so the ferry and object behaviour is not considered.

4.2.1 Crossing 1 - Straight Path, Short Crossing

Crossing 1 is a transit from Start at $[North, East] = [0, 0]$ to Finish at $[North, East] = [100, 30]$. The transit follows a straight line between the start and finish, and is supposed to represent the ferry crossing over a canal or a confined urban space. The motivation for this crossing is that it very much resembles a transit-area in Trondheim, where the milliAmpere ferry is supposed to perform fully autonomous transits. The area is very confined, but it has a layout that allows for good object monitoring. In this crossing, it is assumed that all relevant objects are observable, and the radius of observation is therefore set to infinite.

4.2.2 Crossing 2 - Straight Path, Long Crossing

Crossing 2 start at $[North, East] = [0, 0]$ and finish at $[North, East] = [80, 600]$. The transit follows a straight line between the start and finish, and represent the ferry crossing over a fjord, sound or between two islands in proximity to each other. The transit length of $605.3m$ is not of importance, and could just as well be $6km$, but for the sake of simulation time and data visualization, it is shorter. The motivation for this crossing is to see how the system handles a limited situational awareness. In such long crossings, it is not fair to assume that on-land monitoring systems can cover the whole operational area, and in some cases, the ferry may be limited to the on-board sensors. This is emulated by limiting the radius of observation.

Scenario	Section	Crossing	Behaviour	COLAV system
Scenario 1	4.3.1	1	1	SP-VP
Scenario 2	4.3.2	1	2	SP-VP
Scenario 3	4.3.3	1	3	SP-VP
Scenario 4	4.3.4	1	4	SP-VP
Scenario 5	4.3.5	1	5	SP-VP
Scenario 6	4.3.1	1	1	SP-VP
Scenario 7	4.3.1	1	1	SP-VP
Scenario 10	4.3.6	2	1	SP-VP
Scenario 11	4.3.6	2	1	SP-VP
Scenario 12	4.3.6	2	1	SP-VP
Scenario 20	4.4.2	2	1	MP-VP
Scenario 21	4.4.2	2	1	MP-VP
Scenario 22	4.4.2	2	1	MP-VP
Scenario 41	4.4.1	1	2	SP-VP
Scenario 51	4.4.1	1	2	MP-VP
VO-Scenario 1	4.5.2	1	1	VO
VO-Scenario 2	4.5.3	1	2	VO
VO-Scenario 3	4.5.4	1	3	VO
VO-Scenario 4	4.5.5	1	4	VO
VO-Scenario 5	4.5.6	1	5	VO

Table 4.3: Overview of all the simulated scenarios that are included in this Master's thesis. The number in the scenario name has not other purpose than relieving the author from coming up with original names for every scenario.

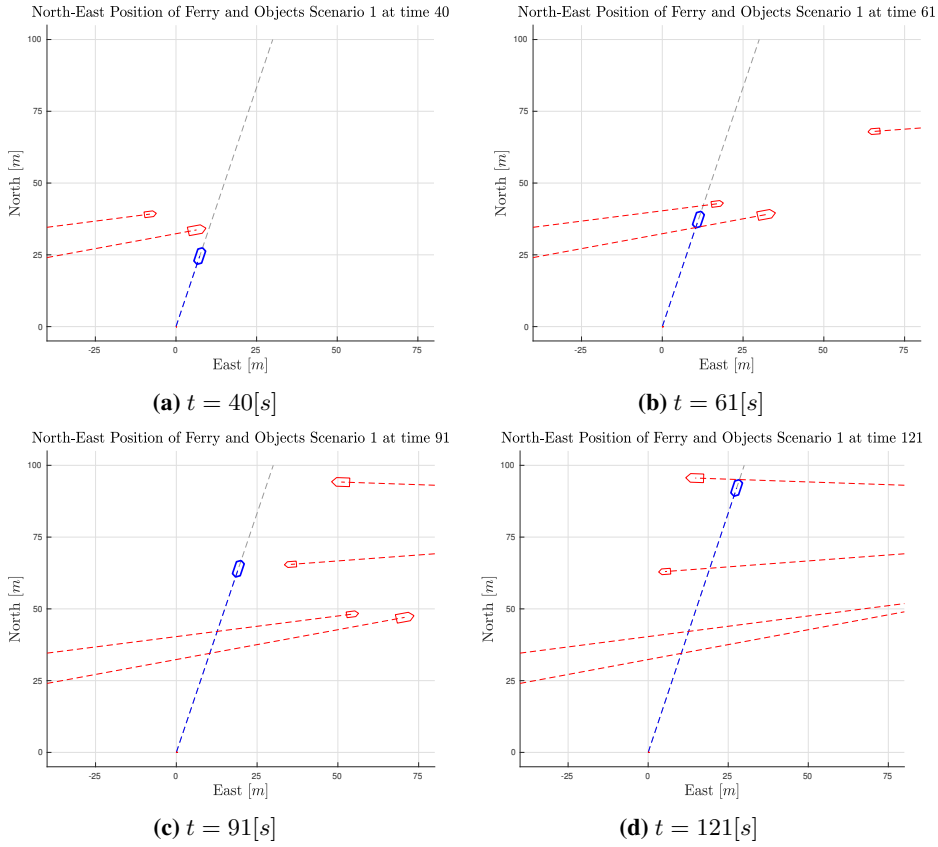
4.3 Single-Path Algorithm

In this section, the simulation results from the SP-VP algorithm are presented.

4.3.1 Scenario 1, 6 and 7 - Measurement Noise

Scenario 1, 6 and 7 are from Crossing 1 with constant object behaviour. In the scenarios, the objects keep constant heading and velocity, and are therefore acting according to the assumptions in the COLAV system and the object detection module does not have any measurement noise. Snapshots of the scenarios during transit can be seen in **Fig. 4.8**. The Distance to Closest Object (DCO) of Scenario 1 can be seen in **Fig. 4.9**. The velocity and planned velocity profiles can be seen in **Fig. 4.10a**. From the figures, one can see that the COLAV algorithm only performs one run of velocity planning, and follows the initial plan the whole trajectory, as would be expected since all of the assumptions hold throughout the transit.

Scenario 6 is the same as Scenario 1 but with the introduction of measurement noise on the estimate of object heading and velocity from the dummy object detection. From **Fig. 4.10b** it is visible that the system is sensitive to noise on the object behaviour. During the transit, the algorithm replans the trajectory five times in addition to the initial plan



(a) $t = 40[s]$ **(b)** $t = 61[s]$

(c) $t = 91[s]$ **(d)** $t = 121[s]$

Figure 4.8: Scenario 1: Overview of the transit. Ferry in blue and moving objects in red.

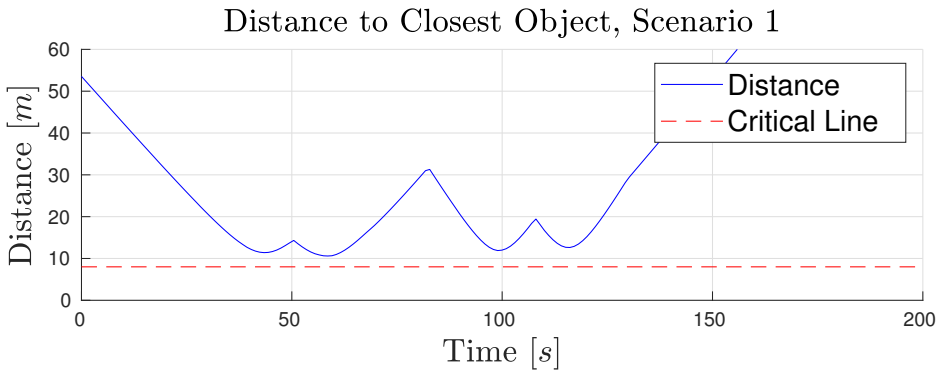


Figure 4.9: Scenario 1: Distance to closest object. Green dashed lines indicate the time of re-planning.

due to the current trajectory being evaluated to be unfeasible because it intersects with the regions of collision around at least one of the objects. This happens because of the previously mentioned limitations with the object representation from **Fig.3.3a** in **Section 3.2.1**. The representation is very sensitive to changes in object behaviour as well as noise in object states because it makes the projections of the objects onto the $path \times time$ space fluctuate. Scenario 7 is the same as Scenario 6, but with the introduction of the improved object representation presented in **Fig. 3.3b** in **Section 3.2.1**. From **Fig. 4.10c** one can see that under the same conditions, the algorithm only performs one replan during the transit. This is due to the increased robustness of the new object representation.

The remainder of the scenarios are run with the improved object representation and no measurement noise.

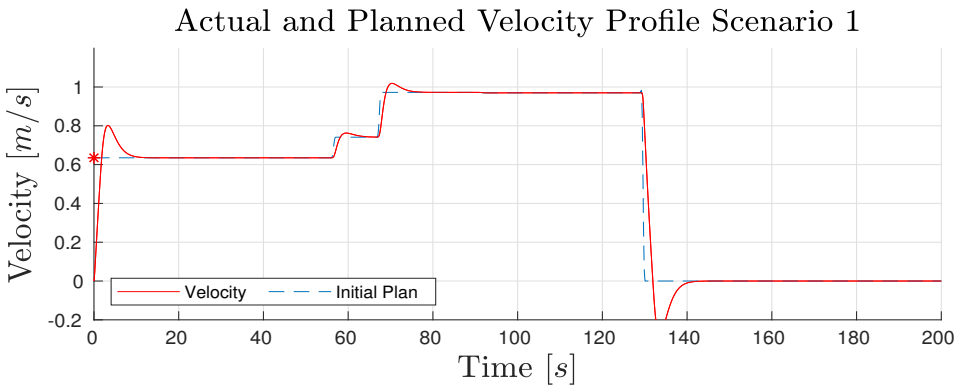
4.3.2 Scenario 2 - Slow Down

Fig. 4.11 contains snapshots of the transit in Scenario 1. In this scenario, the objects slow down as they approach the ferry. One thing to note from this scenario is the timestamps of the snapshots. From **Fig. 4.11b** and **Fig.4.11c**, one can see that it takes the ferry $40s$ to pass in front of the two objects approaching from the port side. **Fig. 4.12** gives an indication of why this is the case. The original plan of the ferry was to start off with a velocity below transit velocity and let the two objects pass in front, before accelerating up to transit velocity for the remainder of the transit. As the moving objects reduce their velocity, the ferry reduces velocity in response. The ferry and objects keep reducing their velocity until the objects come to a complete stop, and the ferry is able to find a collision-free trajectory that passes in front of the objects. After $121s$, another similar situation occurs. The ferry is planning to go behind the object after it has passed, but as the object reduces its velocity and stops, it does so in a way that blocks the path for the ferry and causes a deadlock. Even though the object is not physically interfering with the path of the ferry, the forbidden region that includes the object and safety margins does intersect the path, effectively blocking it, causing the ferry to stop until the simulation terminates. From **Fig. 4.13** one can see that the ferry is clear from any collisions during the transit, even when one of the objects come to a stop in its path. However, the longer the ferry has to wait for the path to clear up, the higher the risk of unexpected situations become, this factor is not modelled in any way in the COLAV system. The SP-VP does not have any way of handling objects that block the predefined path, which is a major limitation. A possible approach to solve this problem will be discussed in a later section.

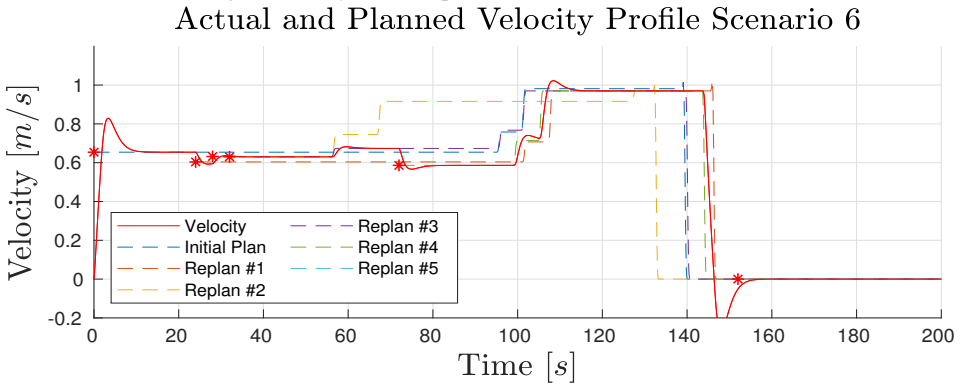
This scenario is not unrealistic to expect from some of the moving objects that the ferry will face in its operating environment, for example, small boats controlled by individuals that find an autonomous vessel interesting and stops to let it pass and get a good close look at it at the same time, not realizing how the COLAV system reacts to such behaviour.

4.3.3 Scenario 3 - Going in Front

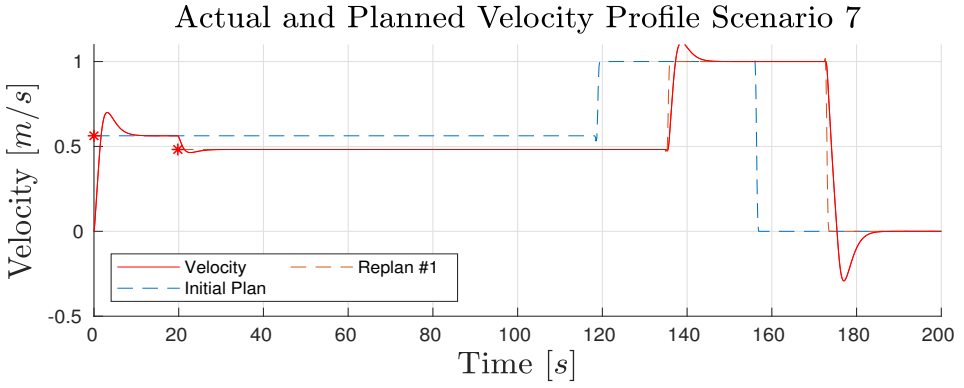
Fig. 4.14 shows six snapshots from Scenario 3, where the objects try to pass in front of the ferry. During the transit, the ferry encounters two situations. The first one occurs when the two objects coming in from the port side speeds up and changes course in order to pass in front of the ferry. From **Fig. 4.15** one can see that the algorithm replans twice during this



(a) Scenario 1: Velocity and planned velocity profile. Since the object behaviour is constant, and there is no noise, the algorithm only needs to plan once.

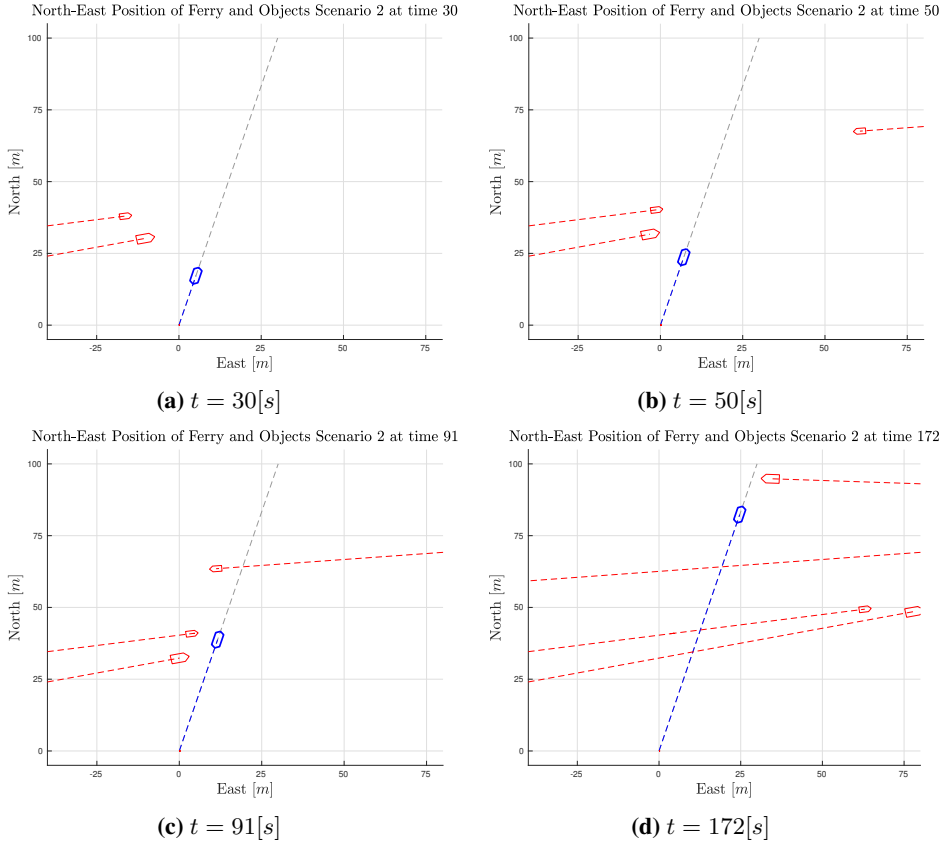


(b) Scenario 6: Velocity and planned velocity profile for Scenario 6. This is the same as in Scenario 1, but with measurement noise. The object representation is the initial one from **Section 3.2.1**. Note that the algorithm replans five times due to the disturbances.



(c) Scenario 7: Velocity and planned velocity profiles. This is the same as in Scenario 6, but with the improved object representation from **Section 3.2.1**. Note that the algorithm only replan once, in comparison to Scenario 6.

Figure 4.10: Scenario 1,6 and 7: Velocity and planned velocity profiles.



(a) $t = 30[s]$ **(b)** $t = 50[s]$

(c) $t = 91[s]$ **(d)** $t = 172[s]$

Figure 4.11: Scenario 1: Overview of the transit. Ferry in blue and moving objects in red.

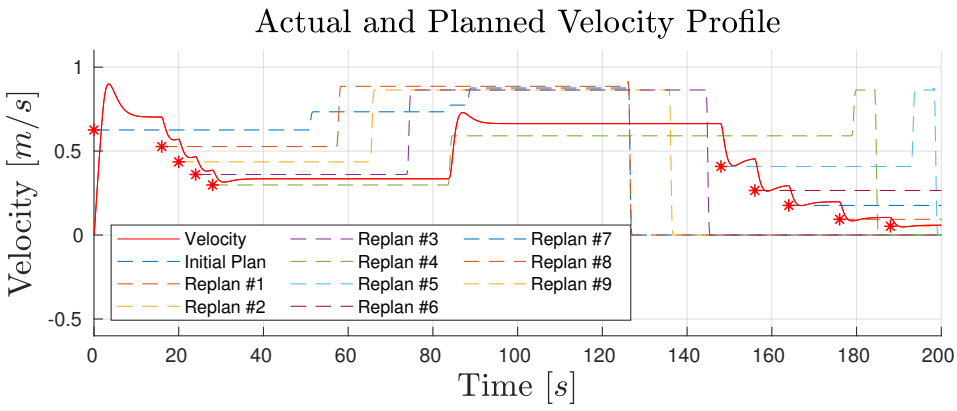


Figure 4.12: Scenario 2: Velocity and the set of planned velocity profiles. Red stars mark the time of re-planning.

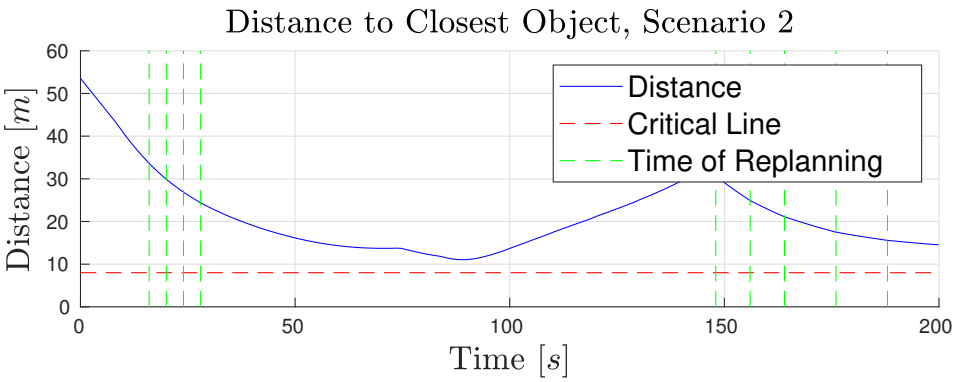


Figure 4.13: Scenario 2: Distance to closest object. Dashed green lines indicate the time of replanning.

situation, once to compensate for the first object, and once for the second object. In the second situation, only one replan is necessary in order to avoid a collision, and complete the transit. The DCO from **Fig. 4.16** shows that the first situation is handled without violating the critical line, while in the second situation, the DCO barely touches it. The low DCO in the second situation is due to the object changing its course to the south as it passes in front of the ferry, while the algorithm assumes that it will continue with the East-Northeast heading it had at last replanning. The duration of the violation is so short that the reactive layer fails to register it before it passes, and therefore does not compensate for it.

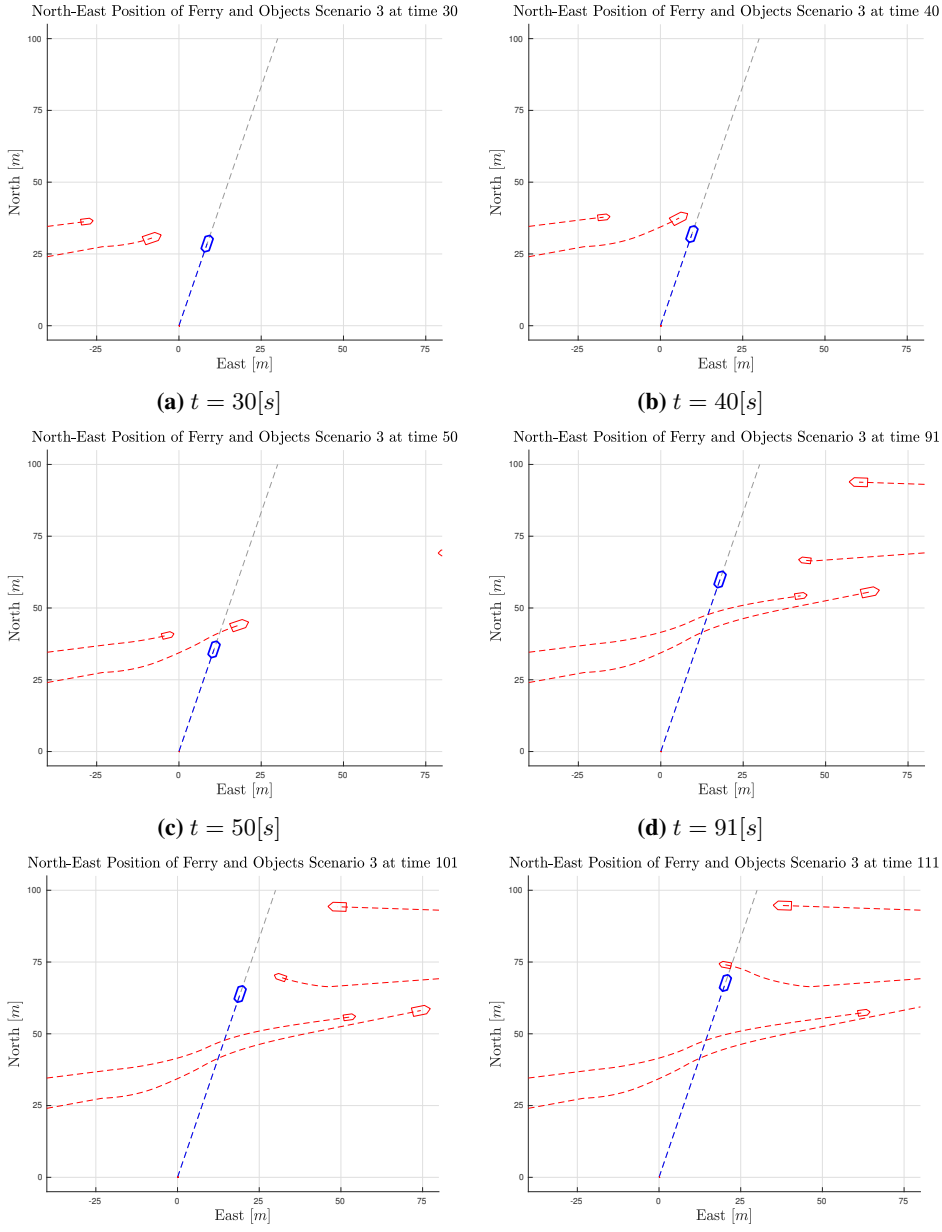
The second situation of this scenario, where the vessel from the right speeds up to pass in front of the ferry is a somewhat realistic one. The object has the right of way since it approaches from the starboard side of the ferry, and while they are not initially on a collision course, this might not be obvious from the view of the object operator.

4.3.4 Scenario 4 - Going Behind

In this scenario, the objects try pass behind the ferry. **Fig. 4.17** shows four snapshots from the transit. The initial plan from **Fig. 4.18** shows that the ferry was to pass behind the first two objects, but instead, the object adjusts course and velocity to go clear behind the ferry. The same is the case for the last of the objects in **Fig. 4.17d**.

Despite the unforeseen changes in the situation, the algorithm does not perform any replans, as can be seen form **Fig. 4.18**, where the initial plan and the actual trajectory coincide throughout the trajectory. This is because none of the changes in object heading or velocity make the initial trajectory infeasible.

This scenario did not introduce any challenges for the algorithm in terms of avoiding collisions, but one could argue that the system would benefit from replanning when rapid changes in the environments occur, such as objects drastically changing velocity and/or course. This could introduce a higher safety factor, wherein this case, it could lead to the ferry keeping a higher velocity and arrive before the last objects intersect the path. This could increase the DCO for the whole last part of the transit, as well as reduce the transit



(a) $t = 30[s]$ **(b)** $t = 40[s]$

(c) $t = 50[s]$ **(d)** $t = 91[s]$

(e) $t = 101[s]$ **(f)** $t = 111[s]$

Figure 4.14: Scenario 3: Overview during the transit. Ferry in blue and moving objects in red.

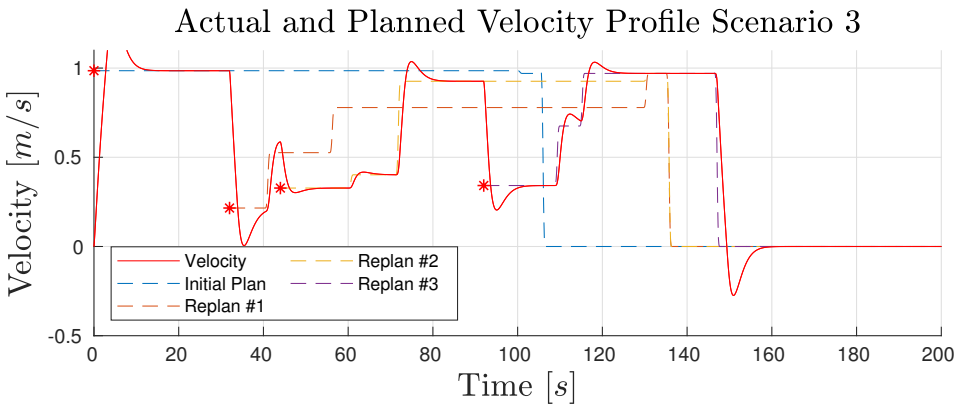


Figure 4.15: Scenario 3: Velocity and the set of planned velocity profiles. Red stars mark the time of re-planning.

time.

4.3.5 Scenario 5 - Follow COLREGs

Snapshots from Scenario 5, with objects acting according to Behaviour 5, "Follow COLREGs", can be seen in **Fig. 4.20**.

The initial plan was to pass behind the first two vessels, as can be seen from **Fig. 4.21**, but instead, the objects give way to the ferry. For the same reasons as in Scenario 4, the algorithm does not generate any reaction to this. For the next two objects, the ferry plans to move in front of them. When the objects speed up and alter their course to pass in front of the ferry, the plan is rendered unfeasible by the reactive layer. A replan is triggered, once for each object, and the algorithm reacts by reducing the velocity in order to let both objects pass.

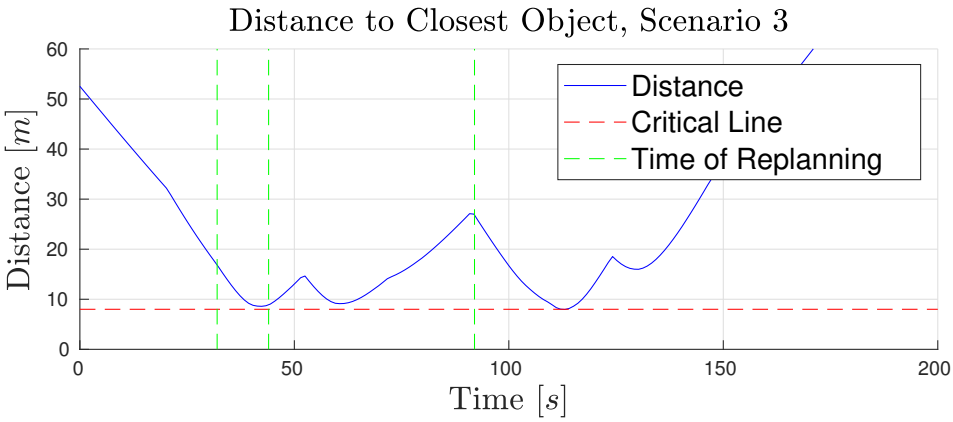
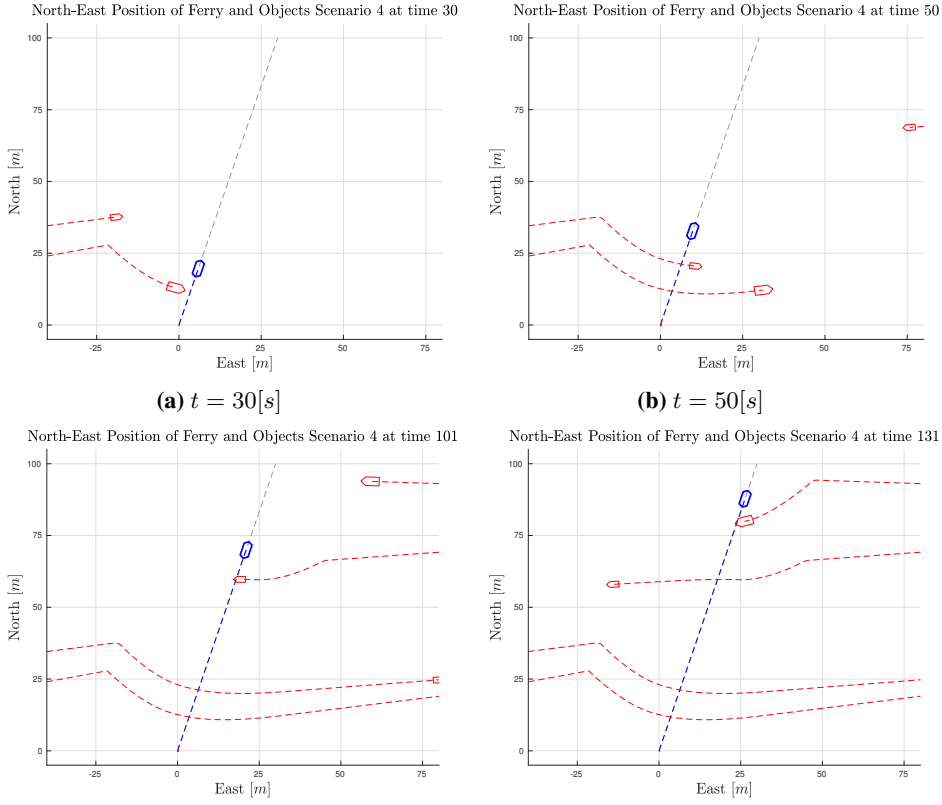


Figure 4.16: Scenario 3: Distance to closest object. Dashed green lines indicate the time of replanning.



(a) $t = 30[s]$ **(b)** $t = 50[s]$

(c) $t = 101[s]$ **(d)** $t = 131[s]$

Figure 4.17: Scenario 4: Snapshots during the transit. Ferry in blue and moving objects in red.

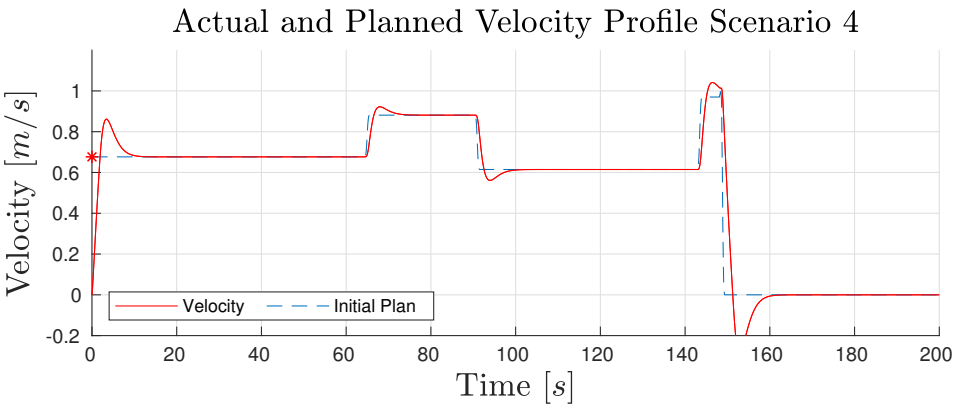


Figure 4.18: Scenario 4: Velocity and the set of planned velocity profiles. Red stars mark the time of replanning.

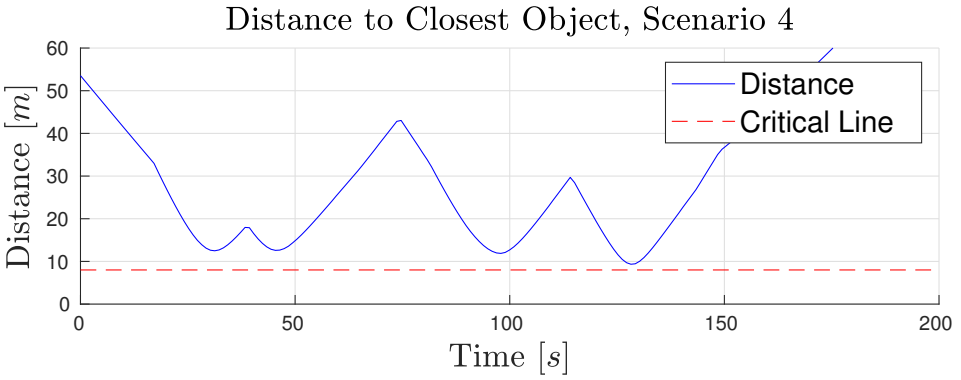
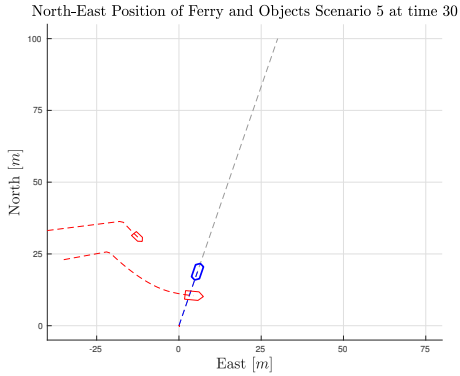
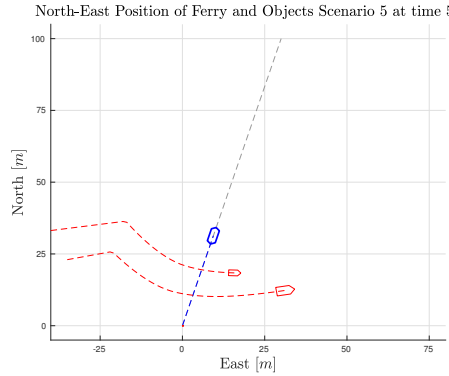


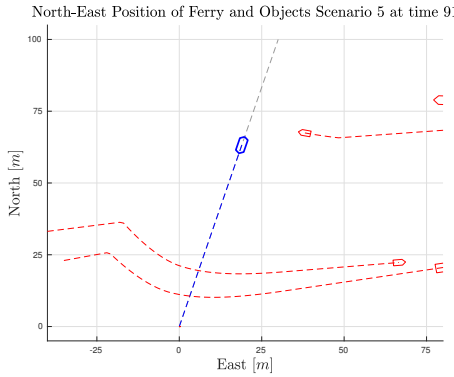
Figure 4.19: Scenario 4: Distance to closest object. Dashed green lines indicate the time of replanning.



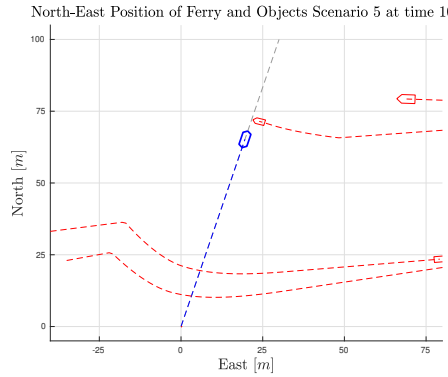
(a) $t = 30[s]$



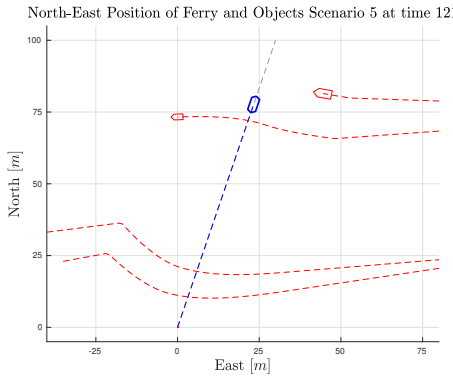
(b) $t = 50[s]$



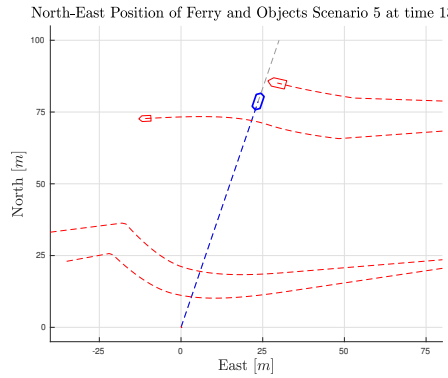
(c) $t = 91[s]$



(d) $t = 101[s]$



(e) $t = 121[s]$



(f) $t = 131[s]$

Figure 4.20: Scenario 5: Overview during the transit. Ferry in blue and moving objects in red.

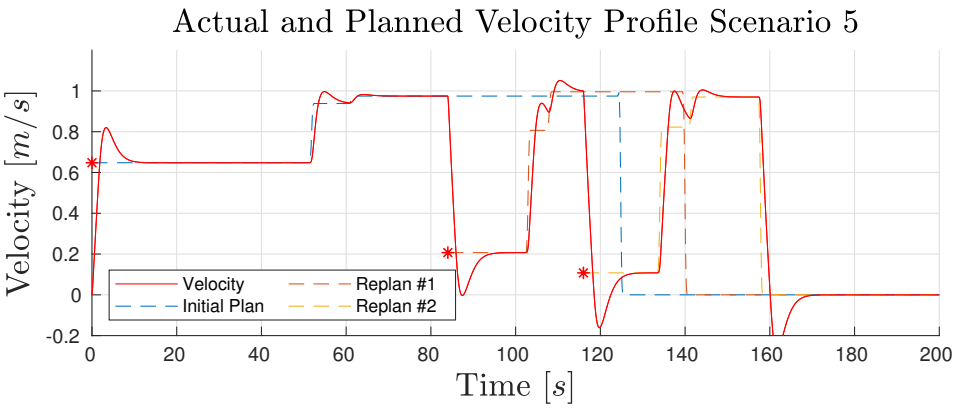


Figure 4.21: Scenario 5: Velocity and the set of planned velocity profiles. Red stars mark the time of trajectory planning.

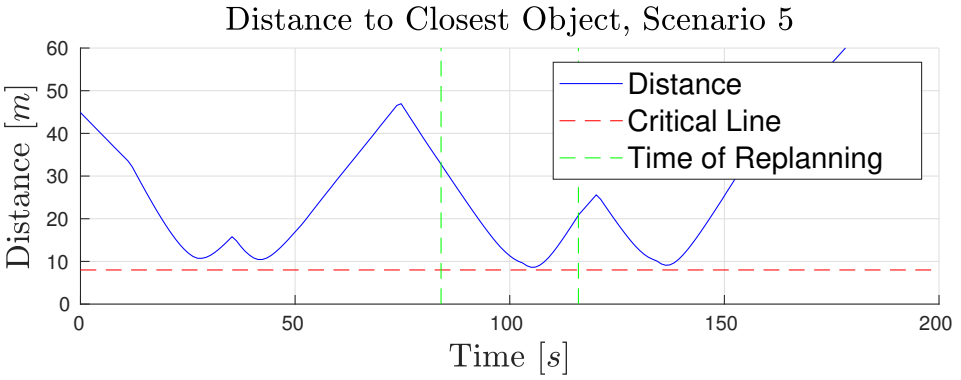


Figure 4.22: Scenario 5: Distance to closest object. Red line indicate the critical distance. Dashed green lines indicate the time of re-planning.

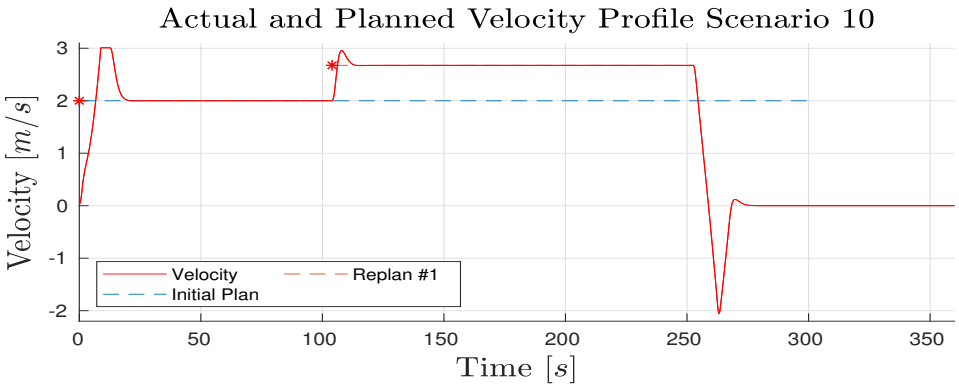
4.3.6 Scenario 10-12 - Region of Observation

This subsection presents the results from Scenario 10-12. The three situations have identical traffic, where two objects approach the path from the ferry port side. The objects have the same velocity and heading, and is moving one behind the other. The first object is initialized in a way that gives it a point of collision with the ferry $400m$ into the path if the ferry keeps the desired transit velocity of $2m/s$. The region of observation for the ferry is $450m$, $150m$ and $75m$ in Scenario 10, 11 and 12 respectively. **Fig. 4.23** give the velocity profile of the ferry, as well as the planned velocity profiles for the three situations.

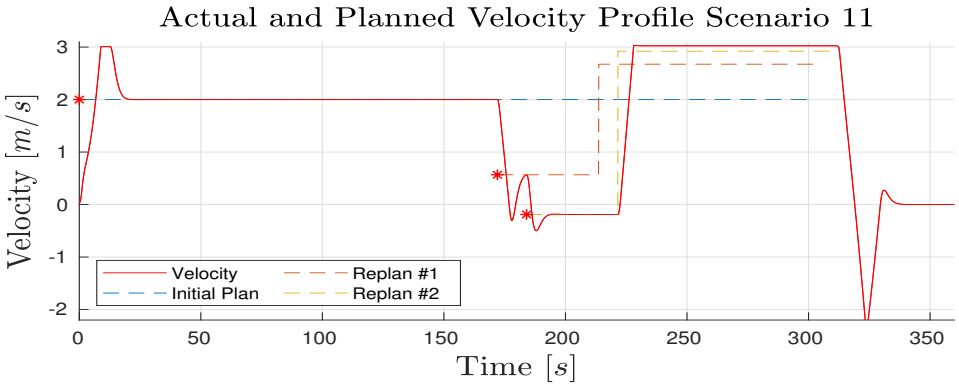
From **Fig. 4.23a** one can see that in Scenario 10, the COLAV algorithm reacts to the first object by slowing down to pass behind it. As the second object appears in the region of observation, it re-planned to speed up and pass in front of both objects. Due to the large region of observation of $450m$, the situation is reacted to with enough time to speed up and pass in front of both objects. In Scenario 11, with a smaller region of observation, there is not enough time to pass in front. Therefore, the ferry has to slow down to let the objects pass. From the velocity profile in **Fig. 4.23b** one can see that the COLAV algorithm first slows down as a reaction to the first object, but as the second object enters the region of observation, it has to reverse the ferry along the path in order to avoid a collision. This can also be seen from the overview of Scenario 11 in **Fig. 4.24**, where the ferry has moved backwards from **Fig. 4.24a** to **Fig. 4.24b**. In Scenario 12, the situation is even more critical. As the first object enters the region of observation, the velocity is re-planned from transit velocity of $2m/s$ to hard reverse of $-1m/s$ in order to avoid entering the ROC of the objects.

These situations revile the consequences of the same problem as seen in Scenario 2 in **Section 4.3.2**. If an object has a velocity vector that has a small component orthogonal to the path, it will spend a long time crossing the path and hence take up much space in the $path \times time$ space. This will give the COLAV system a limited set of options, and can, as we have seen, be forced to stop or reverse in order to avoid a collision. The cause for this bottoms out in one of the fundamentals of this approach, namely the single predefined path. It limits the options of the COLAV either moving forward or backwards along the path at a velocity $U \in [-U_{max}, U_{max}]$.

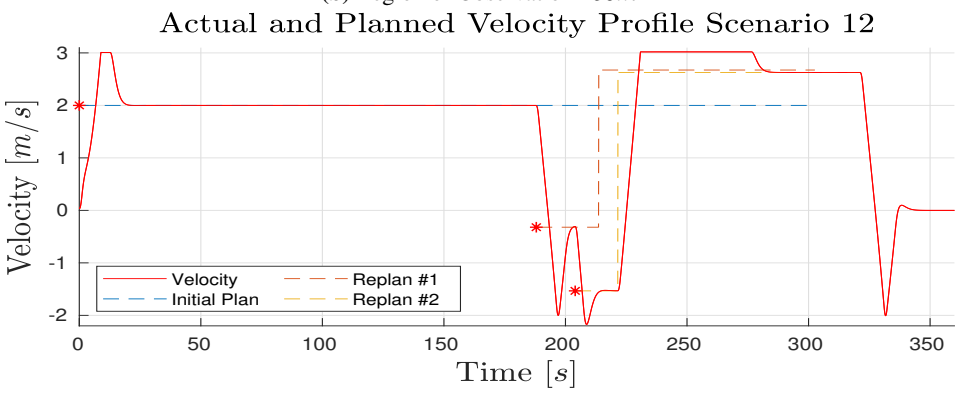
This scenario also highlights another point about this COLAV algorithm and many other algorithms for that matter. The algorithm is only as good as the data available to it. The SP-VP is a deliberate algorithm that plans a global trajectory from the current states to the goal state. If the information is only available out to a certain region of observation, the deliberate algorithm becomes a reactive algorithm with a time horizon defined by the region of observation. In the best case, it has a time horizon of the time it takes the ferry to travel the radius of the region of observation, and in the worst case, the time it takes any fast-moving vessel to travel the distance.



(a) Region of Observation $450m$

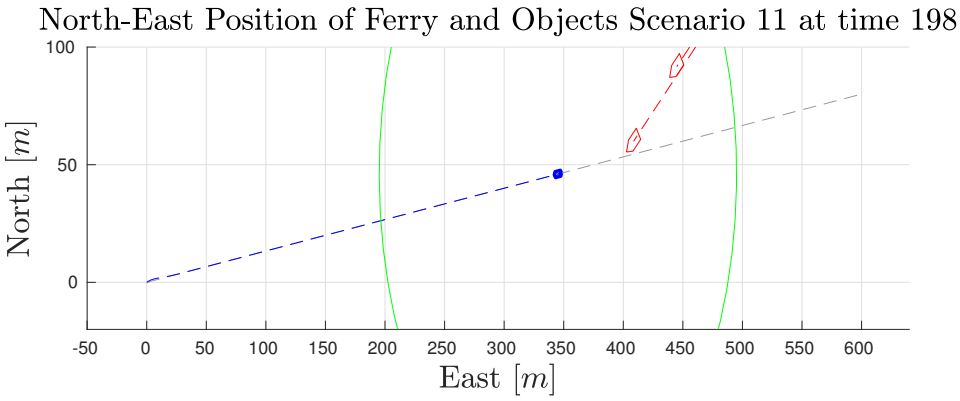


(b) Region of Observation $150m$

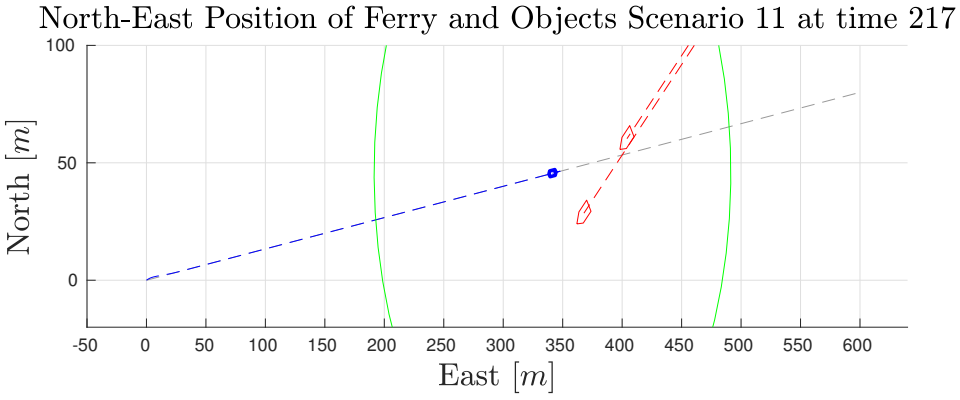


(c) Region of Observation $75m$

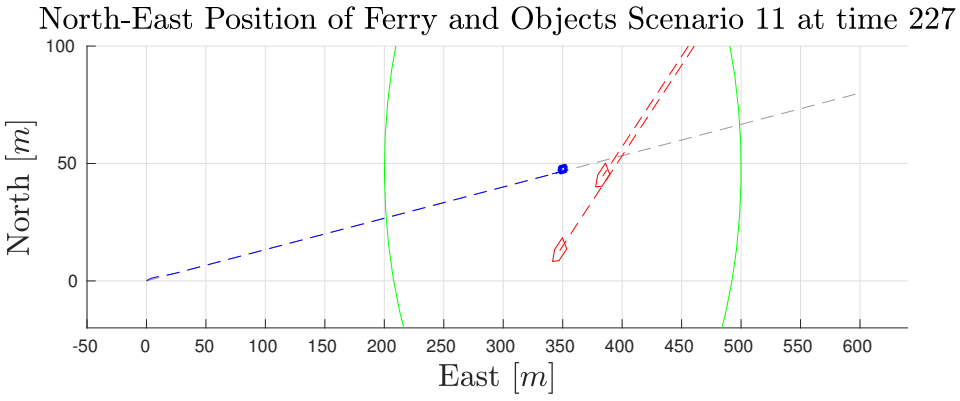
Figure 4.23: Actual and planned velocity profiles for Scenario 10-12.



(a) $t = 192[s]$



(b) $t = 222[s]$



(c) $t = 232[s]$

Figure 4.24: Scenario overview during the transit of Scenario 11. Ferry in blue and moving objects in red. The green circle indicates the region of observation. Note that the ferry is moving backwards on the path form **Fig. 4.24a** to **Fig. 4.24b**.

4.4 Multiple-Path Algorithm

As seen and discussed in the previous section, the SP-VP comes with a set of limitations that are mainly related to the single-path restrictions. Therefore a multiple-path augmentation was made to the system in order to increase the possible actions available to the COLAV algorithm, in the hope of increasing the robustness of the method. The augmentations of the SP-VP to the MP-VP algorithm was presented in **Section 3.3**. In this section, results from the MP-VP are presented and compared to scenarios where the SP-VP had problems. A more thorough evaluation is done in the next section, where the SP-VP and MP-VP algorithms are compared to each other and a VO algorithm.

4.4.1 Scenario 41 and 51 - Slow Down

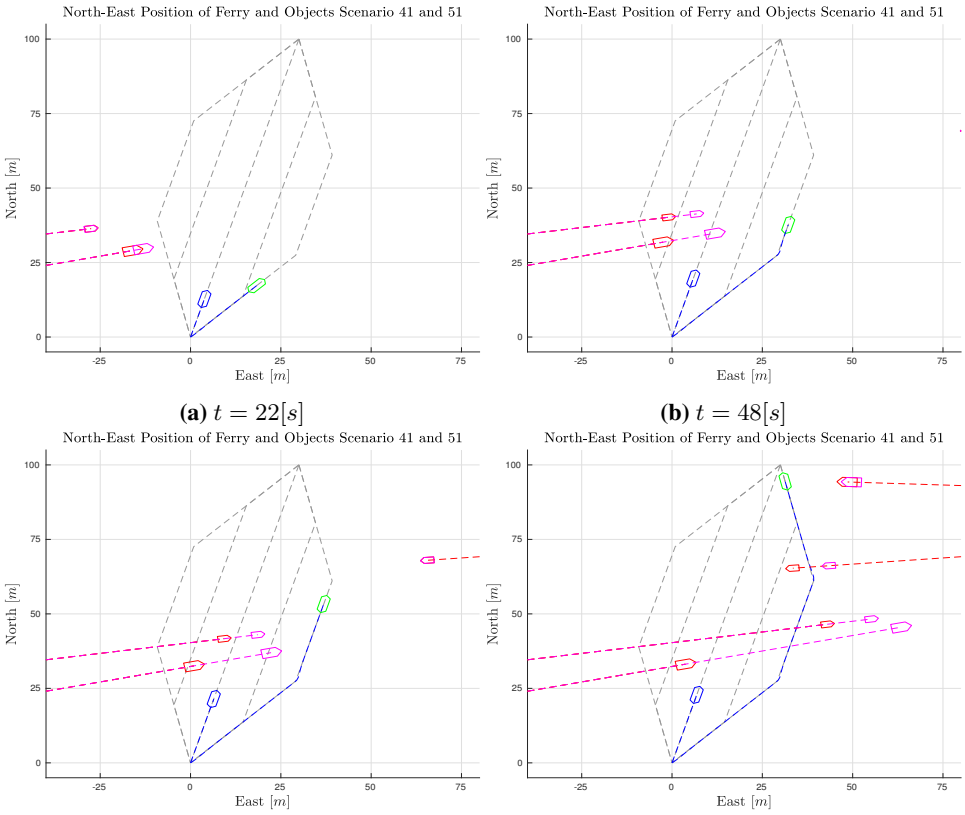
In Scenario 41 and 51, the objects are acting according to Behaviour 2, "Slow Down". The setup is identical in both scenarios, except for the algorithms, where Scenario 41 runs SP-VP, and Scenario 51 runs MP-VP. Snapshots from the scenarios can be seen in **Fig. 4.25**. In the figure, the blue ferry and red objects are from Scenario 41, while the green ferry and pink objects are from Scenario 51. From the snapshots, we see that the MP-VP is able to find a path without encountering a deadlock with any of the objects, in contrast to the SP-VP, that gets stuck in its path waiting for a slow-moving object to pass. The SP-VP eventually manages to clear itself from the deadlock, but this happens long after the MP-VP arrives at its destination.

4.4.2 Scenario 20-22 - Region of Observation

Scenario 20-22 are identical to Scenario 10-12, but the former run the MP-VP, and the latter run the SP-VP. **Fig. 4.27** gives the velocity profiles for the scenarios, and **Fig. 4.26** shows snapshots of the transit from Scenario 21. In the plots, the tracking of the ferry from Scenario 11 is included for comparison. The objects are the same for both scenarios.

By comparing the velocity profiles for Scenarios 20-22 in **Fig. 4.27** with Scenario 10-12 in **Fig. 4.23**, it is apparent that the increased maneuvering options in the MP-VP have removed the need for emergency stopping and reversing. In Scenario 20, the behaviour is identical to Scenario 10. The region of observation of $450m$ gives the algorithm enough time to increase the velocity and pass in front of the objects without branching. In Scenario 21 and 22, the algorithm chooses to branch out as soon as the objects are detected, allowing the ferry to keep a forward velocity throughout the transit.

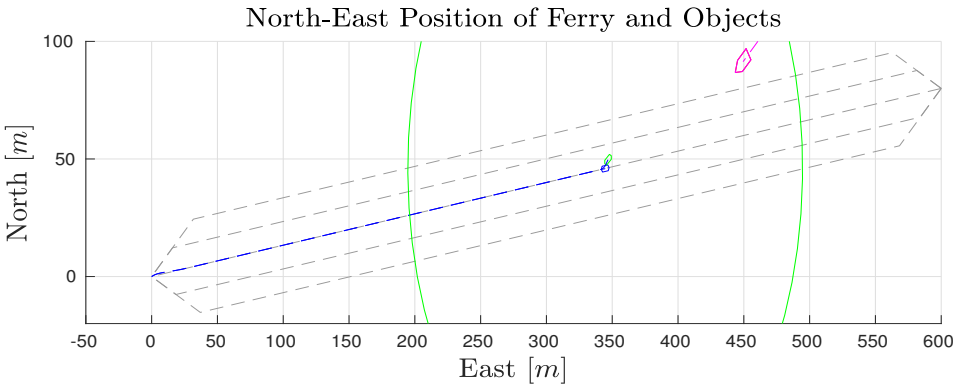
Also with the MP-VP, the short region of observations limits the long-term planning of the deliberate layer, so that the algorithm resembles more a reactive one. The performance is an improvement to the SP-VP, but one could argue that a reactive algorithm with a broader range of possible actions would be better equipped to handle the small region of observation. Then again, the SP-VP and MP-VP are designed to keep within the predefined path(s), which will not be guaranteed with a reactive algorithm like the VO or DW. In addition, SP-VP and MP-VP are designed based on the assumptions that the situational awareness is able to track all objects that will intersect the path in the scope of the transit.



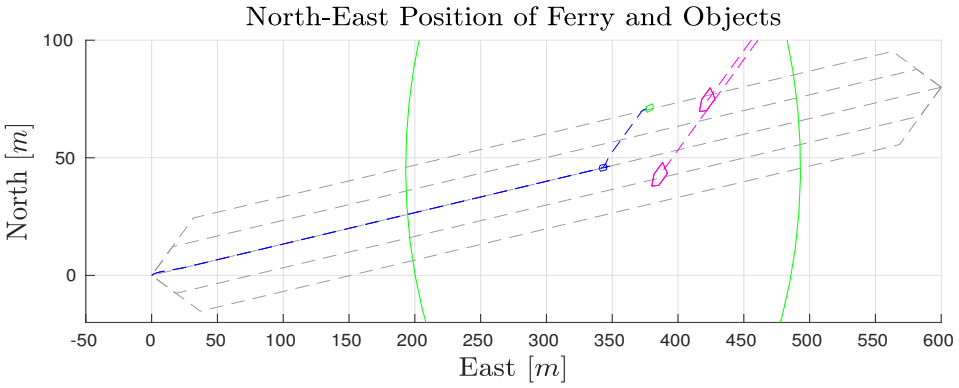
(a) $t = 22[s]$ **(b)** $t = 48[s]$

(c) $t = 66[s]$ **(d)** $t = 101[s]$

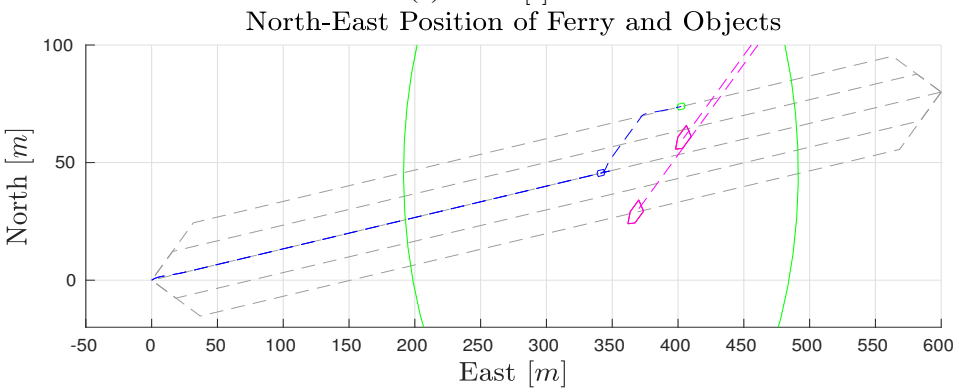
Figure 4.25: Scenario 41 and 51: Scenario overview during the transit. The blue ferry and the red objects belong to Scenario 41, while the green ferry and pink objects belong to Scenario 51.



(a) $t = 178[s]$

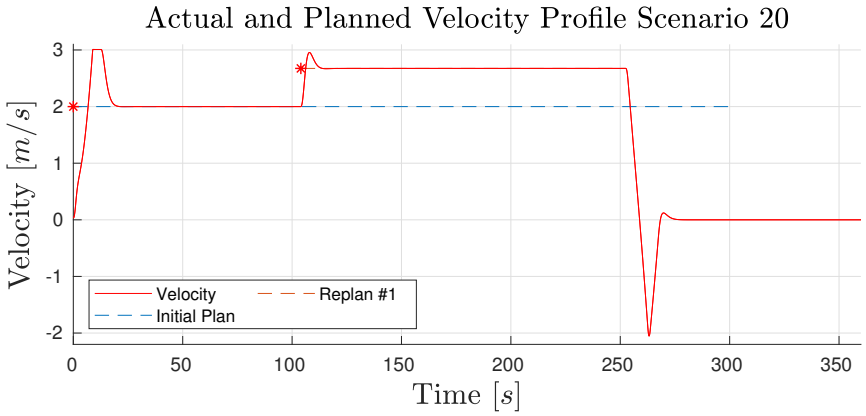


(b) $t = 209[s]$

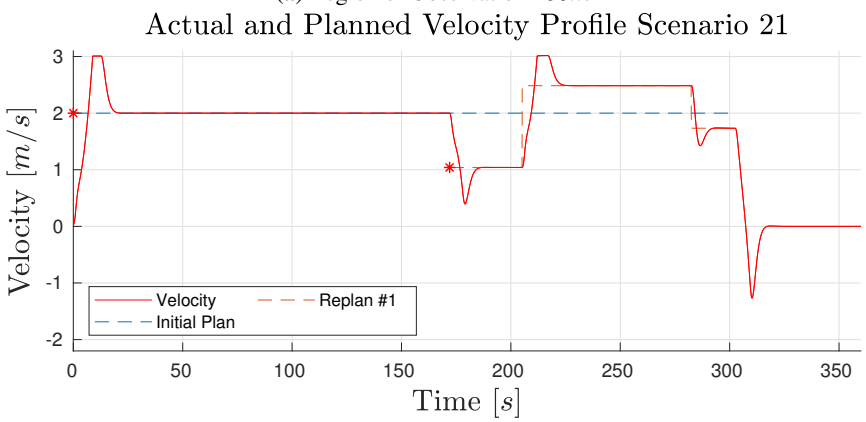


(c) $t = 217[s]$

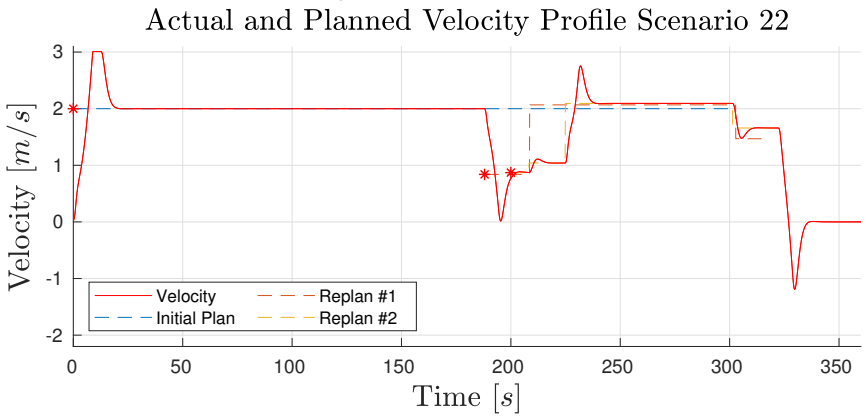
Figure 4.26: Scenario 21: Snapshots from the transit. Ferry from Scenario 21 in green and ferry from Scenario 11 in blue. Objects are the same for both scenarios.



(a) Region of Observation $450m$



(b) Region of Observation $150m$



(c) Region of Observation $75m$

Figure 4.27: Scenario 20-22: Velocity and planned velocity profiles.

4.5 Evaluation and Comparison

In this section, a thorough evaluation of the SP-VP and MP-VP is conducted. For the sake of perspective, the performance is compared to another well-known COLAV algorithm. A version of the velocity obstacle (VO) method was adapted to fit the model and simulator. The method is implemented by Anette Yttisrud in the work done in her project thesis during the spring of 2019. It is based on (Kuwata et al., 2014). The VO used in the comparison does not do any COLREGSs considerations.

The VO uses a LOS guidance law to calculate the desired course

$$\chi_d = \alpha_k + atan2(-e_{ct}/l_{lookahead}, 1) \tag{4.35}$$

where e_{ct} is the cross-track error to the straight-line path from the start-point to the end-point, equal to the path of the SP-VP presented in **Section 3.2**, and

$$\alpha_k = atan2(E_{endpoint} - E, N_{endpoint} - N), \tag{4.36}$$

with N and E as the north and east position of the ferry, respectively. The lookahead distance

$$l_{lookahead} = min(l_{la}, l_{ep}), \tag{4.37}$$

with

$$l_{ep} = \sqrt{(E_{endpoint} - E)^2 + (N_{endpoint} - N)^2}, \tag{4.38}$$

and $l_{la} > 0$ as a constant maximum look-ahead distance. In the simulations l_{la} was set to $25m$. The choice of α_k and $l_{lookahead}$ ensures that the ferry will converge to the endpoint, and not the straight line through the start-point and endpoint.

The scenarios used for comparison of the VO is the same as scenarios 1-5 in **Table 4.3** in terms of path and object initialization. The VO uses the same Dummy Object Detection and reference filter as the SP-VP and MP-VP.

The Scenarios from the VO are presented one by one in the following subsections, where the performance of the VO is commented on and compared to the performance of the respective scenarios for the SP-VP and MP-VP. In the figures where the data from the VO simulations are presented, the tracking of the ferry from the corresponding scenario with SP-VP and MP-VP is included. This is done to aid the comparisons. The objects from the SP-VP and MP-VP scenarios are omitted in order to maintain the readability of the figures. The object data can be found in the section of the respective scenario.

It should be noted that the VO used for this comparison has not been designed for this specific case with confined space and high traffic. It has faults and limitations that there exist solutions to in literature, but have not been implemented, since this is time-consuming, and not within the scope of this thesis.

4.5.1 Performance Metrics

A set of performance metrics have been formulated to aid the comparison of the SP-VP, MP-VP and the VO. The metrics are based on the body acceleration, power usage and

Name	Value
$\dot{u}_{lim}$	$1.1m/s^2$
$\dot{v}_{lim}$	$1.1m/s^2$
$\dot{r}_{lim}$	$0.2rad/s^2$

Table 4.4: Normalizing parameters for the performance metrics.

duration of each transit. The body accelerations metrics are intended to quantify the passenger comfort of the transit. The calculated values for each transit is presented along with the results.

The values for the body acceleration metrics $I_{\dot{u}}$, $I_{\dot{v}}$ and $I_{\dot{r}}$ are calculated according to

$$I_{\dot{u}} = \int_{T_0}^{T_1} \frac{|\dot{u}(t)|}{\dot{u}_{lim}} dt, \tag{4.39}$$

$$I_{\dot{v}} = \int_{T_0}^{T_1} \frac{|\dot{v}(t)|}{\dot{v}_{lim}} dt, \tag{4.40}$$

$$I_{\dot{r}} = \int_{T_0}^{T_1} \frac{|\dot{r}(t)|}{\dot{r}_{lim}} dt, \tag{4.41}$$

where $\dot{u}_{lim}$, $\dot{v}_{lim}$ and $\dot{r}_{lim}$ are normalizing parameters based on limits of what is considered comfortable accelerations for a standing passenger. The values are given in **Tab. 4.4**. The power metric is simply the power usage during the transit

$$I_{power} = \int_{T_0}^{T_1} \boldsymbol{\tau}(t) \cdot \boldsymbol{\nu}(t) dt, \tag{4.42}$$

where $\boldsymbol{\tau}$ is the 3 DOF control input and $\boldsymbol{\nu}$ is the body velocity vector of the ferry. The integration limits T_0 and T_1 are the start-time and end-time of the transit. The value for the time-metric is the duration of the transit in seconds,

$$I_{duration} = T_1 - T_0, \tag{4.43}$$

from the ferry starts moving, to the ferry enters a circle of radius $0.5m$ with the centre in the destination.

4.5.2 VO-Scenario 1

This scenario has constant object behaviour, and therefore concur with the assumptions made in both VO and the velocity-planning methods. **Fig. 4.28** contains snapshots from the transit. Initially, the VO speeds off along the path but diverges to starboard as the two vessels are approaching from the port side. The divergent maneuver becomes extra long as the first vessel from the starboard side blocks the possibility of moving back onto the path. After the object from starboard has passed, the VO moves back to the path and eventually to the destination. The lack of global perspective in the reactive VO algorithm becomes apparent when it chooses a diverging maneuver that creates a situation that calls for a new

diverging maneuver with another object. The unlucky maneuvers result in a transit that is more than a minute longer than the velocity-planning methods, as can be seen in **Table 4.5**.

COLAV Algorithm	$I_{\dot{u}}$	$I_{\dot{v}}$	$I_{\dot{r}}$	$I_{power}(kJ)$	$I_{duration}(s)$
SP-VP	**16.9**	**5.07**	**9.04**	**12.0**	128.0
MP-VP	46.4	30.1	17.8	25.6	**125.0**
VO	58.1	63.0	26.8	26.2	199.0

Table 4.5: VO-Scenario 1: Values for the metrics of VO-Scenario 1 and the corresponding scenarios for SP-VP and MP-VP. The best value in each column is highlighted in bold.

The VO performs the transit without collision, but with considerably higher maneuvering efforts than SP-VP, and overall higher than MP-VP, as can be seen from the values in **Table 4.5**. The SP-VP score overall best, and has by far the lower energy consumption. This is due to the straight-line path, which results in the heading and course reference being aligned through the transit, giving it less drag from the hull. For the same reason, the $I_{\dot{v}}$ is lower for the SP-VP than the two other methods. The SP-VP has the lowest $I_{\dot{v}}$ in all five comparison scenarios, and the reason the value is not closer to zero is that the vessel is unstable in yaw, and induces oscillations in all three degrees of freedom.

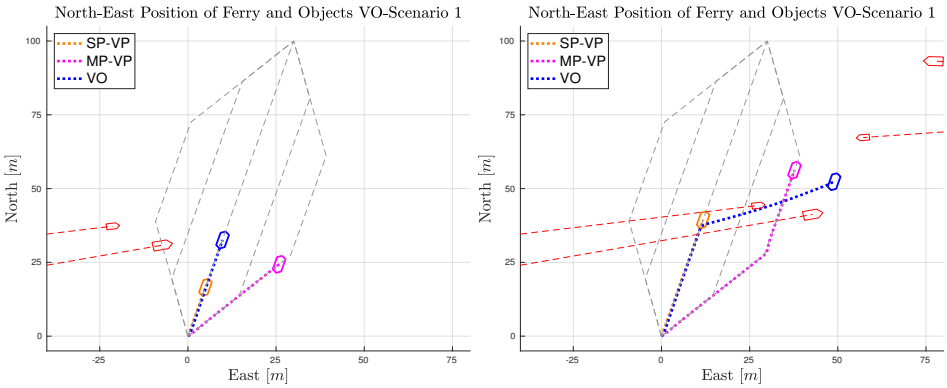
The fixed path(s) of the VP methods ensures that the ferry moves toward the goal, whereas the course of the maneuvers VO performs has a high angle relative to the centre-path, which can induce uncertainty and confusion both to passenger and operators of the objects as to the intentions of the ferry.

4.5.3 VO-Scenario 2

In this scenario, the objects slow down as they approach the ferry. Snapshots from the transit is displayed in **Fig. 4.30**. Also in this scenario the VO finds a collision-free transit. The MP-VP finds the fastest and smoothest transit in terms of velocity profile. This is because it is able to branch out, and therefore avoid the first two objects approaching from the port side. The branching transit is planned in a way that avoids a situation with the two objects approaching from starboard as well. The VO and SP-VP are unable to avoid a situation with the first two objects, where the VO again diverges to the starboard in a way that causes it to halt and stop for the first object from starboard. The SP-VP slows down for the first two objects, but gets stuck on the last objects, as have been seen earlier in **Section 4.3.2**. The SP-VP did therefore not complete the transit before the simulations terminated, but the results will be considered nevertheless. Values for the metrics are given in **Table**

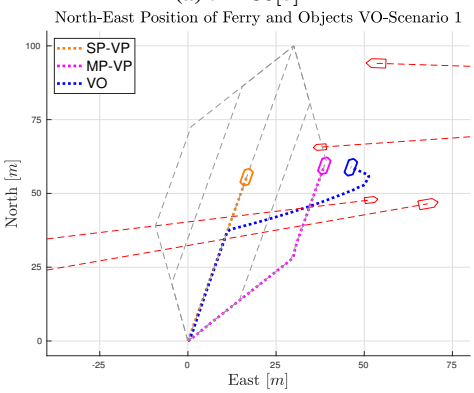
COLAV Algorithm	$I_{\dot{u}}$	$I_{\dot{v}}$	$I_{\dot{r}}$	$I_{power}(kJ)$	$I_{duration}(s)$
SP-VP	26.8	**6.65**	**13.3**	**6.23**	> 200.0
MP-VP	**24.1**	27.2	15.7	21.8	**104.0**
VO	74.9	95.4	36.6	17.7	268.0

Table 4.6: VO-Scenario 2: Values for the metrics of VO-Scenario 2 and the corresponding scenarios for SP-VP and MP-VP. The best value in each column is highlighted in bold.

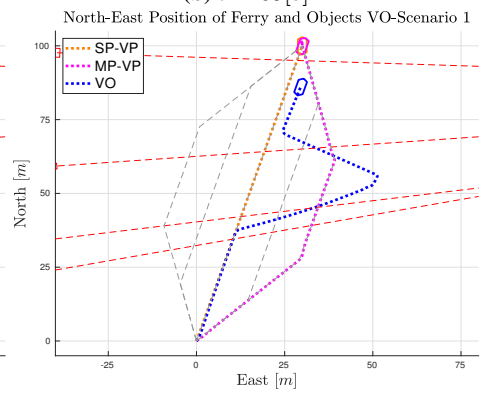


(a) $t = 30[s]$

(b) $t = 69[s]$



(c) $t = 89[s]$



(d) $t = 168[s]$

Figure 4.28: VO-Scenario1: Snapshots from the transit of VO-Scenario1. The ferry in blue, moving objects in red, the ferry from Scenario 1 in orange and the ferry from Scenario 50 in pink. The paths from the multiple-path COLAV has nothing to do with the VO, but are included for comparison.

4.6, where the best value for each metric is highlighted in bold. Once again, SP-VP comes out best on comfort, maneuvering and power, while VO has the worst performance, with high rate-changes compared to the two others. The cause of this can be seen in the velocity reference for the VO in **Fig. 4.31** in combination with plot of the performance metrics as a function of time, given in **Fig. 4.6**. Between $50s$ and $100s$ into the transit, the velocity reference shows oscillatory behaviour, that propagates to the vessel velocity. In **Fig. 4.6** it is clear that the values for VO have most of the growth in this region. Another situation occurs between $120s$ and $170s$ into the transit, the VO gets stuck on first object approaching from starboard. This is caused by the algorithm alternating between trying to pass in front and behind the vessel. The reference filter smooths out this behaviour, and the ferry comes to a near halt. This is a common problem for the VO algorithm, and can be solved adjusting the cost function to penalize changes in velocity and/or heading, but will, as a consequence, reduce the response of the system.

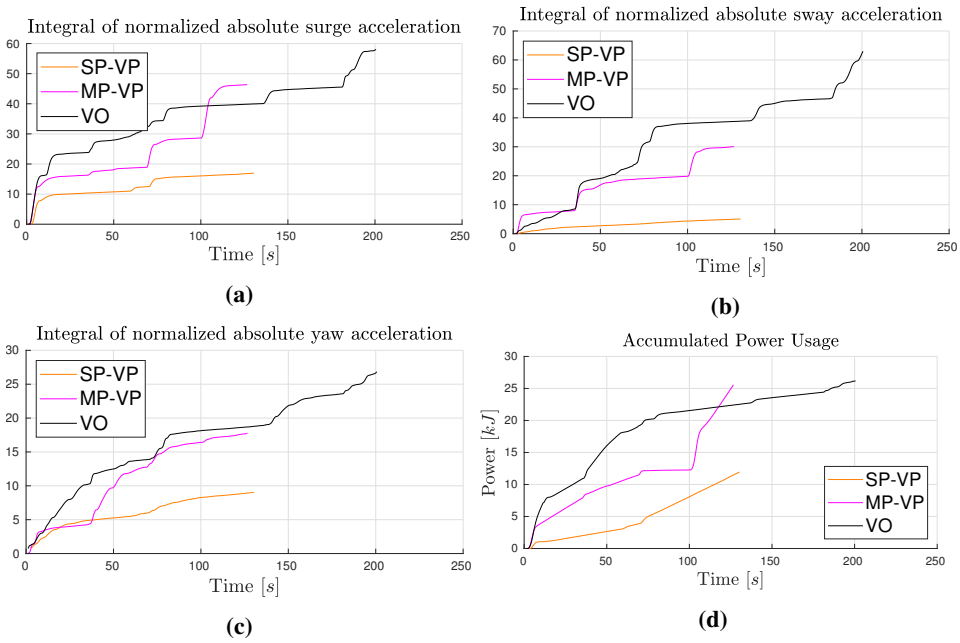


Figure 4.29: VO-Scenario 1: Growth of metrics during the transit. **(a)**: body-x acceleration, **(b)**: body-y acceleration, **(c)**: yaw acceleration, **(d)**: power.

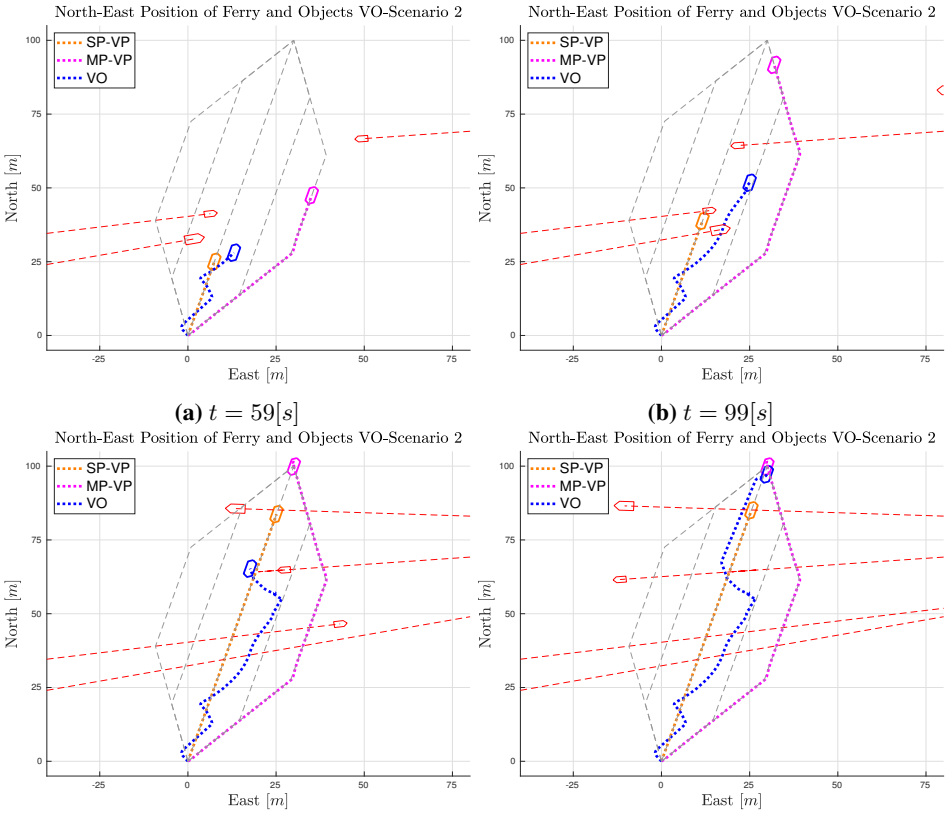
4.5.4 VO-Scenario 3

In this scenario, the objects speed up and try to pass in front of the ferry. Snapshots from the transit are shown in **Fig. 4.33**. This comparison highlights one of the main advantages of the SP-VP and MP-VP, namely that they will not diverge from the predefined paths and therefore not risk to pass through the area that is not guaranteed to be free of static objects. It also shows a weakness that many of COLAV algorithms have, where the vessel can get ”caught” by a passing vessel because the COLAV algorithm keeps diverging from the path in order to avoid a collision.

Once again the SP-VP comes out on top according in the metrics in **Table 4.7**.

COLAV Algorithm	$I_{\dot{u}}$	$I_{\dot{v}}$	$I_{\dot{r}}$	$I_{power}(kJ)$	$I_{duration}(s)$
SP-VP	**59.6**	**12.0**	**24.3**	**16.6**	**145.0**
MP-VP	101.0	52.9	27.7	36.4	164.0
VO	117.0	118.0	45.6	40.6	226.0

Table 4.7: VO-Scenario 3: Values for the metrics of VO-Scenario 3 and the corresponding scenarios for SP-VP and MP-VP. The best value in each column is highlighted in bold.



(a) $t = 59[s]$ **(b)** $t = 99[s]$

(c) $t = 198[s]$ **(d)** $t = 267[s]$

Figure 4.30: VO-Scenario 2: Snapshots from the transit of VO-Scenario 2. The ferry in blue, moving objects in red, the ferry from Scenario 2 in orange and the ferry from Scenario 51 in pink. The paths from the multiple-path COLAV has nothing to do with the VO, but are included for comparison.

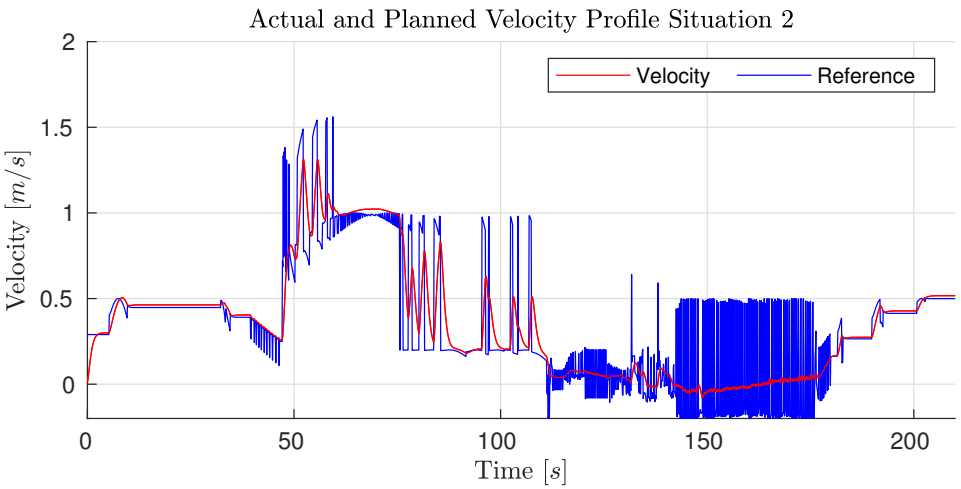


Figure 4.31: VO-Scenario 2: Velocity and velocity reference.

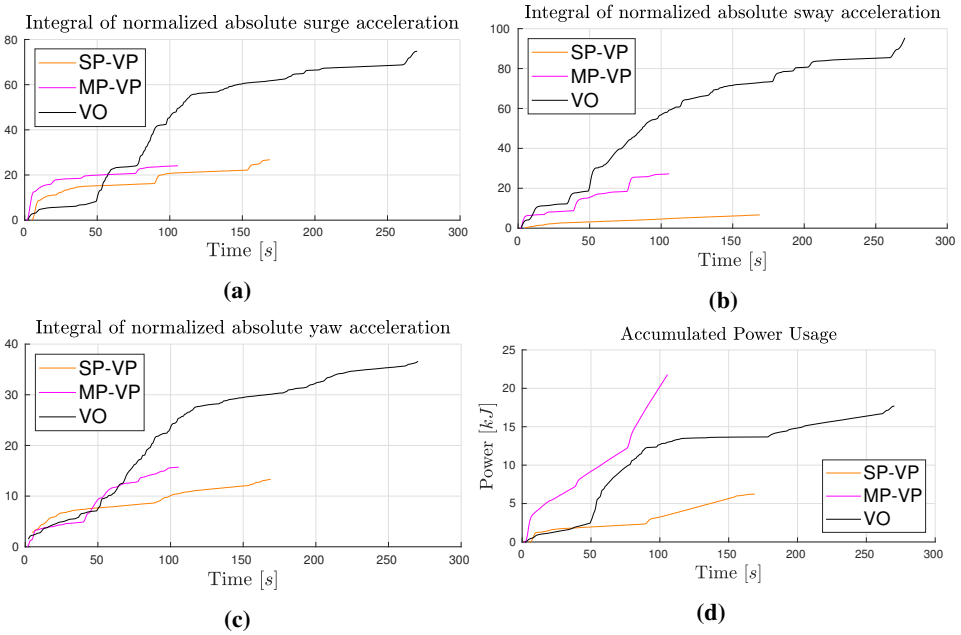
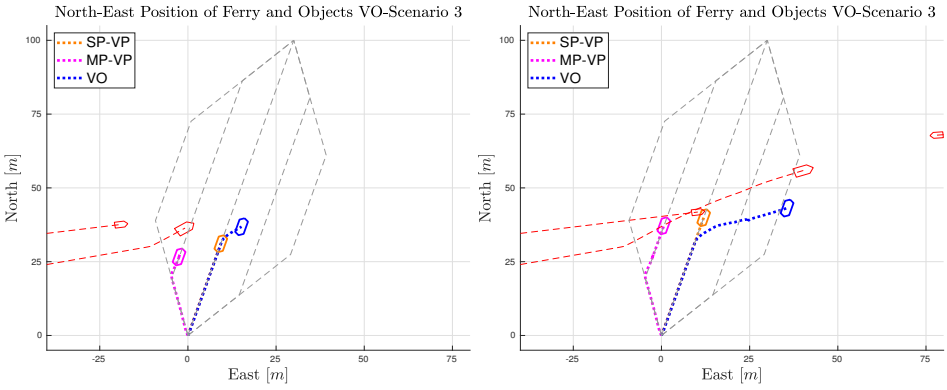
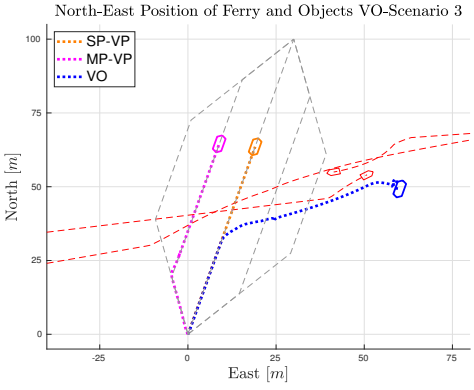


Figure 4.32: VO-Scenario 2: Growth of metrics during the transit. **(a)**: body-x acceleration, **(b)**: body-y acceleration, **(c)**: yaw acceleration, **(d)**: power.

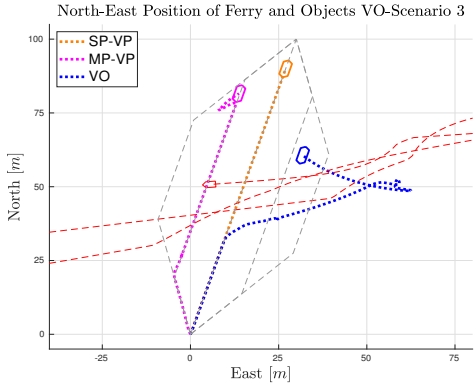


(a) $t = 39[s]$

(b) $t = 69[s]$



(c) $t = 109[s]$



(d) $t = 148[s]$

Figure 4.33: VO-Scenario 3: Snapshots from the transit of VO-Scenario 3. The ferry in blue, moving objects in red, the ferry from Scenario 3 in orange and the ferry from Scenario 52 in pink. The paths from the multiple-path COLAV has nothing to do with the VO, but are included for comparison.

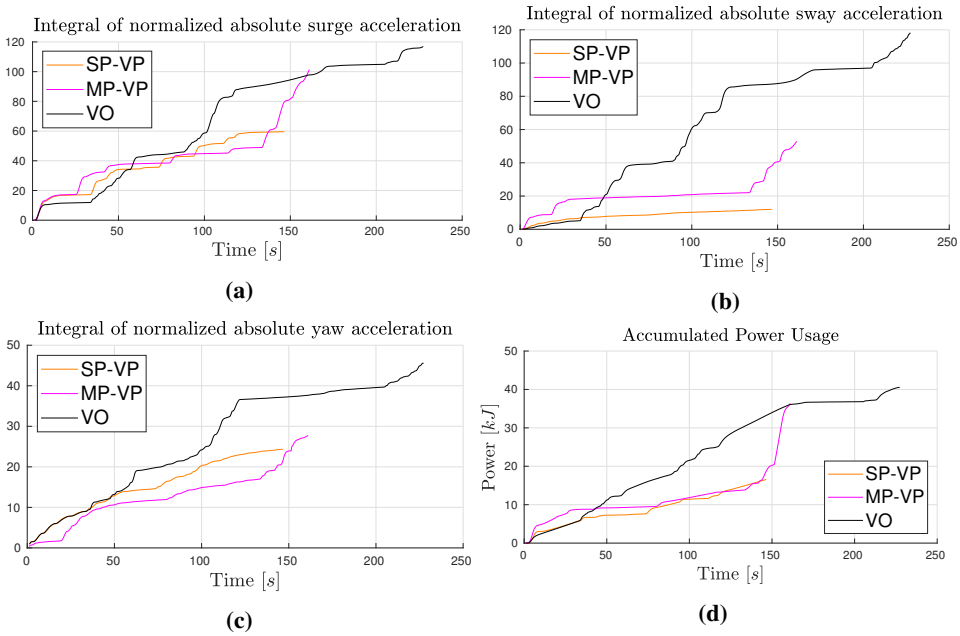


Figure 4.34: VO-Scenario 3: Growth of metrics during the transit. **(a)**: body-x acceleration, **(b)**: body-y acceleration, **(c)**: yaw acceleration, **(d)**: power.

4.5.5 VO-Scenario 4

In this scenario, the objects try to pass behind the ferry. **Fig. 4.35** contains snapshots from the transit. The VO find, once again, a collision-free transit, but encounters a situation with the first object from starboard, when it alters its course to pass behind the ferry. The VO has to reverse the ferry in order to avoid a collision. Both the SP-VP and the MP-VP have shown similar behaviour when objects suddenly alter the course towards the ferry. This is, of course, an extreme situation, where the behaviour of the moving objects is unpredictable and dangerous and is at the limit of what can be expected from a COLAV system. An increased safety-factor, i.e. by increasing the radius of the object representation, could prevent this kind of emergency maneuvers both for the VO, SP-VP and MP-VP.

The time development of the metrics is given in **Fig. 4.36**, and the final values are shown in **Table 4.8**. VO gets a poor score, much due to the unfortunate near-collision, but SP-VP and MP-VP produce a noteworthy result. MP-VP branches out initially, and branches back in as the first two objects are passed, but as it has branched back to the path, it ends up behind SP-VP. This is because the assumptions on object behaviour did not hold, and what initially seemed to be the better option (to branch), turned out not to be. The branching comes at a cost both in terms of energy consumption and passenger comfort, and since the assumptions are faulty, the potential gain of branching should be considerable, for it to be the favourable choice. This can be improved by increasing the cost of branching until the desired adverseness is achieved.

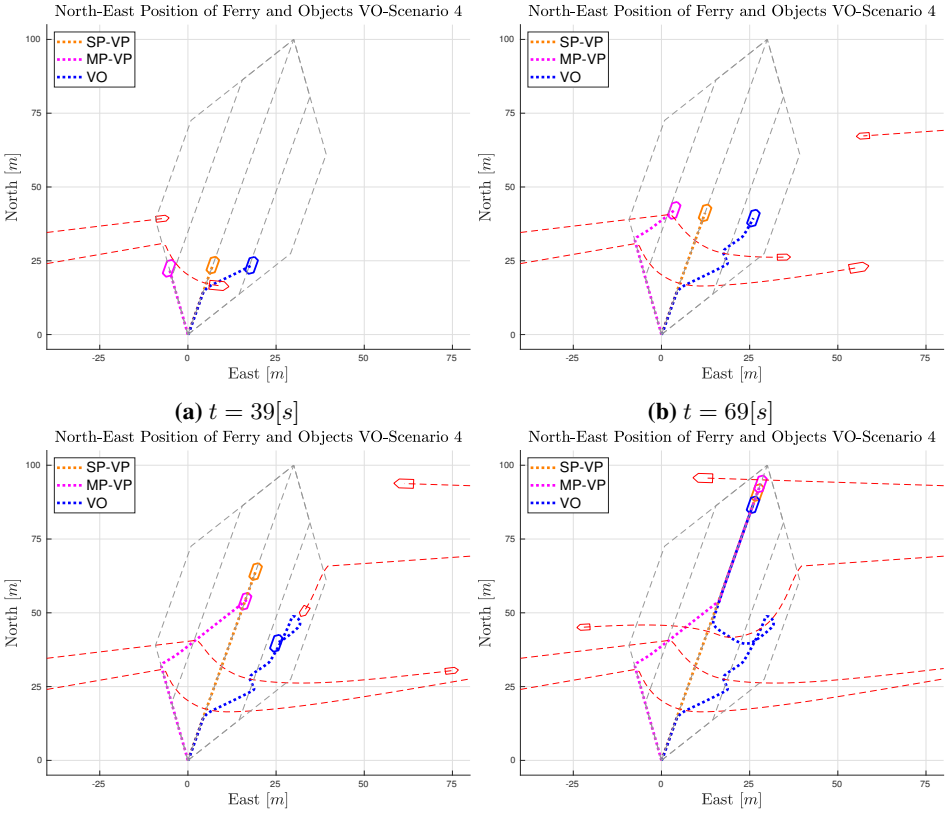
COLAV Algorithm	$I_{\dot{u}}$	$I_{\dot{v}}$	$I_{\dot{r}}$	$I_{power}(kJ)$	$I_{duration}(s)$
SP-VP	**23.6**	**6.17**	**10.3**	**10.3**	148.0
MP-VP	53.1	41.6	18.4	20.0	**143.0**
VO	75.6	96.5	42.0	35.9	160.0

Table 4.8: VO-Scenario 4: Values for the metrics of VO-Scenario 4 and the corresponding scenarios for SP-VP and MP-VP. The best value in each column is highlighted in bold.

4.5.6 VO-Scenario 5

In this scenario, the objects try to follow the COLREGs. Snapshots from the transit can be seen in **Fig. 4.37**. In this MP-VP branches out to pass behind the first two vessels approaching from the port side, and manages to pass the two objects from the starboard, before they approach the path, giving it a problem-free, short duration, as can be seen from the values of **Table 4.9**, where it has the by far best $I_{\dot{u}}$ value, and overall best score. The SP-VP, on the other hand, ends up in close quarters with the final two objects and has to perform emergency braking, as have been seen from the velocity profile in **Fig. 4.21** in **Section 4.3.5**.

In this scenario one can once again see how the VO performs a transit at considerable higher maneuvering efforts than both the SP-VP and MP-VP. It is highly reactive to changes in object behaviour, but this comes at the cost of both passenger comfort. The two VP-algorithms performs the transit at shorter time, with less maneuvering, as can be seen by **Table 4.9**.



(a) $t = 39[s]$ **(b)** $t = 69[s]$

(c) $t = 99[s]$ **(d)** $t = 148[s]$

Figure 4.35: VO-Scenario 4: Snapshots from the transit of VO-Scenario 4. The ferry in blue, moving objects in red, the ferry from Scenario 4 in orange and the ferry from Scenario 53 in pink. The paths from the multiple-path COLAV has nothing to do with the VO, but are included for comparison.

COLAV Algorithm	$I_{\dot{u}}$	$I_{\dot{v}}$	$I_{\dot{r}}$	$I_{power}(kJ)$	$I_{duration}(s)$
SP-VP	61.6	**9.98**	21.0	**15.8**	156.0
MP-VP	**22.3**	23.9	**16.2**	19.0	**114.0**
VO	80.3	83.9	30.6	27.6	176.0

Table 4.9: VO-Scenario 5: Values for the metrics of VO-Scenario 5 and the corresponding scenarios for SP-VP and MP-VP. The best value in each column is highlighted in bold.

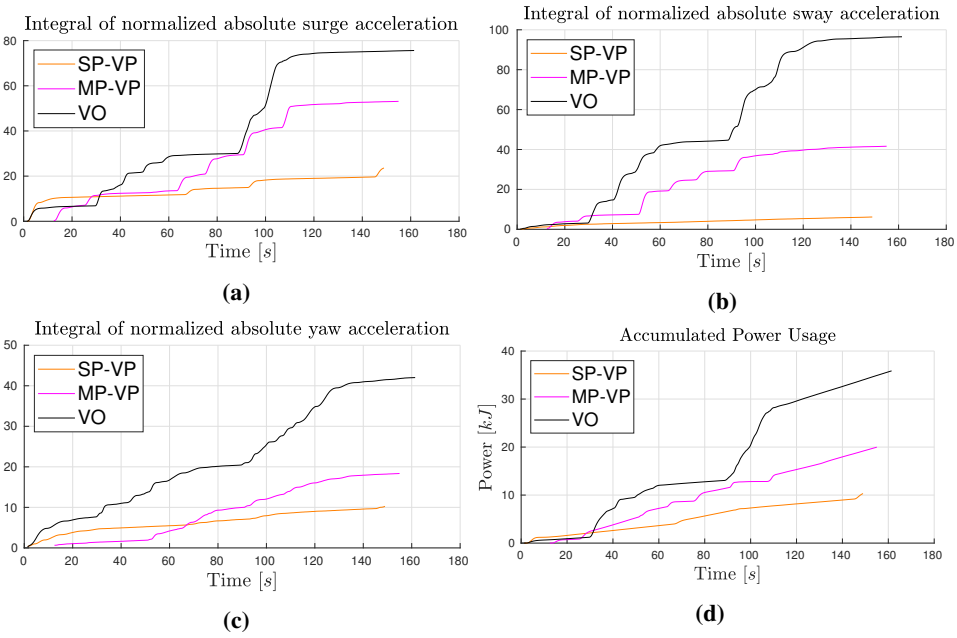
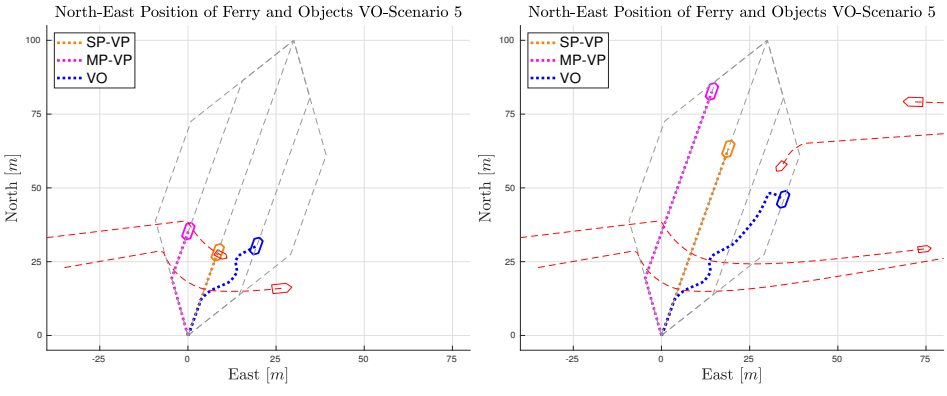
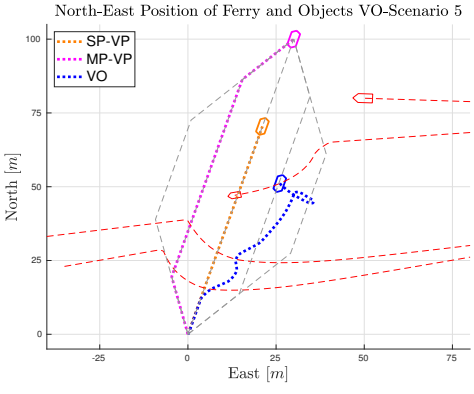


Figure 4.36: VO-Scenario 4: Growth of metrics during the transit. **(a)**: body-x acceleration, **(b)**: body-y acceleration, **(c)**: yaw acceleration, **(d)**: power.

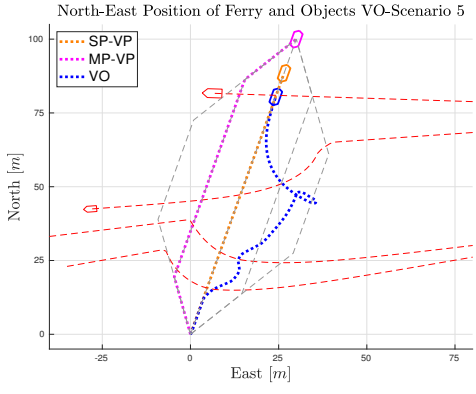


(a) $t = 49[s]$

(b) $t = 99[s]$



(c) $t = 119[s]$



(d) $t = 158[s]$

Figure 4.37: VO-Scenario 5: Snapshots from the transit of VO-Scenario 5. The ferry in blue, moving objects in red, the ferry from Scenario 5 in orange and the ferry from Scenario 54 in pink. The paths from the multiple-path COLAV has nothing to do with the VO, but are included for comparison.

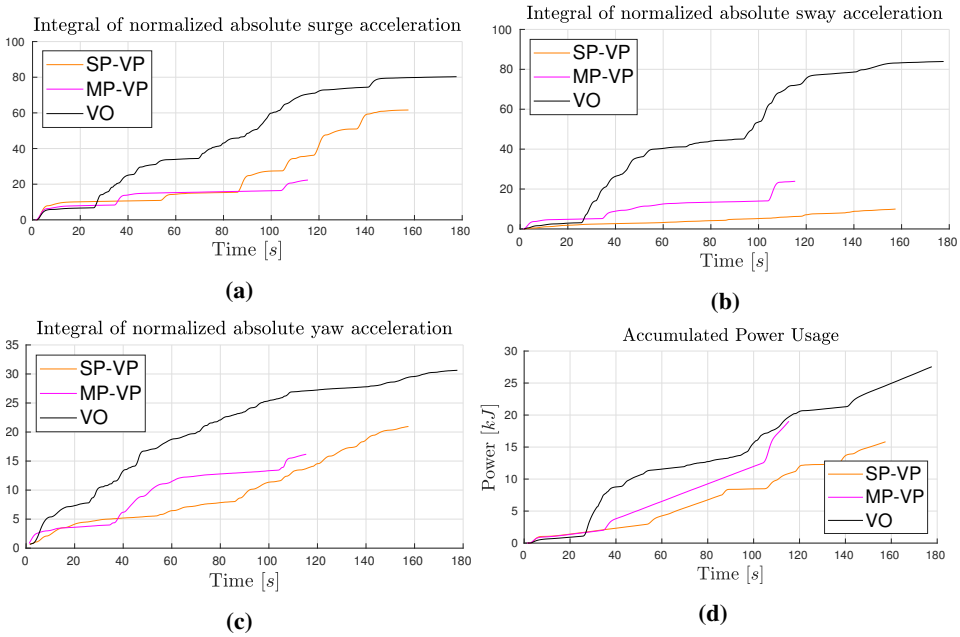


Figure 4.38: VO-Scenario 5: Growth of metrics during the transit. **(a)**: body-x acceleration, **(b)**: body-y acceleration, **(c)**: yaw acceleration, **(d)**: power.

4.6 Discussion

In a scenario with four moving objects and constant object behaviour, both SP-VP and MP-VP are able to find a collision-free trajectory with ease. The trajectories give a predictable behaviour with low maneuvering effort compared to a VO-based system.

The improved object representation increases the robustness to noise in the object estimations. Nevertheless, the trajectory planning is only as good as the available object information. If the object state estimations are faulty, the planned trajectory will be as well.

The SP-VP, due to the restrictions of the predefined path, runs the risk of stopping for considerable periods when objects have a small velocity-component orthogonal to the path. This is a safe, intuitive and energy-efficient method of handling the situation, but it comes at the cost of transit time.

The cost of branching in terms of passenger comfort and increased energy consumption should be better reflected in the cost of choosing a branching path in the node search problem. The assumptions on object behaviour are weak, and therefore, the potential gain of branching should be considerable before it is pursued.

The fixed branching angle of the MP-VP gives the method the same weakness as the SP-VP where it can get stuck if a slow-moving object is too close and therefore blocks the branching possibilities. This can be improved by a dynamic branch angle but will, in turn, introduce higher complexity in the algorithm and higher maneuvering efforts.

The branching option of MP-VP reduces the need for long halts, emergency maneuvers and backing up the ferry, compared to the SP-VP. If the branching option is made available only in critical situations, the MP-VP can combine the comfortable and predictable characteristics of the SP-VP with the increased response and safety introduced by the branching.

When the situational awareness of the algorithm is restricted, the global perspective of the deliberate layer is reduced, and the algorithm, in practice, becomes a reactive algorithm with a time-horizon defined by the perception-range. In cases with short range, the SP-VP renders useless, since the whole concept of adapting velocity collapses. The MP-VP is somewhat more robust to this, due to its increased range of actions, yet it is still very restricted. In such cases, a reactive algorithm like the VO would be a better choice.

Chapter 5

Experimental Results

In this chapter, the results from the sea trials of the COLAV system are presented and discussed. In the first two subsections, the experimental platform and the testing environments are described. In the following subsections, the experimental data is visualized. The data is discussed as it is presented, and a summary of the discussion is included in the last section of the chapter.

5.1 Experimental Platform

The platform that is used to perform the experiments is the milliAmpere ferry. A system overview of the ferry can be seen in **Fig. 5.1**. The guidance and control system was introduced in the simulator overview in **Section 4.1.1**, and will therefore not be addressed here. The vessel and navigation system, on the other hand, is different from the simulations, where a vessel and thruster model was used to simulate the system.

The ferry has an overall length of $5m$, a beam of $2.8m$ and weighs approximately $1670kg$. The vessel has a flat-bottomed hull with no keel. A side-view and a front-view of the ferry can be seen in **Fig. 5.2**.

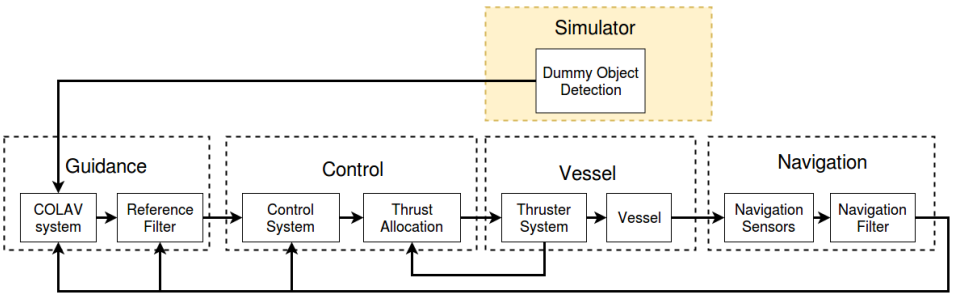
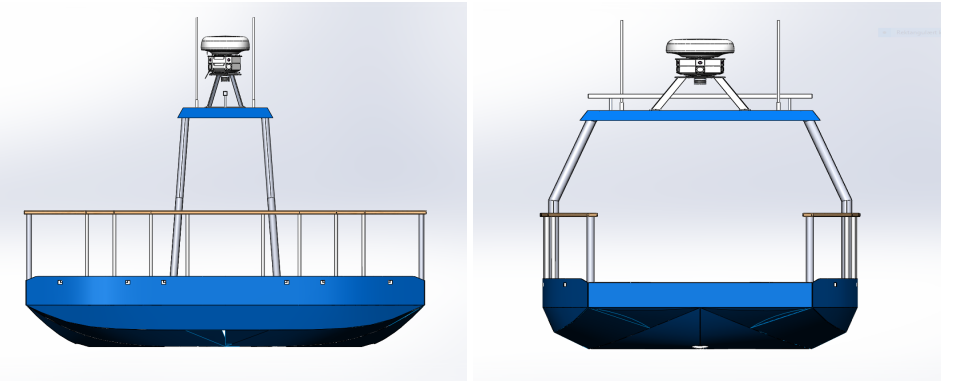


Figure 5.1: Overview of the ferry and GNC on the experimental platform.



(a) **(b)**

Figure 5.2: CAD drawings for the hull, roof and sensor jig, courtesy of Glenn Angell. **(a)** Side-view of the milliAmpere platform. **(b)** Front view of the milliAmpere platform.

Navigation

The base of the navigation system is IMU sensors and a GPS. The IMU is an Xsens MTi-20, which outputs the linear acceleration and rotational velocities of the vessel. The GPS is a Vector™ VS330 GNSS Receiver, set up with an antenna separation of $2m$. The GPS has RTK capacity, providing a position accuracy of $10mm$ in the horizontal plane, $20mm$ in the vertical dimension and $0.05^o deg$ in heading. The navigation node uses an Error State Kalman Filter for sensor fusion of the GPS and IMU data. The filter is described in detail in (Sæther, 2019).

On Board Computer

The computing power on the vessel comes from the onboard computer (OBC), an Axiomtek eBOX670-883-FL with an Intel Core I7 processor. The OBC runs Ubuntu OS and the Kinetic ROS distribution.

5.2 Testing Environments

The sea trials are performed in the harbour basin at Pirkaia in Trondheim city. There are few vessels located in the area, and the traffic is at a minimum, the basin does also have a good shielding from currents and waves. **Fig. 5.3** shows a satellite picture of the location. The picture has a north-up orientation. A local NED frame with origo in the red dot, located on top of the blue-green vessel, is used. All the experimental data is given in the local NED frame. The red line illustrates the path that is used during testing. In the case of the MP-VP algorithm, the red line is the centerpath. The path goes between the north-west destination at $[N, E] = [26, -79]$ and the south-east destination at $[N, E] = [-41, -11]$ in the local $\{n\}$. All experiments are performed with virtual objects

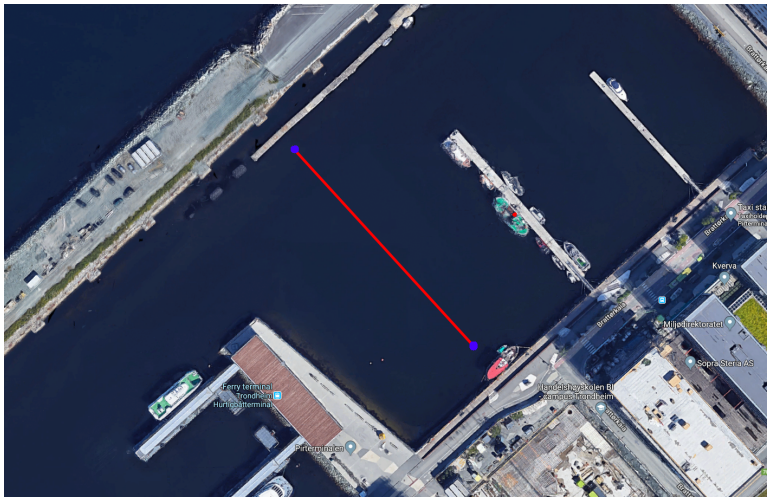


Figure 5.3: Location for the sea trials, a harbour basin located in Pirkaia. The red line illustrates the path that is used. The red dot on top of the blue-green vessel marks the origin of the local NED frame. Courtesy of Google Maps.

from the object detection module introduced in **Section 4.1.5**. None of the features from the testing-area are considered, so the objects can move freely in the local NED frame.

5.3 Experimental Results and Discussion

In this section, the experimental results are presented. Firstly, a description and overview of the experiments are given, followed by a visualization of the results. Only a subset of the transits performed during testing is presented and commented on in this chapter. The results for the rest of the transits are included in **Appendix B**.

5.3.1 Overview

A total of nine transits were performed. The transits are named **Transit n**, where the number n is given from the chronological order they were performed. **Table 5.1** gives an overview of the transits, with the COLAV algorithm and object behaviour. All transits go between the two positions marked in **Fig. 5.3**, the odd-numbered transits travel from the south-east point to the north-west point, while the even number transits travel the opposite direction. Transit 1-7 are run with four moving objects, approaching at an angle close to 90^o to the path. Transit 8-9 are run with two objects at a steeper angle to the path.

5.3.2 Transit 1

This is the simplest of all the transits, in terms of collision avoidance. The constant object behaviour concur with the assumptions in the algorithm, giving a predictable scenario with

Figure 5.4: The milliAmpere ferry from at sea trials the 2 June 2019 . Nice conditions with calm water and only the occasional light breeze.

Situation	Section	COLAV algorithm	Behaviour
Transit 1	5.3.2	SP-VP	1
Transit 2	5.3.3	SP-VP	2
Transit 3	B.1	SP-VP	3
Transit 4	5.3.4	MP-VP	2
Transit 5	B.2	MP-VP	3
Transit 6	B.3	SP-VP	2
Transit 7	5.3.5	MP-VP	1
Transit 8	B.4	SP-VP	1
Transit 9	B.5	MP-VP	1

Table 5.1: Overview of the nine transits performed during the experiments. Some of them are included in **Chapter. 5**, the rest can be found in **Appendix B**.

minor changes. **Fig. 5.6** shows four snapshots of the situation during the transit, and **Fig. 5.7** shows the velocity and velocity reference during the transit.

The transit is performed without any critical situations, it starts off slowly, to pass behind the first object approaching from the right and proceeds in transit velocity for the rest of the transit. As a passenger, the transit was fairly comfortable, but as can be seen from the velocity profile, the ferry has some oscillatory behaviours. The actual velocity of the ferry oscillates around the velocity reference throughout the transit. The cause for this is not the COLAV system, but a mismatch between the ferry model and the ferry. It mainly originates from the thruster system, where the delay in the thrusters has been underrated in the modeling, both in azimuth angle turn rate, and in the ramp-up of the propellers. In addition, an unmodeled delay of up to $0.4s$ in the communication between the thrust allocation and thrusters was discovered by Anders Pedersen in his work on determining model parameters. The consequence is that the control system is unable to follow the acceleration reference from the reference filter, and the filter ”runs away” from the ferry,

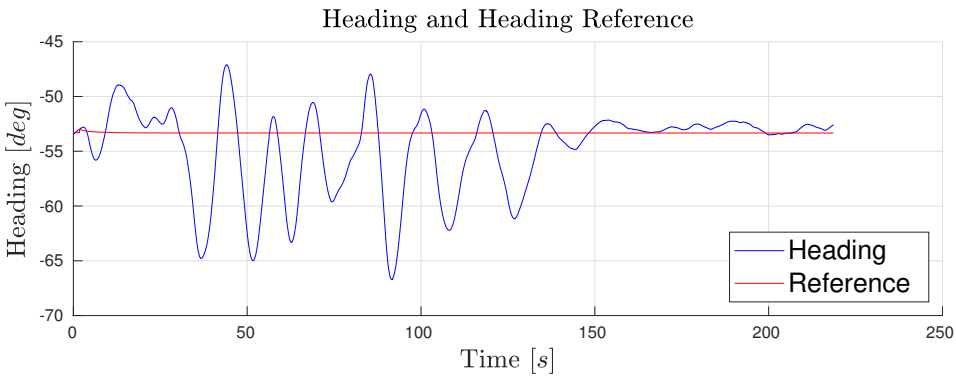


Figure 5.5: Transit 1: Heading and heading reference.

which results in the control system having to catch up with the reference. A solution to this could be to, either improve the thruster model, include logic in the reference filter to reset the filter states to the current states when the tracking error of the filter is above a certain limit or improve the thruster-assumptions in the reference filter to better match the actual thruster model. Ideally, all three solutions should be realized.

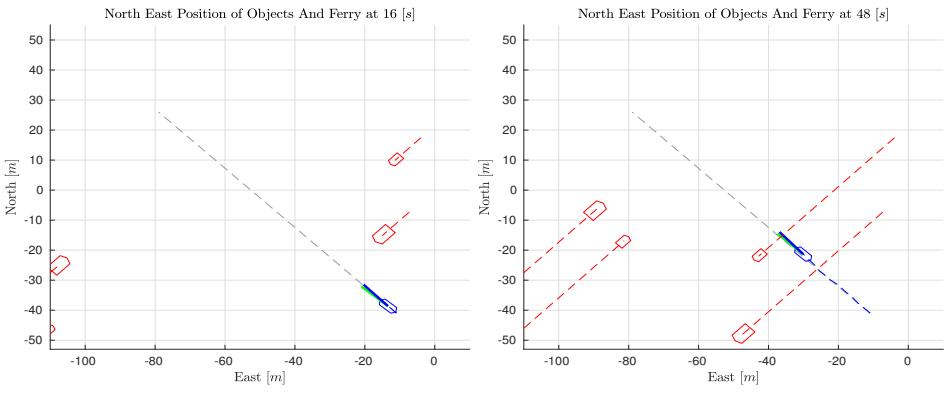
Another contribution to the problem is that the ferry is unstable in yaw. This causes it to diverge from the heading reference at the initial acceleration, as can be seen from the heading and heading reference in **Fig. 5.5**. This, in turn, calls for a yaw moment, and hence, a rotation of the azimuth thrusters which reduces the available thrust in the surge direction and thereby further increases the absolute tracking error.

5.3.3 Transit 2

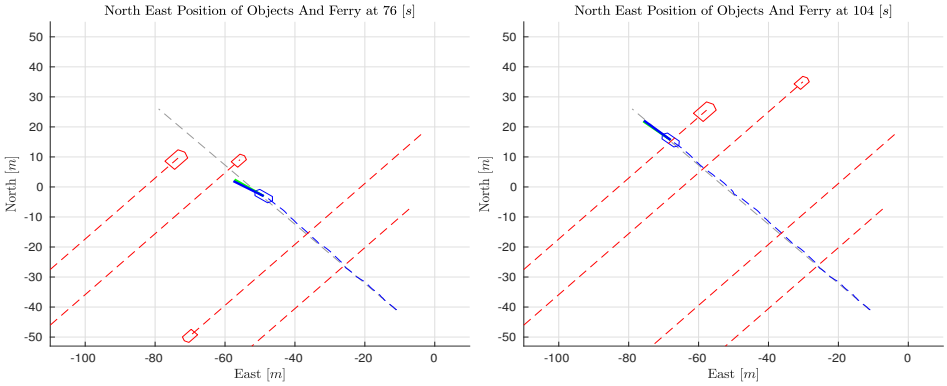
In this transit, the objects slow down as they approach the ferry. The transit is very similar to what was seen in **Section 4.3.2**, where the ferry has to stop and wait for a slow-moving object to pass. The transit is performed collision-free and is not very interesting in terms of collision avoidance, but it highlights another interesting feature, namely the ”stop-and-go” possibilities of a fully actuated vessel. From **Fig. 5.11** one can see that the ferry comes to a halt between $50s$ and $150s$ into the transit. Due to the DP capabilities of the vessel, it is able to track a constant pose reference, as can be seen from the absolute tracking error in **Fig. 5.8**, along with the heading and heading reference from **Fig. 5.9**, where the tracking error is less than $0.25m$ and heading error is less than $3deg$ during the halt. This way of handling a situation might be more time consuming than the alternatives, but in turn, has little risk attached to it, since it makes the intentions of the ferry highly visible to the operators of the other vessels. The halt arguably also aids the passenger comfort, given that the duration is limited, by reducing the maneuvering effort.

5.3.4 Transit 4

This transit is a multiple-paths transit where the objects slow down close to the ferry. Snapshots of the transit are included in **Fig. 5.12**. The ferry initially branches out to



(a) $t = 16[s]$ **(b)** $t = 48[s]$



(c) $t = 76[s]$ **(d)** $t = 104[s]$

Figure 5.6: Transit 1: Snapshots of the situation. Moving objects in red, ferry in blue with green heading vector and blue course vector.

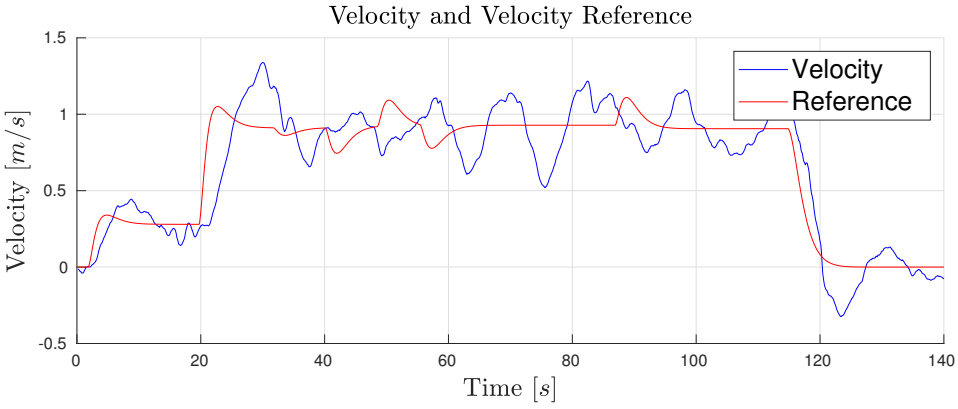


Figure 5.7: Transit 1: Velocity and velocity reference.

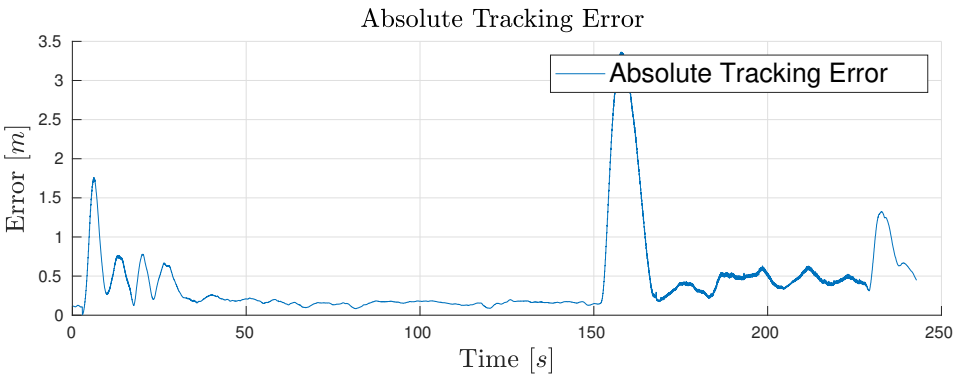


Figure 5.8: Transit 2: Absolute tracking error.

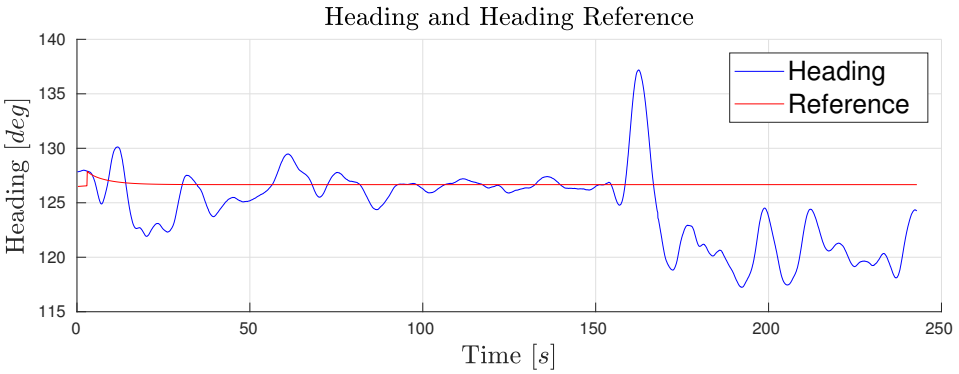
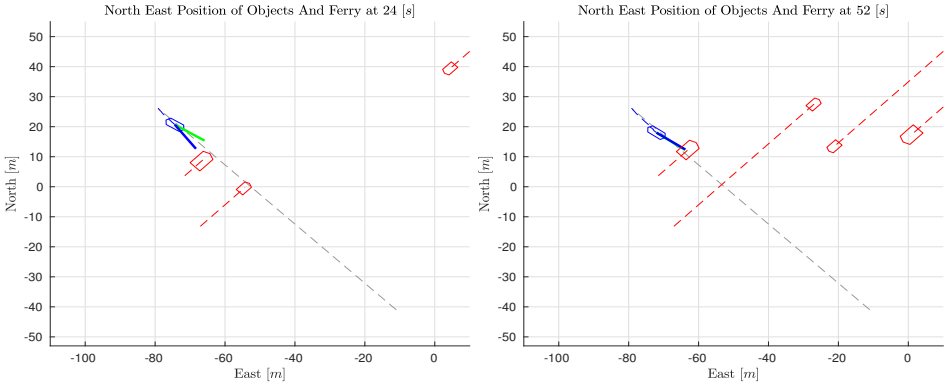
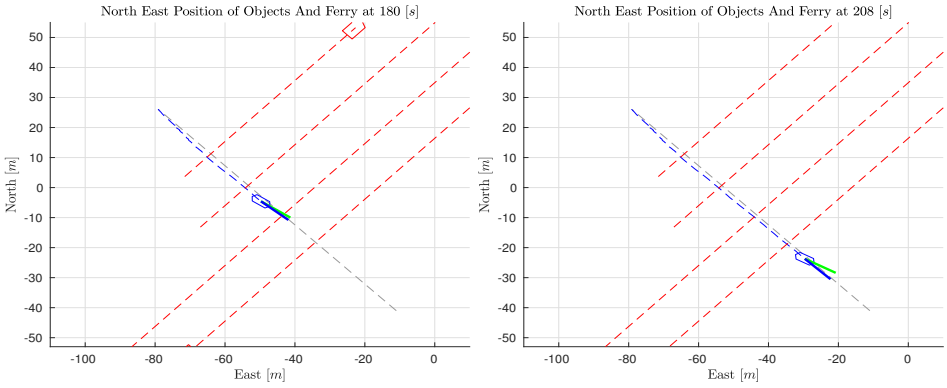


Figure 5.9: Transit 2: Heading and heading reference.



(a) $t = 24[s]$ **(b)** $t = 52[s]$



(c) $t = 180[s]$ **(d)** $t = 208[s]$

Figure 5.10: Transit 2: Snapshots of the situation. Moving objects in red, ferry in blue with green heading vector and blue course vector.

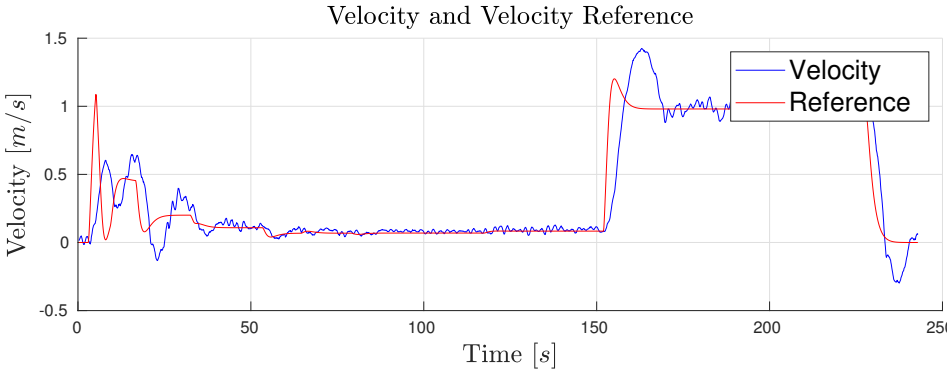
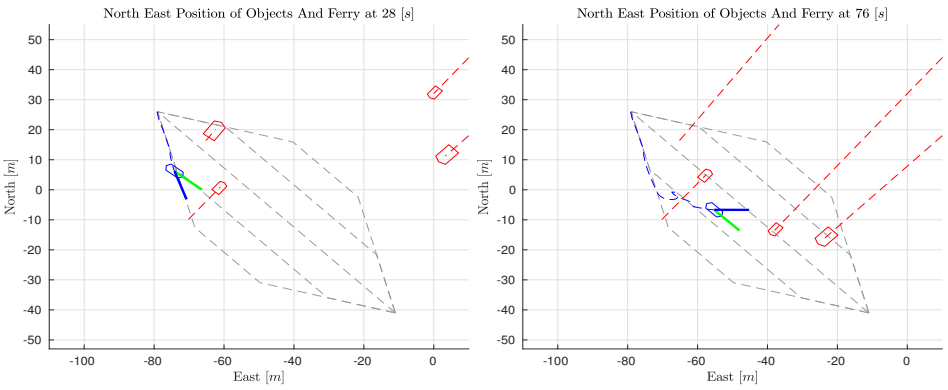


Figure 5.11: Transit 2: Velocity and velocity reference.

pass behind the two objects from the starboard side, and subsequently branches back to the centerpath in order to pass behind the objects from the port side. The velocity and velocity reference in **Fig. 5.13** show that the ferry has to reduce the velocity or stop on three occasions during the transit due to the moving objects reducing their velocity as the ferry gets closer. Similar situations have been seen previously in simulations. In many of the situations where this is the case, the object is not intersecting the path, but the added size of the object-representation is what intersects and halts the ferry. In reality, it is therefore clear to proceed, but the system is held back by its perception of the objects. An improvement to this could be to augment the object representation to include a factor representing the uncertainty. The added dimensions could, for example, be a function of how close the object is to the ferry, and the direction of travel the object has with respect to the ferry, travelling away, in proximity to the ferry would give a tighter area behind it than an object approaching from a distance on a near collision course.



(a) $t = 28[s]$ **(b)** $t = 76[s]$

(c) $t = 124[s]$ **(d)** $t = 156[s]$

Figure 5.12: Transit 4: Snapshots of the situation. Moving objects in red, ferry in blue with green heading vector and blue course vector.

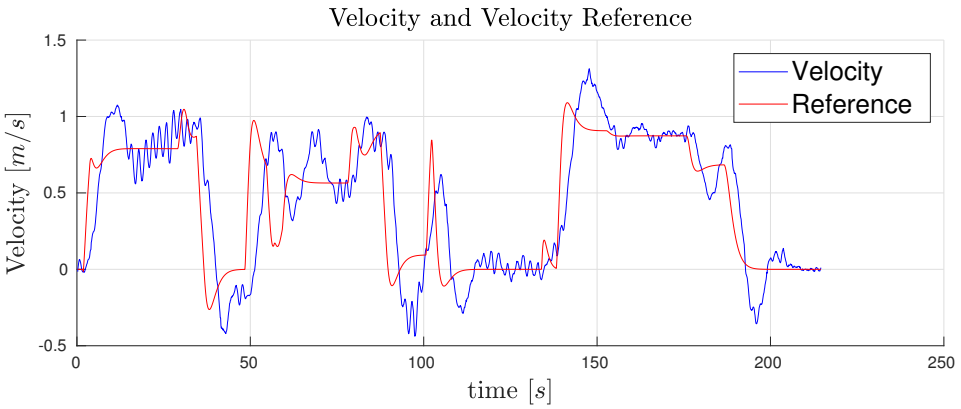


Figure 5.13: Transit 4: Velocity and velocity reference.

5.3.5 Transit 7

In Transit 7, the objects keep a constant heading and velocity. Snapshots from the crossing are included in **Fig. 5.15**. This scenario has the same initial conditions as Transit 1, where the only difference is that Transit 1 runs the SP-VP and Transit 7 runs the MP-VP.

In the transit, the ferry branches out initially to pass behind the two objects approaching from starboard, and subsequently branches back to the centerpath for the last half of the transit. This is similar to what we saw in Transit 4, but since the objects keep a constant velocity, the ferry avoids the reducing of velocity and stopping. An interesting comparison to Transit 1 is that, despite the branching, Transit 7 and Transit 1 have the same arrival time. It is hard to reason why the MP-VP favours branching in this case, but it might be that it was able to avoid using the nodes of one of the objects. This theory can be supported by **Fig. 5.14**, where the distance to the closest object is visualized for Transit 1 and Transit 7. The graphs show that the branching transit has a greater distance throughout most of the transit. In any case, one could argue that Transit 1 is better in terms of passenger comfort and power consumption and that the added safety in the increased distance to closest object does not compensate for the increased risk introduced by the misleading intention and maneuvering related to the branching.

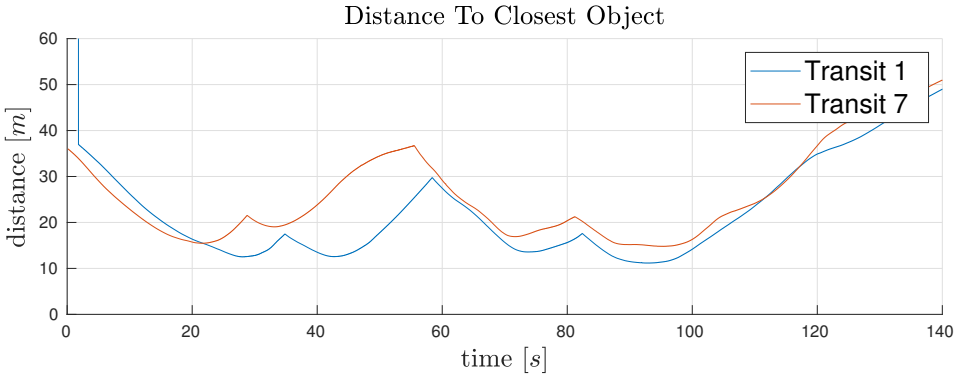
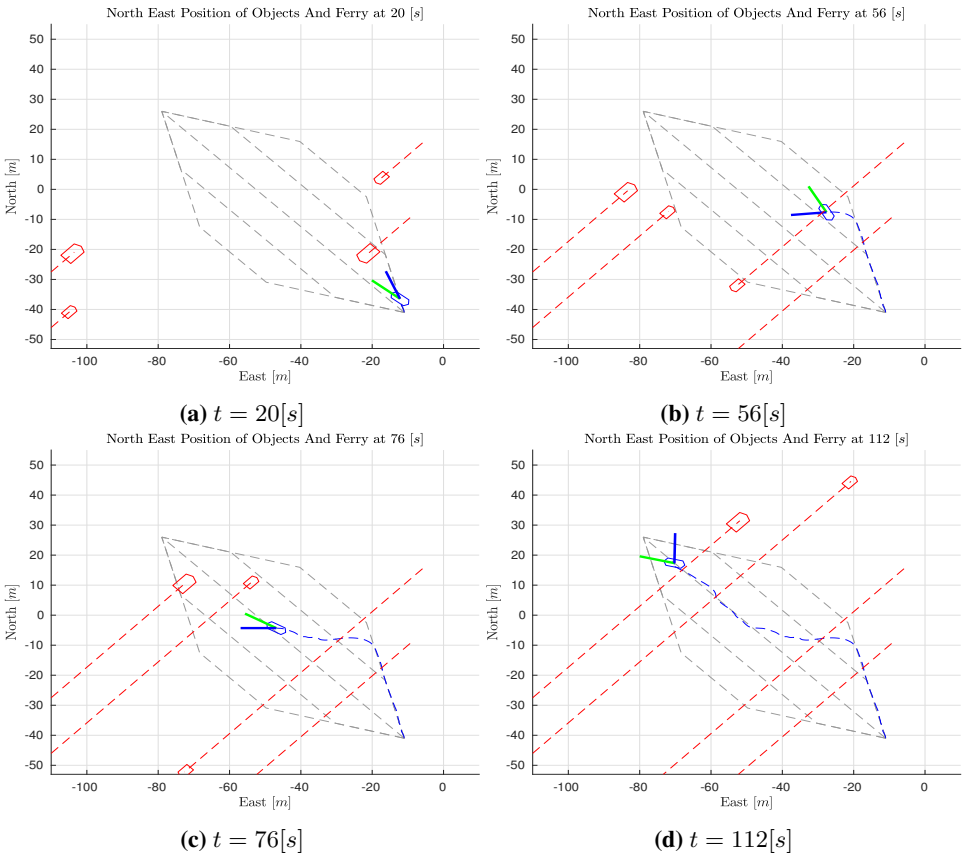


Figure 5.14: Transit 1 and 7: Distance to closest object.



(a) $t = 20[s]$ **(b)** $t = 56[s]$

(c) $t = 76[s]$ **(d)** $t = 112[s]$

Figure 5.15: Transit 7: Snapshots of the situation. Moving objects in red, ferry in blue with green heading vector and blue course vector.

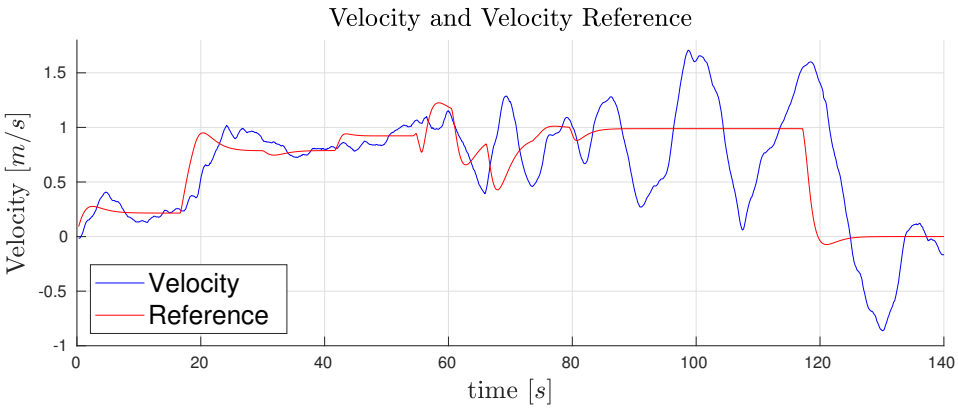


Figure 5.16: Transit 7: Velocity reference and actual velocity.

5.4 Discussion

The COLAV system runs problem-free on the OBC, and interfaces with all sub-modules correctly. The behaviour at sea trials is sufficiently similar to the simulations, with respect to the COLAV system. The parameters of the reference filter were somewhat off, due to a faulty thruster system model, which led to most of the error between simulations and sea trials. Improvement of this is left to future work.

As long as the COLAV system is run with a situational awareness system that consists of the same dummy object detection module used in simulations, the differences between sea trials and simulations are minor with respect to the behaviour of the COLAV system. It is therefore arguably more rewarding to spend resources on improving the emulated situational awareness and simulator than spending time preparing for sea trials. That being said, there is no good substitute for passenger feedback in a simulator, so sea trials are a necessity in the development process of autonomous passenger ferries.

As a passenger during sea trials, the situations when the ferry was halting and waiting for objects to pass, gave more of an impression that the ferry was avoiding a collision in a safe manner than the situations where the ferry branched out. A contributing factor to this is, of course, the oscillations in both heading and velocity that have been discussed earlier.

The DP capabilities of the vessel make the ”stop-and-go” strategy very viable. The ferry is stable during stationkeeping and uses very little power compared to maneuvering around an object. The energy aspect might change in an environment with more wind and current, but the investigation of this is left for future work.

During the sea trials, it became apparent that a $100m$ transit with four intersecting objects in the short time-span is a quite high-traffic situation. In many of the situations, it would be favourable to simply postpone the transit for half a minute or so. That being said, testing in worst-case situations is beneficial and can reveal unforeseen behaviour. Another

point that became apparent during sea trials was how aggressive and unreasonable the object behaviours were at times, especially considering the scale of the operational area. The behaviour would in many cases be considered reckless and dangerous.

Chapter 6

Conclusions and Future Work

Two versions of a complete COLAV system for autonomous transit with passenger ferries have been implemented and tested through simulations and sea trials. Both methods are based on path-velocity decomposition. The systems have been tested with up to four moving objects intersecting the transit, with different object behaviours. Both systems delivered satisfactory results in simulations as well as in sea trials.

The systems were also compared to a velocity obstacle algorithm. In the comparison, both the SP-VP and MP-VP method produced more predictable, energy-efficient and comfortable transits. This is much due to the predefined path and the global perspective of the deliberate layer.

The SP-VP method proved to be a simple but effective way of trajectory planning. It produced the most comfortable transits, with the least maneuvering, but suffered from long transit duration in situations with slow-moving objects. For short transits with high traffic, the ”stop-and-go” approach proves to be a safe, intuitive and predictable way of avoiding collision, but the benefits decrease as the transit length is increased and the environment is less confined.

The MP-VP method improved on the transit time of the SP-VP method in most scenarios, but at the same time introduced more maneuvering by branching in scenarios where it was not needed. The negative effect of this is less apparent in longer transit, where the branching maneuver makes up a smaller part of the transit. The possibility of branching increases the number of modes for collision avoidance and makes the system more robust to sudden changes in the environment. By adjusting the threshold for branching, and introducing a dynamic branch angle, the best features from each method can be achieved in the MP-VP.

The following should be investigated in future work:

- Increase the threshold for branching so that ”stop-and-go” is the primary mode for collision avoidance.

- Introduce a dynamic branch angle, to ensure that the ferry can get ”unstuck” from all stationary or slow-moving objects.
- Improve the reference filter to ensure a feasible trajectory reference at all times.
- Improve the object representation so that it better reflects the areas of risk around each particular object.

Bibliography

Amazon, December 2016. First prime air delivery. Available at https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011.

Bjørnø, J., 2016. Thruster-assisted position mooring of C/S Inocean Cat I drillship. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Brekke, E., Ruud, K. A., Eidsvik, J., 09 2018. LIDAR Extended object tracking of a maritime vessel using an ellipsoidal contour model. In: Sensor Data Fusion: Trends, Solutions, Applications (SDF). Bonn, Germany, pp. 1–6.

Campbell de Oliveira, S., Abu-tair, M., Naeem, W., 05 2013. An automatic COLREGs-compliant obstacle avoidance system for an unmanned surface vehicle. Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment 228, 108–121.

COLREGs, November 1995. International regulations for preventing collisions at sea. Available at http://inoa.net/zeilen/colreg.html#RULE8.

Eriksen, B.-O., Breivik, M., 08 2017. MPC-based mid-level collision avoidance for ASVs using nonlinear programming. In: IEEE Conference on Control Technology and Applications. Hawaii, USA.

Eriksen, B.-O., Breivik, M., 2018. A model-based speed and course controller for high-speed ASVs. In: 11th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles. Opatija, Croatia.

Eriksen, B.-O., Breivik, M., Pettersen, K., Wiig, M., 2016. A modified dynamic window algorithm for horizontal collision avoidance for AUVs. In: IEEE Multi-Conference on Systems and Control (MSC). Buenos Aires, Argentina.

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

Fraichard, T., Laugier, C., May 1993. Path-velocity decomposition revisited and applied to dynamic trajectory planning. In: [1993] Proceedings IEEE International Conference on Robotics and Automation. Atlanta, USA, pp. 40–45 vol.2.

FutureAgenda, January 2019. Autonomous transport , future agenda. Available at https://www.futureagenda.org/insight/autonomous-transport.

Gat, E., 1997. On three-layer architecture. Artificial Intelligence and Mobile Robots, 195–210.

Hagen, I. B., Kufoalor, D. K. M., Brekke, E. F., Johansen, T. A., 2018. MPC-based collision avoidance strategy for existing marine vessel guidance systems. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). Brisbane, Australia, pp. 7618–7623.

Hoberock, L., 1977. A survey of longitudinal acceleration comfort studies in ground transportation vehicles. In: Journal of Dynamic Systems, Measurement, and Control. Vol. 99.

Kant, K., Zucker, S. W., 1986. Toward efficient trajectory planning: The path-velocity decomposition. The International Journal of Robotics Research 5, 72–89.

Kirk, J., March 2015. Dijkstra's minimum cost path algorithm. Available at https://se.mathworks.com/matlabcentral/fileexchange/20025-dijkstra-s-minimum-cost-path-algorithm.

Kongsberg Maritime, 2018. Autonomous ship project, key facts about YARA Birkeland. URL https://www.km.kongsberg.com/ks/web/nokbg0240.nsf/AllWeb/4B8113B707A50A4FC125811D00407045?OpenDocument

Kortenkamp, D., Simmons, R., 2007. Springer Handbook of Robotics. Springer-Verlag, Berlin, Heidelberg, Ch. Robotic Systems Architectures and Programming, pp. 187–206.

Kuwata, Y., Wolf, M. T., Zarzhitsky, D., Huntsberger, T. L., Jan 2014. Safe maritime autonomous navigation with COLREGs, using velocity obstacles. IEEE Journal of Oceanic Engineering 39 (1), 110–119.

Loe, Ø. A. G., 2008. Collision avoidance for unmanned surface vehicles. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Lussier, B., Chatila, R., Guiochet, J., Ingrand, F., Killijian, M.-O., Powell, D., 05 2019. State of the art on dependability of autonomous systems.

M. Lavalle, S., Kuffner, J., 01 2000. Rapidly-exploring random trees: Progress and prospects. Algorithmic and computational robotics: New directions.

NRK, May 2019. Her føres passasjerene over fjorden av en selvkjørende ferge. Available at https://www.nrk.no/hordaland/her-fores-passasjerene-over-fjorden-av-en-selvkjorende-ferge-1.14563924.

Pedersen, A. A., 2019. Optimization based system identification for the milliAmpere ferry. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway.

RioTinto, January 2018. Rio Tinto autonomous trucks now hauling a quarter of Pilbara material. Available at http://www.mining.com/rio-tinto-autonomous-trucks-now-hauling-quarter-pilbara-material/.

ROS.org, November 2018. About ROS.
URL http://www.ros.org/about-ros/

Sang, L., Yan, X.-p., Wall, A., Wang, J., Mao, Z., 2016. CPA calculation method based on AIS position prediction. Journal of Navigation 69, 1409–1426.

Scania, March 2019. Autonomous trucks on public roads in Singapore. Available at https://www.scania.com/global/en/home/experience-scania/features/autonomous-truck-platoon-in-singapore.html.

Sæther, B., 2019. Development and testing of navigation and motion control systems for milliAmpere. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Thyri, E. H., 2018. Trajectory planning and following for autonomous passenger ferries.

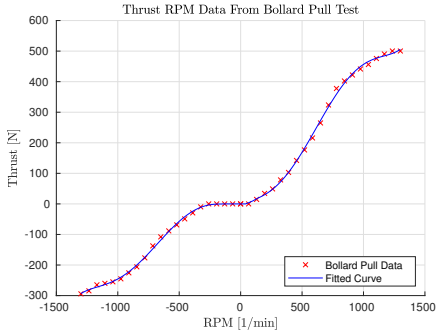
Wärtsilä, 2018. Wärtsilä achieves notable advances in automated shipping with latest successful tests. Available at https://www.wartsila.com/media/news/28-11-2018-wartsila-achieves-notable-advances-in-automated-shipping-w

Appendix A

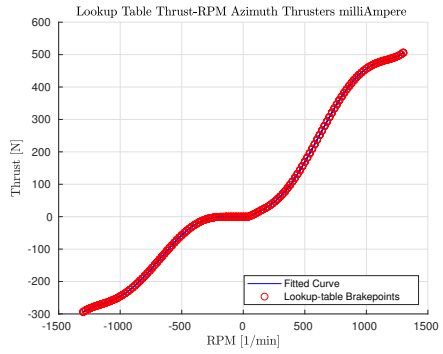
Thruster Model

Thrust Model

The mapping between the RPM and thrust is based on data from a bollard pull test performed on the ferry 06.06.2018 by Tobias Torben when he developed the thrust allocation system. The data can be seen in **Table. A.1**. To make a model of the thrust as a function of RPM, 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 **Fig. A.1a**. From the polynomial, an evenly distributed data set of corresponding RPM and thrust values were calculated, for use in a linear look-up table. The fitted curve and the brake-points can be seen in **Fig. A.1b**.



(a) 5th degree polynomial curve, fitted to the data from a bollard pull test. The data is listed in **Table. A.1**.



(b) Brake-points relating RPM to thrust for the thruster model look-up table.

Figure A.1: Curve-fitting of the RPM and thrust 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 A.1: Results from the bollard pull test performed with the milliAmpere ferry 06.06.2018.

Azimuth-angle Model

The dynamics of the azimuth angle is also modeled based on data from the same day of testing. Data from step inputs in the azimuth angle setpoints are presented in **Table. A.2**. Since this is all 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

$$e_\theta = \theta_{set} - \theta, \tag{A.1}$$

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. A.2**. The azimuth angle model from Simulink is displayed in **Fig. A.2**.

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 inputs a 3 DOF thruster force $\boldsymbol{\tau}_{thruster}$, therefore a transformation needs to be done. The position of the thrusters can be seen in **Fig. A.3**. Both thrusters are placed on the front-aft centerline of the vessel, symmetrical about the midship beam. The relationship between

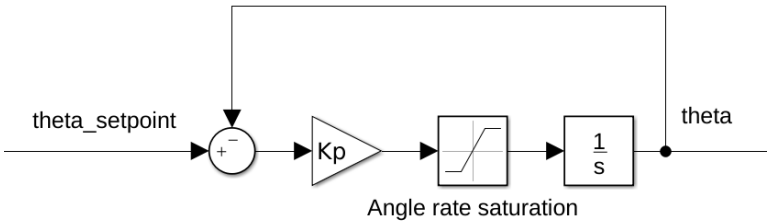


Figure A.2: Azimuth angle modeling in Simulink. The angular velocity is set proportional to the error, and saturated to max angular rate. The saturation value is found from the data in **Table. A.2**.

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 A.2: 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 thruster forces and angles and the 3 DOF force is

$$\boldsymbol{\tau}_{thrusters} = \begin{bmatrix} cos(\theta_{front}) & cos(\theta_{rear}) \\ sin(\theta_{front}) & sin(\theta_{rear}) \\ -l_{TA}sin(\theta_{front}) & l_{TA}sin(\theta_{rear}) \end{bmatrix} \begin{bmatrix} F_{front} \\ F_{rear} \end{bmatrix}, \tag{A.2}$$

where $l_{TA} > 0$ is the front and rear thruster arm.

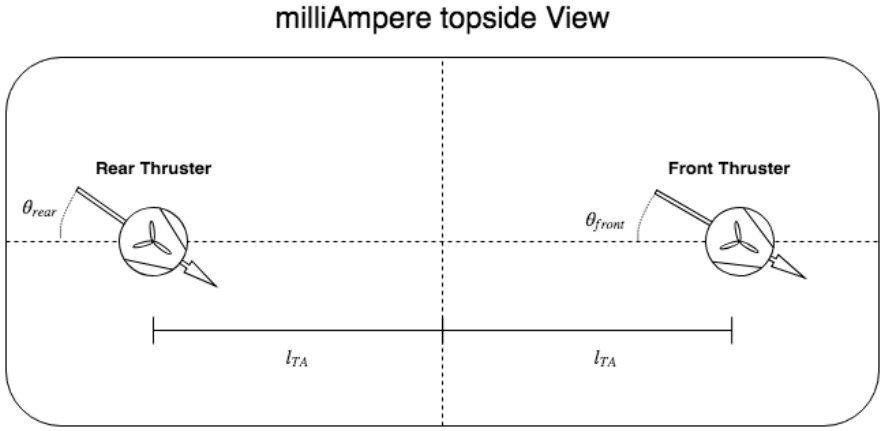


Figure A.3: Topside view of the thruster layout on milliAmpere. The ferry is symmetrical, and hence the front and rear thruster arm is the same.

Appendix B

Additional Experimental Results

B.1 Transit 3

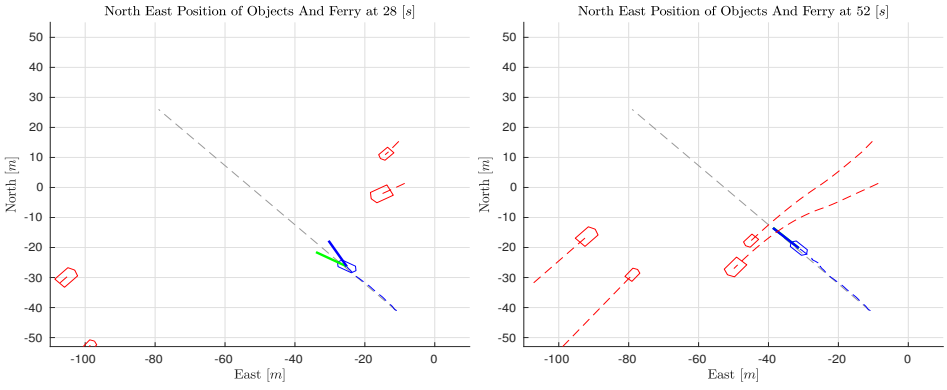
In Transit 3, the objects try to pass in front of the ferry. The transit runs the SP-VP algorithm. An overview of the transit can be seen in **Fig. B.1**. The velocity profile velocity profile is shown in **Fig. B.2**. The velocity has similar oscillations as have been seen earlier. The cause for this is discussed in **Section 5**. This effect of the instability in yaw can also be seen in **Fig. B.1a** where the heading and velocity is clearly not aligned, despite the heading reference being constant, and aligned with the course reference.

B.2 Transit 5

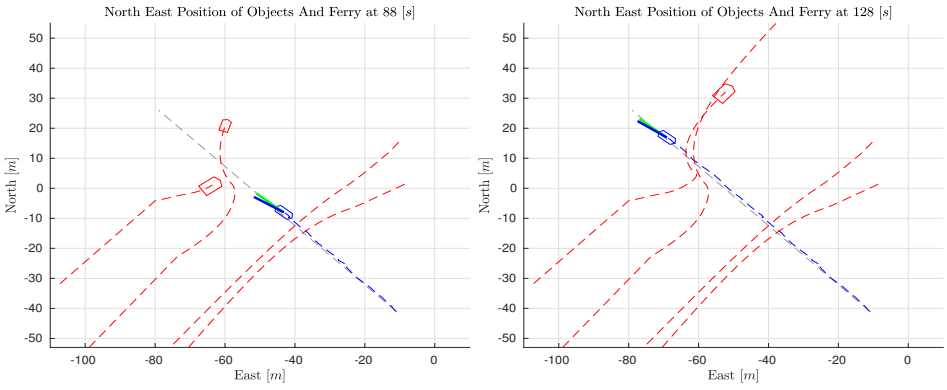
In Transit 5, the objects try to pass in front of the ferry. Snapshots from the transit can be seen in **Fig. B.3**. The transit runs the MP-VP algorithm. The velocity profile can be seen in **Fig. B.4**. This is the transit scenario transit with where the ferry was the most unstable, which is reflected both in the velocity reference in **Fig. B.4** and in the heading and heading reference in **Fig. B.5**. At one point, about $80s$ into the transit, the tracking-error in heading is $70deg$.

B.3 Transit 6

Transit 6 runs the SP-VP algorithm. The objects slow down as they get close to the ferry. Snapshots are shown in **Fig. B.6** and the velocity profile is shown in **Fig. B.7**. The transit starts of by waiting for an object to pass, and thereafter proceed in transit velocity. During



(a) $t = 28[s]$ **(b)** $t = 52[s]$



(c) $t = 88[s]$ **(d)** $t = 128[s]$

Figure B.1: Transit 3: Snapshots of the situation. Moving objects in red, ferry in blue, with green heading vector and blue course vector.

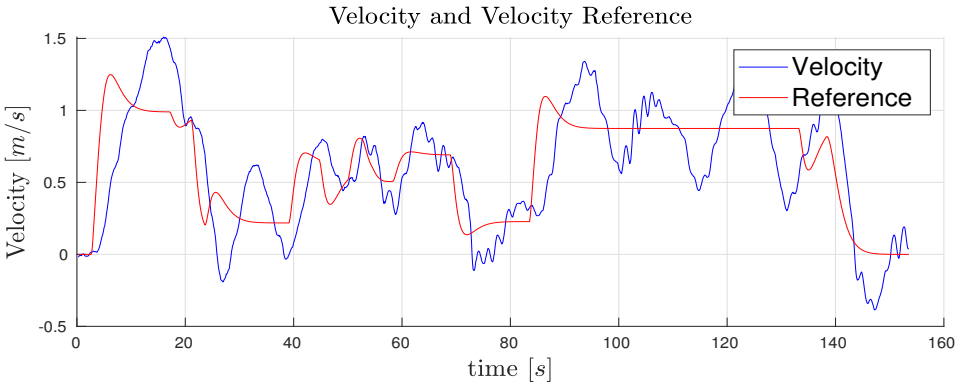
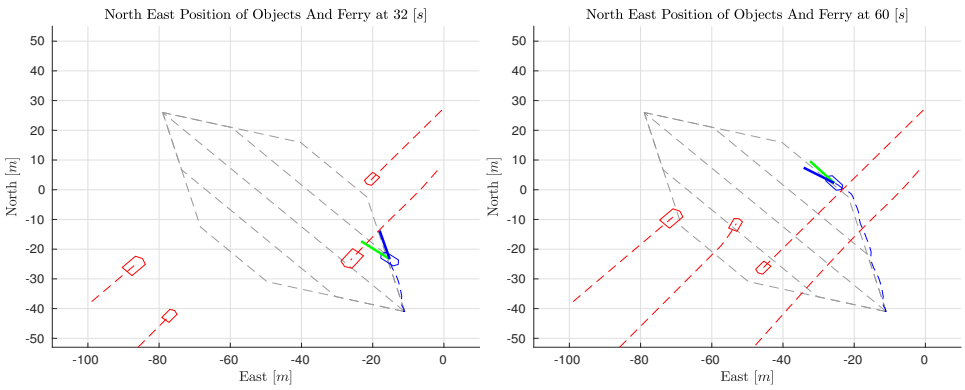
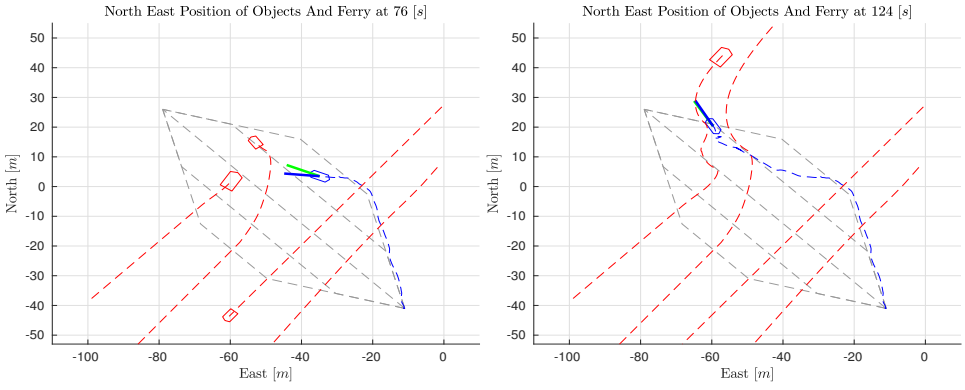


Figure B.2: Transit 3: Velocity and velocity reference.



(a) $t = 32[s]$ **(b)** $t = 60[s]$



(c) $t = 76[s]$ **(d)** $t = 124[s]$

Figure B.3: Transit 5: Snapshots of the situation. Moving objects in red, ferry in blue, with green heading vector and blue course vector.

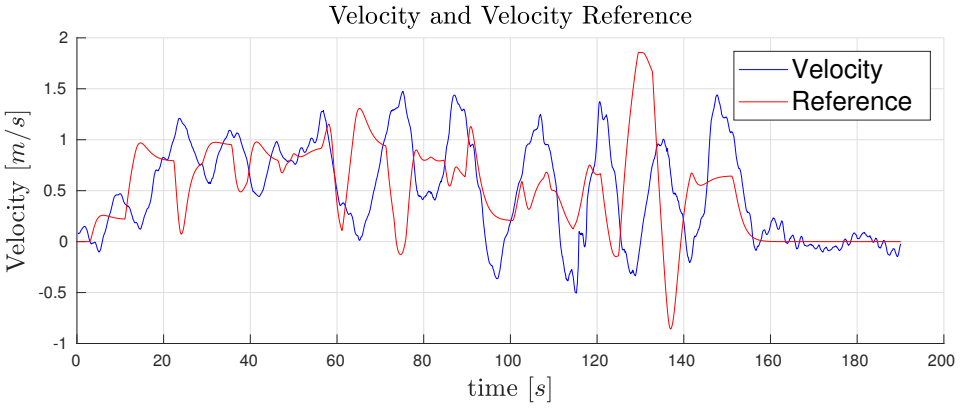


Figure B.4: Transit 5: Velocity and velocity reference.

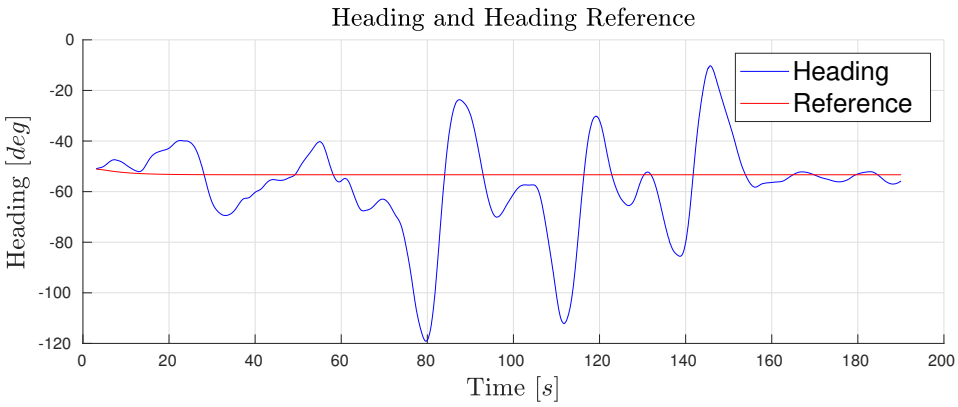


Figure B.5: Transit 5: Heading and heading reference.

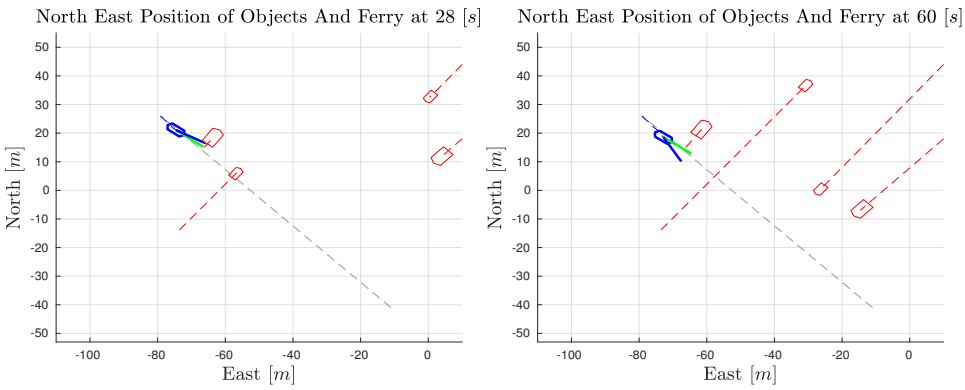
this transit, the heading remained somewhat stable, as can be see in **Fig. B.8**. This is reflected in the velocity, where it is able to track the reference without (big) oscillations.

B.4 Transit 8

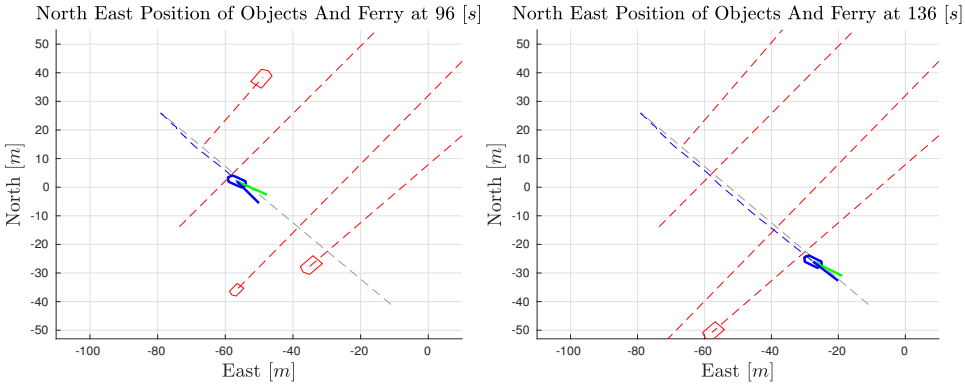
Transit 8 runs the SP-VP algorithm. The objects have constant behaviour. In Transit 8 and Transit 9, the objects have a steeper approach angle to the path than the rest of the transits. The algorithm starts off slowly until the objects pass, and speeds up for the rest of the transit. This is very similar to typical simulations, and show clearly how the SP-VP functions.

B.5 Transit 9

Transit 9 runs the MP-VP algorithm. The objects have constant behaviour, and have identical initial conditions as Transit 8. The algorithm branches out, and thereby keeps close to transit velocity the whole transit, and arrives $20s$ earlier, compared to Transit 8.



(a) $t = 28[s]$ **(b)** $t = 60[s]$



(c) $t = 96[s]$ **(d)** $t = 136[s]$

Figure B.6: Transit 6: Snapshots of the situation. Moving objects in red, ferry in blue, with green heading vector and blue course vector.

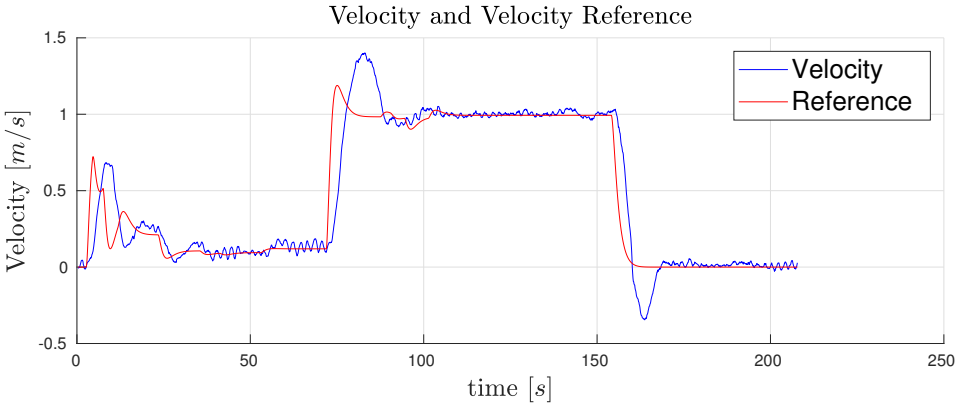


Figure B.7: Transit 6: Velocity and velocity reference.

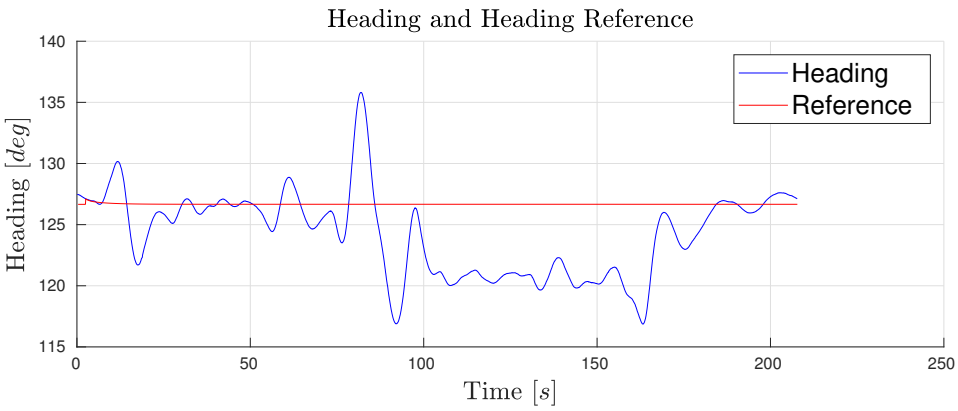
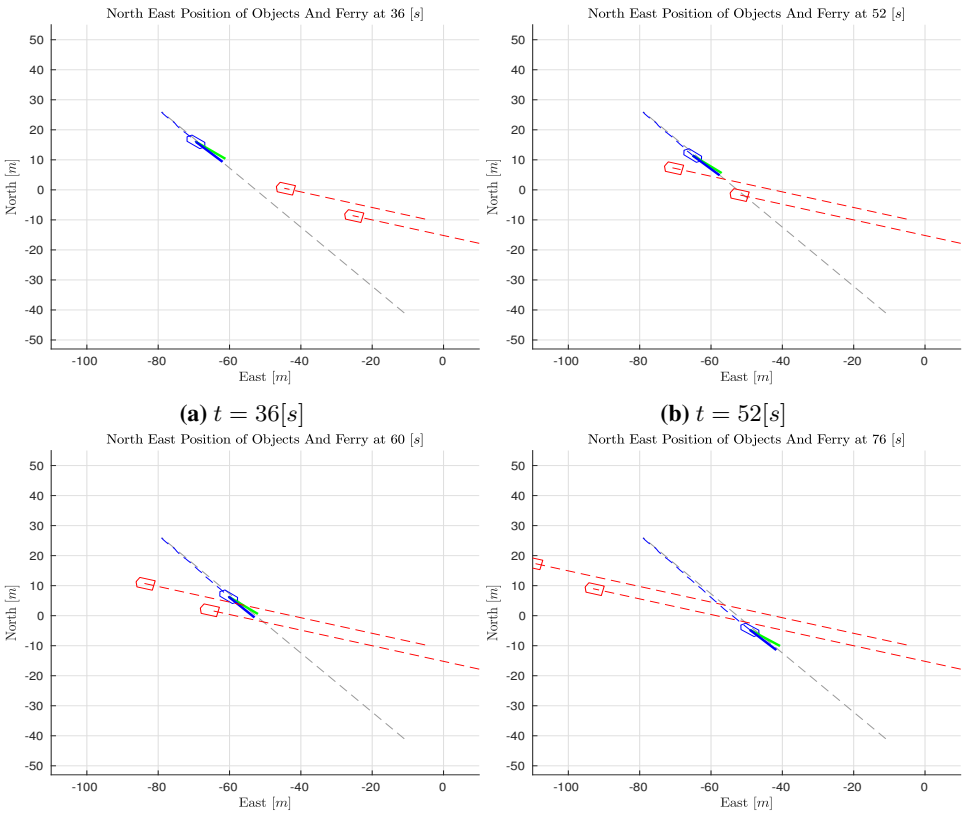


Figure B.8: Transit 6: Heading and heading reference.



(a) $t = 36[s]$ **(b)** $t = 52[s]$

(c) $t = 60[s]$ **(d)** $t = 76[s]$

Figure B.9: Transit 8: Snapshots of the situation. Moving objects in red, ferry in blue, with green heading vector and blue course vector.

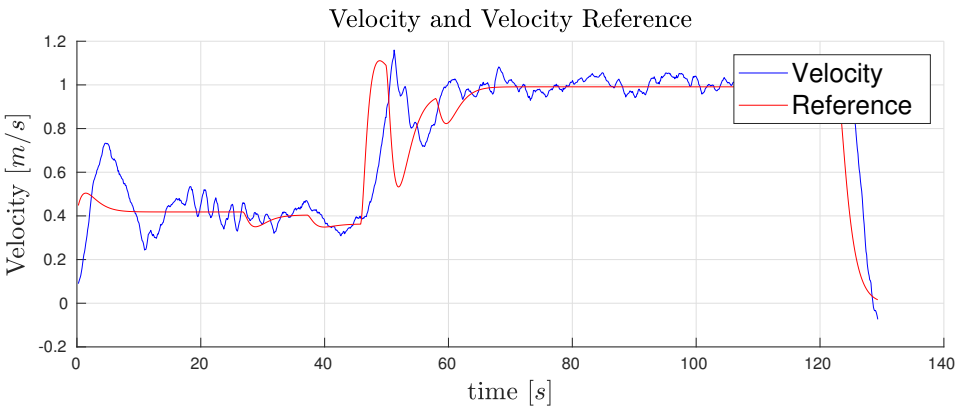
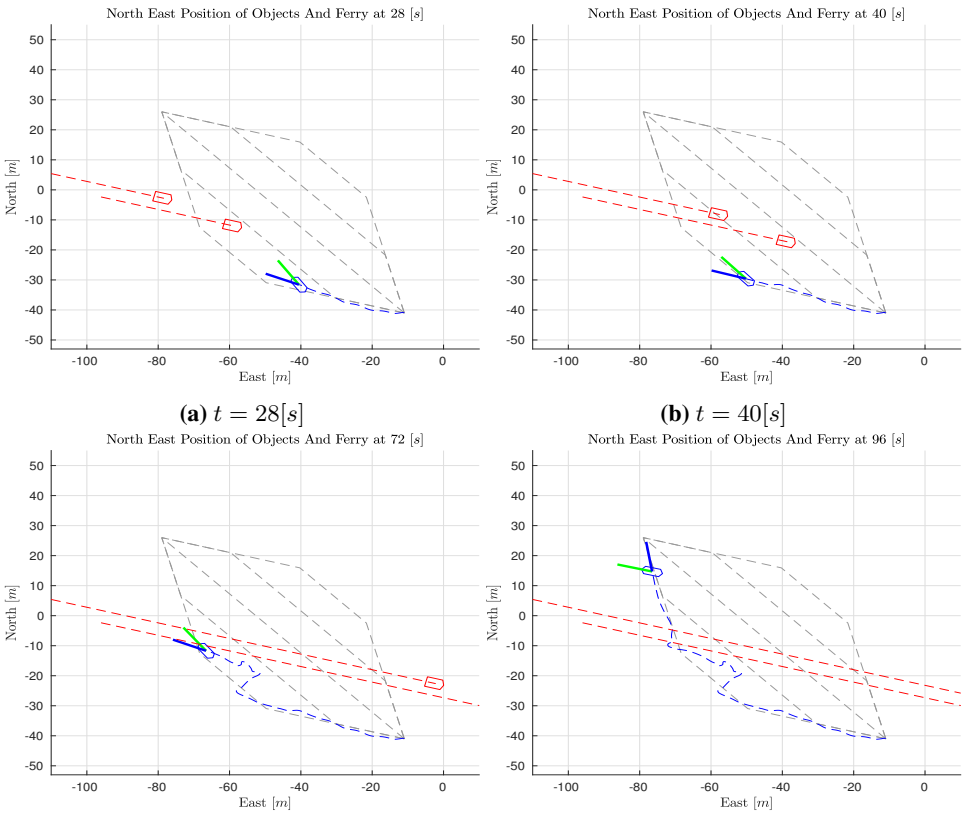


Figure B.10: Transit 8: Velocity and velocity reference.



(a) $t = 28[s]$ **(b)** $t = 40[s]$

(c) $t = 72[s]$ **(d)** $t = 96[s]$

Figure B.11: Transit 9: Snapshots of the situation. Moving objects in red, ferry in blue, with green heading vector and blue course vector.

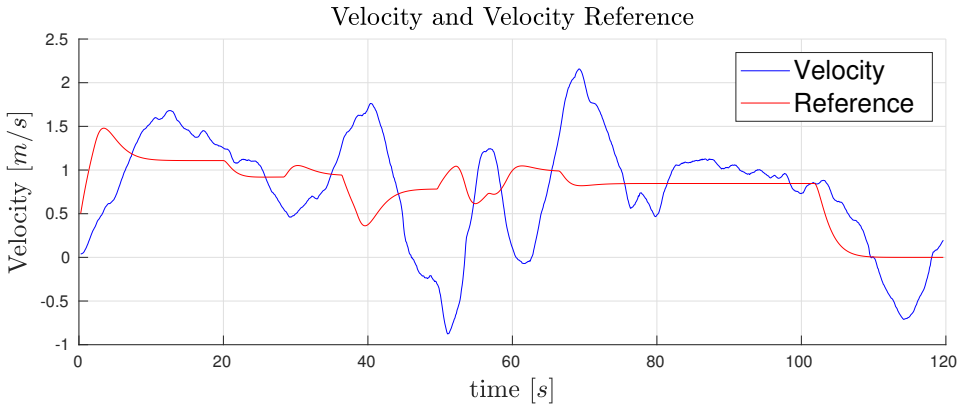


Figure B.12: Transit 9: Velocity and velocity reference.

Emil Hjelseth Thyri

A Path-Velocity Decomposition Approach to Collision Avoidance for Autonomous Passenger Ferries

NTNU
Norwegian University of
Science and Technology