Preface

Things to declare:

MATLAB simulator files provided by Emil Thyri

Guidance and general assistance: Morten Breivik, Emil Thyri

ENC charts for scenario inspiration: Olex AS

Casadi's Constrained Multiple Shooting example from example pack lay the foundation of the algorithm.

Software used:

MATLAB, Inkscape, Draw.io.

Other tools:

CasADi. IPOPT.

Erlend Hestvik, 20.12.2021

Abstract

test acronym for error control: Automatic Identification System (AIS)

Sammendrag

test acronym for error control: AIS

Contents

List of Figures

Acronyms

AIS Automatic Identification System

ASV Autonomous Surface Vessel

COLREGs Convention on the International Regulations for Preventing Collisions at Sea

dCPA distance at Closest Point of Approach

DOF Degrees Of Freedom

IPOPT Interior Point OPTimizer

LOS Line of Sight

MPC Model Predictive Control

NED North East Down

NLP NonLinear Programming

OCP Optimal Control Problem

OS Own Ship

RK4 Runge Kutta 4th order

tCPA time to Closest Point of Approach

TS Target Ship

1 Introduction

- something something this thesis is about trajcetory planning
- had this idea I wanted to try out
- This chapter explains my motivation for the thesis. discusses previous work of the same subject. Explains the problem as I see it, and my contributions to a solution. lastly an outline of the rest of the thesis for a quick intro of every section.

1.1 Motivation

- Worked on the same subject matter for a "fordypningsprosjekt" (finn godt ord).
- Autonomous vehicle control is an important milestone on the journey to a fully autonomous life.
- It's also just fricking cool on a conceptual level.
- Fishing industries and other marine industries are 'behind the curve' and not given as much attention as land based industries.
- A great learning opportunity for practical implementations of theory learned over the past two years.
- All in all a highly relevant project for the career trajectory I want.
- AI is pretty cool too I guess
- wanted to see if there could be a difference if autonomous vessels had more advanced prediction algorithms.
- just make something up.
- Find picture of some autonomous vessel or ferry

1.2 Previous Work

- cite Loe 2007 For in-depth look at many different methods.
- cite Vagale et al. 2021 For a review of path planning algorithms.
- cite Zhang et al. 2021 For another big review on navigation systems for ASV

- cite Huang et al. 2020 For another review of Collision avoidance.
- cite Park et al. 2020 For an alternative approach to Trajectory planning with similar-ish results.
- Cite someone to prove that Autonomous surface vessels are real? :thinking_emoji:
- item I *need* to use Convention on the International Regulations for Preventing Collisions at Sea (COLREGs) at least once so that the formating looks nice later.

1.3 Problem Description

- Many papers on trajectory planning emjoy simple "cpa" prediction.
- Many algorithms end up creating a very 'active' vessel, which is different from how most humans navigate.
- Trajectory planning and collision avoidance in one package
- Would it help if we had the tools to more accurately predict other vessels.

1.4 Contributions

- A novel MPC based path following trajectory planner that accounts for both static and dynamic obstacles.
- An evaluation of the fitness of numerical optimization as trajectory planning backbone.
- documented simulations experimenting with the difference 'Prediction Level' makes.
- documented problems that numerical optimization based trajectory planner algorithms might encounter.
- propose mitigation methods for aformentioned problems.

1.5 Outline

- Chapter 2 is background theory.
- Chapter 3 is algorithm development.
- Chapter 4 is simulation results.
- Chapter 5 is conclusion and future work.

2 Background Theory

This chapter will introduce the concepts and theory neccessary to understand the design and intent behind the trajectoy planning algorithm, as well as the discussion on it's functionality. The goal of the chapter is is to give the reader enough intuition of the applied theory that the proposed arguments and solutions should make sense. In addition, the chapter is structured so that it should be easy to quickly navigate and read about specific topics.

2.1 Vessel Modelling

A mathematical model is a tool for describing physical systems and expressing how they change over time, respond to external forces, and how stable the system might be. Models are very useful when designing control systems as they translate the physical into equations that computers can understand. When making a model it is often useful to separate the dynamics of the different parts of the system we are interested in, these are the Degrees Of Freedom (DOF) the system has, and is often the directions the system can move, though they can also just be nondescript generalized coordinates. Deciding which DOF to separate out and model the dynamics of is often what separates models from each other, it is pointless to model an aspect of a system that there is no intent to interact with. For example a ship has six DOF, see Figure 1, for modelling a control system for stationkeeping all six are important because stationkeeping involves keeping the whole ship as steady as possible. When modelling for path following on the other hand it is not important what the heave, roll, or pitch of our vessel is and so the dynamics of those DOF can safely be ignored. The model used to describe our vessel in this thesis is based on the theory and notation by (Fossen 2011), and is a 3DOF nonlinear mass-damper system with thruster dynamics and no external disturbances such as wind or currents. The dynamics of the vessel can be described by the differential equations below:

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu} \tag{2.1}$$

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} \tag{2.2}$$

where $\boldsymbol{\eta}$ is the pose of the vessel parameterized in the tangetnal plane North East Down (NED), $\boldsymbol{\nu}$ are the velocities of the vessel, parameterized in the BODY frame of the vessel. And $\boldsymbol{\tau}$ are forces and torque applied to the system. The NED frame can be said to be

inertial for short distance control objectives, and in this frame he x-axis points towards true north, the y-axis points east and the z-axis points down towards the center of the planet. Thus; $\boldsymbol{\eta} = [x\ ,\ y\ ,\ \psi]^T \in \mathbb{R}^3$ are the vessel's North, East and Heading values, which are the three DOF of the system. The velocities $\boldsymbol{\nu} = [u\ ,\ v\ ,\ r]^T \in \mathbb{R}^3$ are the surge, sway, and yaw rate of the vessel. In the BODY frame there are no fixed rules for where the axis are pointing, but the common convention for modelling vehicles is that the x-axis points along the lognitudal axis of the vessel, the y-axis points along the lateral axis and the z-axis points along the vertical axis. This is also seen in Figure 1. The anchor point for the BODY frame is arbitrary but always fixed to the vessel and moves with it. The forces and toruqe $\boldsymbol{\tau} = [F_X\ ,\ F_Y\ ,\ F_N]^T \in \mathbb{R}^3$ are the forces acting along the logitudal axis, lateral axis, and torque about the vertical axis of the vessel's BODY frame. (TODO: vent hæ? ble plutselig usikker på rammedefinisjonen til $\boldsymbol{\tau}$. Det stemmer vel at det er BODY?). The rotation matrix $\mathbf{R}(\psi)$ rotates the BODY velocities into NED movement about the vessel's heading, and is definied as:

$$\mathbf{R}(\psi) = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

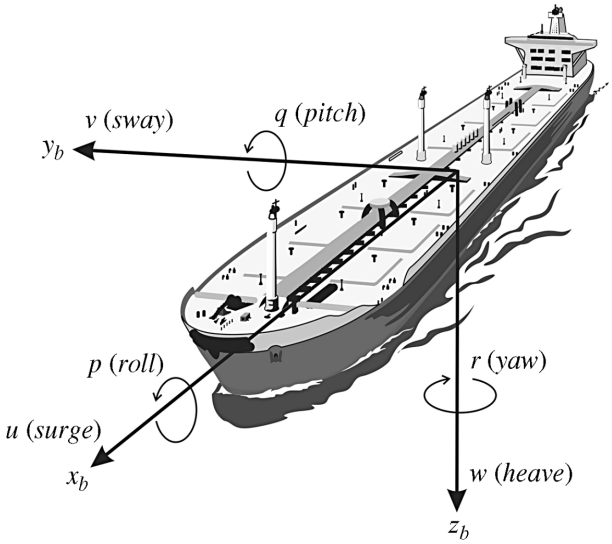


Figure 1: A ships 6 degrees of Freedom, from (Fossen 2011).

In (2.2) the $\mathbf{M}$ matrix is the inertia matrix of the system, which describes how 'heavy' the DOF are to nudge, in addition to the vessel's inherent inertia from being massive the vessel must also push water out of the way when it moves, this is what is known

as hydrodynamic added mass and is linearly added to the inertia matrix. The coriolis matrix $\mathbf{C}$ also has to include hydrodynamic added mass, however for the purpose of this thesis it is not important to know the parameters for either of these matrices or for the dampening matrix $\mathbf{D}$. That is because a trajectory planning algorithm needs to work regardless of vessel parameters, (Pedersen 2019) explains more in-depth how system parameters can be found. Continuing on, the dampening matrix is a linear combination of the linear dampening stemming from water viscosity and non-linear dampening from cross-flow, once again the parameters themselves are not strictly relevant to this thesis, but intuition is important. The result are matrices in the following form:

$$\mathbf{M} = \begin{bmatrix} m11 & m12 & m13 \\ m12 & m22 & m23 \\ m31 & m32 & m33 \end{bmatrix} \tag{2.4}$$

$$\mathbf{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & c13(\boldsymbol{\nu}) \\ 0 & 0 & c23(\boldsymbol{\nu}) \\ c31(\boldsymbol{\nu}) & c32(\boldsymbol{\nu}) & 0 \end{bmatrix} \tag{2.5}$$

$$\mathbf{D}(\boldsymbol{\nu}) = \begin{bmatrix} d11(\boldsymbol{\nu}) & 0 & 0 \\ 0 & d22(\boldsymbol{\nu}) & d23(\boldsymbol{\nu}) \\ 0 & d32(\boldsymbol{\nu}) & d33(\boldsymbol{\nu}) \end{bmatrix} \tag{2.6}$$

The dampening matrix can be a bit of a compuational nightmare and can be simplified to a linear and diagonal matrix without too much of a detrimental impact on our simulations. The justification for this simplification is the underlying assumption that the reference output from the trajectory planner algorithm will be parsed through a final control module that will account for dampening. The risk is that the end result from the trajectory planner turns out to be infeasible, but that's a problem for another thesis.

$$\mathbf{D}(\boldsymbol{\nu}) = \begin{bmatrix} d11 & 0 & 0 \\ 0 & d22 & 0 \\ 0 & 0 & d33 \end{bmatrix} \tag{2.7}$$

Finally, a word on heading vs course. Throughout this thesis the terms course and heading might be used interchangeably, but the words are strictly not synonymous. Heading is equivalent with yaw as both denote a rotation about the vessel's third axis, the difference between the two is that yaw is often a relative term describing a change by

some degrees from one arbitrary pose to another. Heading is an absolute term and is often based on compass directions, meaning 0°heading equates to the nose of the vessel pointing towards true north. Neither heading nor yaw is equivalent with course, which is strictly the direction of travel relative to true north. In a simplified world void of disturbances heading and course will align during straight line travel, but external forces such as wind or currents will cause the two angles to deviate. Likevise sideslip caused by a non-zero sway velocity when turning will also introduce a deviation between course and heading (Fossen 2011). However this difference is mostly unrelated to the work put forth in this thesis, and so the terms heading and course might be used interchangeably. Althought it often makes sense to deliberately pick one term over the other.

2.2 Trajectory Planning

(TODO: Savner et lite avsnit om Shortest Signed Angle eller WrapTo2Pi.) Because the vessel dynamics are described by a model expressed as a set of time-invariant ordinary differential equations, any desired state can be reached by solving for the sequence of inputs that will take the vessel from a given initial condition to said state. In the context of this thesis "state" refers to the pose, $\boldsymbol{\eta}$, of the vessel. The simplest application of this would be moving in a straight line from point A to point B. The solution is simply to find the input sequence which turns the vessel to the correct course and then maintaining a forward speed until point B is reached. The straight vector line from point A to point B can be thought of as the desired or reference path, while the sequence of states achieved by applying the input sequence is the trajectory. Instead of having just one destination there might be multiple waypoints forming the path, and the optimal input sequence that makes the controlled vessel, from here on called "Own Ship" (OS), travel along the path depends on what criteria are considered important. A trajectory generated with fuel economy in mind might look very different from a trajectory generated with shortest transit time in mind, even if both are following the same path. Other factors such as obstacles or disturbances will also influence the trajectory, combining all the factors and generating the desired input sequence is the act of trajectory planning.

There are many methods for trajectory planning. Some are conceptually simple and fast to compute, but lack robustness and situational adaptability. Other method can be incredibly complex and and computationally expensive, but in return incredibly robust to disturbances and adaptable to any situation. An example of a simple trajectory planner would be a Line of Sight (LOS) guidance law while something extremely advanced would

be training a deep neural network. For an overview: in this thesis a LOS guidance law will be applied to generate a reference trajectory, the reference is then used as part of a formulation of an Optimal Control Problem (OCP) with a cost to penalize deviation from the reference in addition to other factors. The OCP is then discretized as a NonLinear Programming (NLP) problem using a method called direct multiple shooting, finally the NLP is solved with an Interior Point OPTimizer (IPOPT) solver.

Line of Sight Guidance

This guidance method is perhaps the most intuitive; consider the waypoints $\mathrm{WP_k}$ and $\mathrm{WP_{k+1}}$, the simplest path from one to the other would be straight line. Therefor the most obvious control method would be to maneuver onto the straight line, and follow it along to the end. The distance of the OS to the straight line is called the cross track error y_e and the distance along the line to the end is called the along track error x_e. The along track error is not of any importance to this thesis, it is assumed that the controlled vessel will maintain a steady velocity, and there are no temporal constraints on reaching the goal.

As explained in (Lekkas and Fossen 2013); given the OS's position (x,y), the cross track and along track errors from the straigth line as defined by $\mathrm{WP_k}$ ($\mathrm{x_k}$,$\mathrm{y_k}$) and $\mathrm{WP_{k+1}}$ ($\mathrm{x_{k+1}}$, $\mathrm{y_{k+1}}$) are:

$$\begin{bmatrix} x_e \\ ye \end{bmatrix} = \mathbf{R}^T(\gamma_p) \begin{bmatrix} x - x_k \\ y - y_k \end{bmatrix} \tag{2.8}$$

where R is the rotation matrix from the inetiral frame to the straight line's frame. γ_p is the horizontal path-tangental angle, or the 'angle' of the straight line path in relation to the inetrial frame if that makes more sense, see Figure 2 for a visual decomposition. The rotation matrix $\mathbf{R}$ is given by:

$$\mathbf{R}(\gamma_p) = \begin{bmatrix} cos(\gamma_p) & -sin(\gamma_p) \\ sin(\gamma_p) & cos(\gamma_p) \end{bmatrix} \tag{2.9}$$

with γ_p:

$$\gamma_p = \mathrm{atan2}(y_{k+1} - y_k, x_{k+1} - x_k) \tag{2.10}$$

The control objective is to drive $y_e(t) \to 0.$ as t trends towards infinity. Assuming a steady velocity this is done by selecting a course that steers the OS in the direction that reduces y_e. How fast the error y_e is suppressed is tuned by a proportional gain factor,

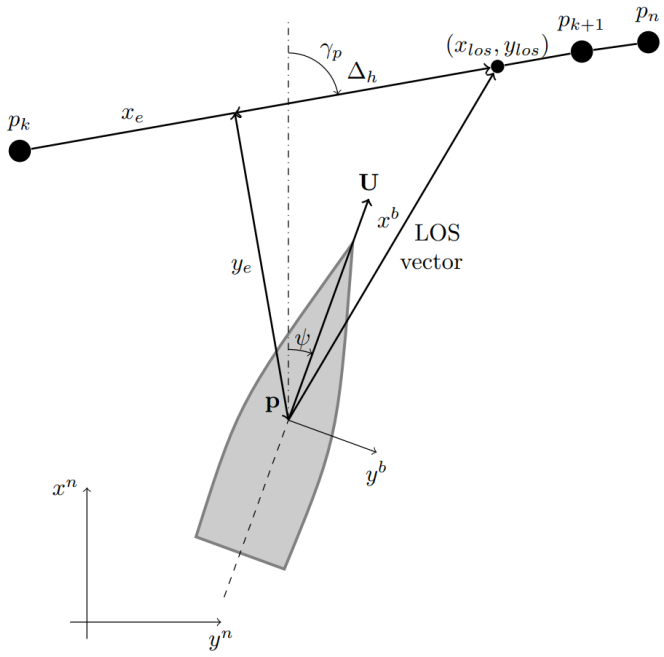


Figure 2: Line of sight guidance geometry for straight lines, here with zero sideslip. Image courtesy of (Lekkas and Fossen 2013)

Δ, that is often called look ahead distance. The desired heading is given by:

$$\psi_d = \gamma_p + \arctan(\frac{-y_e}{\Delta}) \tag{2.11}$$

and consequently desired course:

$$\chi_d = \psi_d + \beta \tag{2.12}$$

where β is the sideslip of the OS. Because real life situations are rarely, if ever, devoid of disturbances that introduce sideslip and crab angles there is one common improvement that can be made: Integrate the cross track error and use both current cross track error and it's integral when calculating desired heading. The equation for $\dot{y}_{int}$ and ψ_d then become:

$$\dot{y}_{int} = y_e \tag{2.13}$$

$$\psi_d = \gamma_p - \arctan(K_p y_e + K_i \dot{y}_{int}) \tag{2.14}$$

where K_p and K_i are gain parameters proportional to the lookahead distance, typically $K_p = (1/\Delta)$, $K_i = Kp * \kappa$ with $\kappa > 0$ being some design variable.

A reference trajectory is generated by using the LOS law as described to guide the contolled vessel from it's initial position through all the waypoints, and saving the desired positions and velocities after each time step. For a path with more than two waypoints a

simple index incrementation can be used when the vessel is within a certain distance from it's current target waypoint. The reference trajectory from t_0 to N (TODO: Dette er vel teknisk sett blanding av diskret og kontinuerlig notasjon) iterations of LOS applications is of the form:

$$\overline{\boldsymbol{\eta}}_{ref} = [\boldsymbol{\eta}_{t0} \ , \ \boldsymbol{\eta}_{t+1} \ , \ \dots \ , \boldsymbol{\eta}_N] \in \mathbb{R}^{3\times N} \tag{2.15a}$$

$$\overline{\boldsymbol{\nu}}_{ref} = [\boldsymbol{\nu}_{t0} \ , \ \boldsymbol{\nu}_{t+1} \ , \ \dots \ , \boldsymbol{\nu}_N] \in \mathbb{R}^{3\times N} \tag{2.15b}$$

Optimal Control Problem

Numerical optimization is a vast field within mathematics, (Wright, Nocedal et al. 1999) explains it well: There are no universal optimization algorithm. Instead an algorithm must be tailored to the optimization problem. Within the context of trajectory planning there are different parameters to optimize for, some examples are: maintaining steady velocity, suppressing sway, minimalizing fuel waste, minimizing distance to goal, and there are many more. The general expression for an optimization problem can be written as simple as:

$$\underset{x\in\mathbf{R}^n}{\text{Minimize}} \quad f(x) \tag{2.16a}$$

$$\text{Subject to:} \quad c_i(x) = 0, \quad i \in \mathcal{E} \tag{2.16b}$$

$$c_i(x) \geq 0, \quad i \in \mathcal{I} \tag{2.16c}$$

Where $f(x)$ is some continuous function, c_i are constraint functions on the system which $f(x)$ exists in, and $\mathcal{E}$ and $\mathcal{I}$ are indices pertaining to if the constraint c_i is an equality or inequality constraint. In the context of this thesis the thing to minimize is some nebulous cost function associated with path following, and the constraints are the physical model of the system that guarantees feasibility as well as safety constraints to avoid collisions. The cost function is then some function of the vessel's state, reference trajectory, and control input. The two constraints are the system dynamics from (2.2) and (2.1). And then additional constraints for collision safety and initial conditions. A new general OCP definition is thus given by the following:

$$\text{Minimize} \quad L(\boldsymbol{\theta}(t), \boldsymbol{\theta}_{ref}(t), \boldsymbol{\tau}(t)) \tag{2.17a}$$

$$\text{Subject to:} \quad \dot{\boldsymbol{\theta}}(t) = \mathbf{J}(\boldsymbol{\theta}, \boldsymbol{\tau}) \tag{2.17b}$$

$$\mathbf{h}(\boldsymbol{\theta}(t), \boldsymbol{\tau}(t)) \leq \mathbf{0} \tag{2.17c}$$

$$\boldsymbol{\theta}(t_0) - \overline{\boldsymbol{\theta}}_0 = \mathbf{0} \tag{2.17d}$$

where L is the cost function, $\boldsymbol{\theta} = [\boldsymbol{\eta}^T \ , \ \boldsymbol{\nu}^T]^T$ and $\boldsymbol{\tau}$ is still the same as in (2.2). $\mathbf{J}(\boldsymbol{\theta}, \boldsymbol{\tau})$ Are the model dynamics (2.1), (2.2). $\overline{\boldsymbol{\theta}}_0$ are the given intial conditions of the system. The solution to the optimization problem is the series of inputs $\boldsymbol{\tau}$ which minimizes the integral of the cost L from t_0 to t_{end}. And L has the form of a quadratic function akin to a weighted least squares: (TODO: Eh, grei formulering?)

$$L(\boldsymbol{\theta}(t), \boldsymbol{\theta}_{ref}(t), \boldsymbol{\tau}(t)) = (\boldsymbol{\theta}(t) - \boldsymbol{\theta}_{ref}(t))^T \mathbf{Q} (\boldsymbol{\theta}(t) - \boldsymbol{\theta}_{ref}(t)) + \mathbf{K}_\tau \boldsymbol{\tau}^2 \tag{2.18}$$

Where the diagonal of the $\mathbf{Q}$ matrix are the weight coefficients of deviating from the reference and $\mathbf{K}_\tau$ denote the associated cost of applying force in the three DOF.

Solving the OCP can be done using a multitude of methods, Eriksen and Breivik 2017 suggest discretizing the OCP into a NLP using a method called direct multiple shooting.

NonLinear Programming

The author would like to note that the technique used in this section, direct multiple shooting, is outside the scope of the author's knowledge. Everything the author knows about this technique was learned over the course of the master thesis project, and it's highly recommended to read the full formulation by (Eriksen and Breivik 2017) which is the formulation that the implementation for this thesis heavily builds upon. Another great resource for direct multiple shooting are the video lectures of (Gros 2017). Also note that functions and definitions from the previous section on OCP carry over, for example $\boldsymbol{\theta} = [\boldsymbol{\eta}^T \ , \ \boldsymbol{\nu}^T]^T$ still holds.
Direct multiple shooting is a OCP discretization technique where both the states and control inputs are excplicitly defined as decision variables. The NLP is then a reformulation of (2.17) where L is redefined as a discretized cost function with N_p control intervals

steps:

$$\Phi(\boldsymbol{\omega}, \boldsymbol{\omega}_{ref_{1:N_p}}) = \sum_{k=0}^{N_p-1} ((\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_{ref_{k+1}})^T Q(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_{ref_{k+1}}) + K_{\boldsymbol{\tau}} \boldsymbol{\tau}_{\boldsymbol{k}}^2) \tag{2.19}$$

where $\boldsymbol{\omega} = [\boldsymbol{\theta}_0^T \ , \ \boldsymbol{\tau}_0^T \ , \ ... \ , \ \boldsymbol{\theta}_{N_{p-1}}^T \ , \ \boldsymbol{\tau}_{N_{p-1}}^T \ , \ \boldsymbol{\theta}_{N_p}^T]^T \in \mathbb{R}^{9N_p+6}$ (TODO: Er det rett at det skal være N_{p-1} i subscript og ikke $NP-1$?). is a vector containing $9N_p + 6$ decision variables. Because $\boldsymbol{\tau}_k$ does not have an accociated reference in this thesis; it is separated out as it's own part of the function. Here, $\mathbf{Q}$ is still a sparse 6x6 matrix where the diagonal contain the tuning parameters, and $\mathbf{K}_{\boldsymbol{\tau}}$ are still tuning parameters on control input. The complete NLP will end up in the form of:

$$\min_{\boldsymbol{\omega}} \quad \boldsymbol{\Phi}(\boldsymbol{\omega}, \boldsymbol{\omega}_{ref_{1:N_p}}) \tag{2.20a}$$

$$\text{Subject to:} \quad \boldsymbol{\omega}_{lb} \leq \boldsymbol{\omega} \leq \boldsymbol{\omega}_{ub} \tag{2.20b}$$

$$\mathbf{g}(\boldsymbol{\omega})_{lb} \leq \mathbf{g}(\boldsymbol{\omega}) \leq \mathbf{g}(\boldsymbol{\omega})_{ub} \tag{2.20c}$$

where $\boldsymbol{\omega}_{lb}$ and $\boldsymbol{\omega}_{ub}$ are the lower and upper bounds on the permitted values for $\boldsymbol{\omega}$, this is meant to reduce the space searched when solving the NLP, as well as limit the decision variables to physically feasible values. $\mathbf{g}(\boldsymbol{\omega})$ is a vector of constraint functions that are similarly bound by an lower and upper bounds, where the bounds define if any given function in $\mathbf{g}$ is an equality or inequality constraint. Due to the way direct multiple shooting defines the decision variables the programmed solver that solves the NLP is free to place the states and velocities anywhere within the constraints. It is therefor important to implement equality constraints that force the ending of one control interval and the beginning of the next to line up. this is called closing the shooting gaps, an illustration of what shooting gaps are can be seen in Figure 3. These equality constraints are called shooting constraints and to create them begin by defining an integrator function $\mathbf{F}(\boldsymbol{\theta}_k, \boldsymbol{\tau}_k)$ using any technique, in this thesis the following Runge Kutta 4th order (RK4) method will be used:

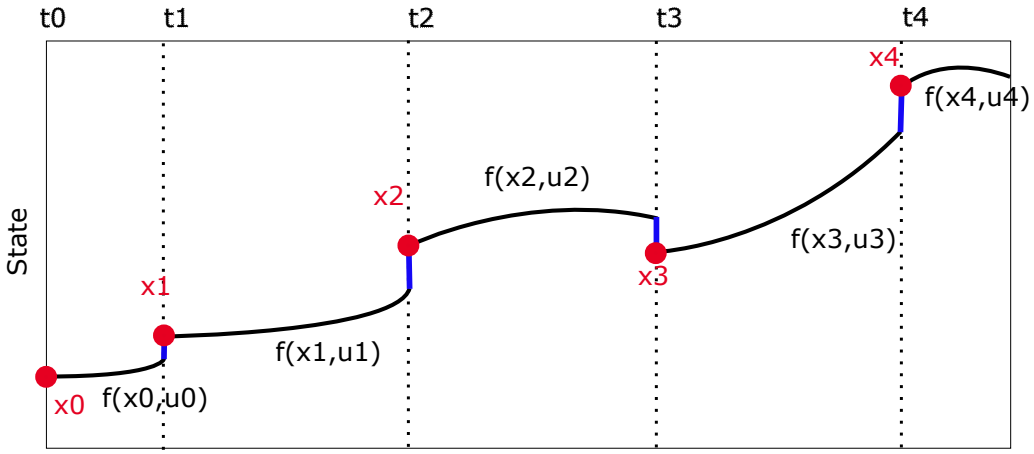


Figure 3: A physically feasible trajectory is formed by "pinching" the shooting gaps close. Reproduction from (Gros 2017).

$$
\begin{aligned}
k_1 &= \mathbf{f}(\boldsymbol{\theta}_k \ , \ \boldsymbol{\tau}_k) \\
k_2 &= \mathbf{f}(\boldsymbol{\theta}_k + \frac{h}{2} k_1 \ , \ \boldsymbol{\tau}_k) \\
k_3 &= \mathbf{f}(\boldsymbol{\theta}_k + \frac{h}{2} k_2 \ , \ \boldsymbol{\tau}_k) \\
k_4 &= \mathbf{f}(\boldsymbol{\theta}_k + h k_3 \ , \ \boldsymbol{\tau}_k) \\
\mathbf{F}(\boldsymbol{\theta}_k, \boldsymbol{\tau}_k) &= \boldsymbol{\theta}_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
\tag{2.21}
$$

with h being a selected discretized time step size. With $\mathbf{F}$ it is now possible to calculate $\boldsymbol{\theta}_{k+1}$ given $\boldsymbol{\theta}_k$ and $\boldsymbol{\tau}_k$ The shooting constraints are then formed as:

$$
\mathbf{g}(\boldsymbol{\omega}) = \begin{bmatrix} \overline{\boldsymbol{\theta}}_0 - \boldsymbol{\theta}_0 \\ \mathbf{F}(\boldsymbol{\theta}_0, \boldsymbol{\tau}_0) - \boldsymbol{\theta}_1 \\ \mathbf{F}(\boldsymbol{\theta}_1, \boldsymbol{\tau}_1) - \boldsymbol{\theta}_2 \\ \vdots \\ \mathbf{F}(\boldsymbol{\theta}_{N_{p-1}}, \boldsymbol{\tau}_{N_{p-1}}) - \boldsymbol{\theta}_{N_p} \end{bmatrix}
\tag{2.22}
$$

Setting the lower and upper bounds for $\mathbf{g}$ equals to zero enforces the equality constraints and pinches the shooting gaps close. The final missing piece for the trajecotry planner is to formulate constraints to ensure a collision free trajectory. Similarly to the shooting constraints the obstacle constraints are also placed in $\mathbf{g}$, their formulation is discussed in Chapter 2.3

The theory behind constructing an NLP in a way that a machine can understand and solve it is a topic for a whole new thesis. In this thesis CasADi, (Andersson et al. 2019), is

used as a framework for constructing the NLP, the NLP is solved with an IPOPT solver, (Wächter and Biegler 2006), that comes included with CasADi. A practical explanation of constructing and solving the NLP is the topic of Chapter 3.

Model Predictive Control

With the system dynamics modelled, and a control law formulated as an NLP it is now possible to conduct trajectory planning by selecting a discretized time step size, h, deciding how many control intervals to predict foward in time, and then solving the NLP from any initial condition (which will still be discussed in Chapter 3). Because the IPOPT solver solves for all control intervals simultaneously, its output contains the optimal trajectory as decided by the selected cost function. It also contains optimal velocities and control inputs needed to achieve the desired state, as described by the system dynamics. However it is unrealistic to assume that the modelled dynamics are able to perfectly represent reality, blindly following the optimal trajectory is therefor a fool's errand. This is where the control technique Model Predictive Control (MPC) comes into play. MPC is a method in which the system is simulated from the present until the end of the control period. The first control inputs from the solution are saved and applied to the system for it's next control interval, the rest of the solution is then discarded and new measurements of the state of the system are taken. Using the new measurements as the new intial conditions, the process starts over; Simulate until the end of control period, apply first control input to next control interval, discard rest of solution, redo measurements, repeat. This introduces feedback to the system, which allows it to react and adjust to unexpected disturbances, this greatly increases the robustness of the automated system. (Qin and Badgwell 1997) (TODO: usikker på den her, den er delvis relevant for det som er skrevet.)

2.3 Collision Avoidance

- Mangler kanskje et lite avsnitt om state machine for å holde på COLREGs klassifikasjoner frem til situasjonen er klarert.

Having constructed the means of finding an optimal trajectory, the next task at hand is making sure the trajectory is collision free. It would be difficult to claim any sort of optimality without asserting if the trajectory is able to effectively avoid obstacles, collision

avoidance is therefor just as important a task as the construction of the trajectory planning algorithm. Collision avoidance is an umbrella term for many different smaller tasks; from risk assessment to escape maneuvers. For the purposes of this thesis it is assumed that information about obstacles in the near vicinity of the OS is readily available and not subject to disturbances or distortion. The task at hand can then be separated out into two pieces: Static obstacle avoidance and COLREGs compliance.

Static Obstacles

A static obstacle is any object or hindernace in the water that does not move on a timescale comparable to the one of the controlled vessel, such as skerries or a pier. The static obstacle $\mathcal{O}_{s_i}$ is presumed to be in the form of a polygon with n corners parameterized in NED so that:

$$\mathcal{O}_{s_i} = \begin{bmatrix} N_1 & E_1 \\ N_2 & E_2 \\ \vdots & \vdots \\ N_n & E_n \end{bmatrix}^T \tag{2.23}$$

where the point $(N_1\ ,\ E_1)$ is the first point of the polygon defining the obstacle, and the following points are sequential in either a clockwise or counter-clockwise direction. With N obstacles that can be put together on the form:

$$\mathcal{O}_s = [\mathcal{O}_{s_1}\ ,\ \mathbf{NaN}\ ,\ \mathcal{O}_{s_2}\ ,\ ...\ ,\ \mathbf{NaN}\ ,\ \mathcal{O}_N] \in \mathbb{R}^{2\times c} \tag{2.24}$$

where $\mathbf{NaN} = [NaN\ ,\ NaN]^T$ is inserted between each obstacle to separate them, and c is some function of N and n to get the correct column dimension. (TODO: wow dette var dårlig beskrevet, alternativ formulering: the column dimension c is dependant on the amount and shape of the obstacles.). When the obstacles are gathered in this form, consrtaints can be generated the followig way:
Consider a fan of scan lines radiating from the OS with fixed length and angle. The intersection point between a scan lines and the line between any two columns in $\mathcal{O}_s$ forms the basis of the constraint $\mathbf{o}_{sc} = (o_n\ ,\ o_e)$ in NED.
The constraint function is the cross track error between the position of the vessel and a

line orthogonal to the scan line which crosses the point $\mathbf{o}_{sc}$, calculated like in (2.8):

$$\mathbf{o}_{sc_{ye}} = -\sin(\gamma_p) * (N - o_n) + \cos(\gamma_p) * (E - o_e) \tag{2.25}$$

where (N, E) are the north and east position of the OS, and γ_p is the angle of the orthogonal line w.r.t NED. See Figure 4 for a visualization of the geometry. The scan lines are generated at each discretized step of the reference trajectory, and every found intersection between a line the static obstacles has it's own associated cross track error that is added to the function $\mathbf{g}(\omega)$:

$$\mathbf{g}(\omega) = \begin{bmatrix} \vdots \\ \mathbf{o}_{sc_{ye}} \\ \vdots \end{bmatrix} \tag{2.26}$$

with the lower bounds of $\mathbf{g}(\omega)$ defining the allowed distance beteen the vessel and the constraint lines, while the upper bounds should be infinite. This creates a convex free set bound by the static obstacles near the OS. (TODO: kutt den siste setningen hvis jeg ikke kommer på mer å skrive.)

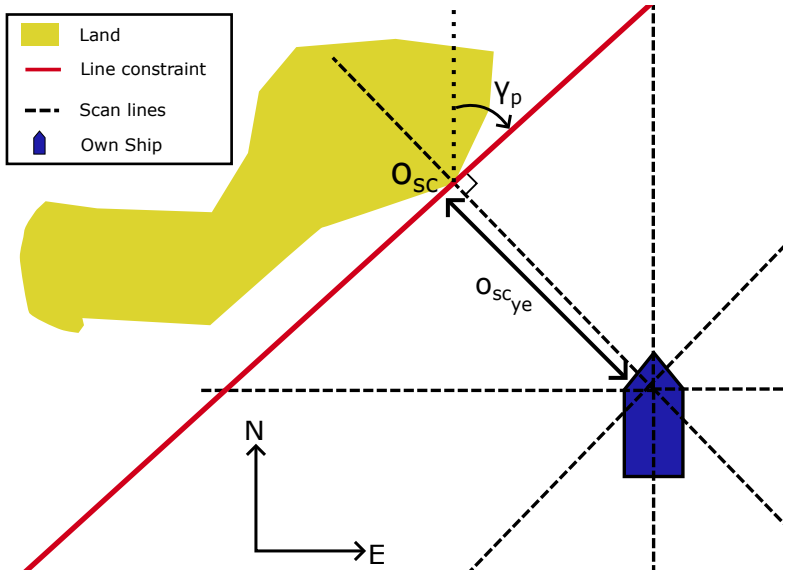


Figure 4: Geometry for straight line constraints used to handle static obstacles.

COLREGs Compliance

The COLREGs (IMO 1972) are a set of rules developed with the purpose of preventing collisions between two or more vessels at sea. The rules are sectioned into six parts; A - General, B - Steering and Sailing Rules, C - Light and Shapes, D - Sound and Light

Signals, E - Exemptions, F - Verification of Compliance. In part A it is written "These Rules shall apply to all vessels upon the high seas and in all waters connected therewith navigable by seagoing vessels.", which means any aspiring Autonomous Surface Vessel (ASV) must be able to comply. It is part B that is the most relevant to the work of this thesis, as it contains the rules for maneuvering in the vicinity of other vessels. The following is a non-exhaustive list of the rules that are most relevant for this thesis, a more comprehensive examination of the rules can be found in (Cockcroft and Lameijer 2012).

Rule 7: Risk of Collision

(d) *In determining if risk of collision exists the following considerations shall be among those taken into account:*

(d)(i) *such risk shall be deemed to exist if the compass bearing of an approaching vessel does not appreciably change;*

(d)(ii) *such risk may sometimes exist even when an appreciable bearing change is evident, particularly when approaching a very large vessel or a tow or when approaching a vessel at close range.*

Rule 8: Action to avoid collision

(a) *Any action taken to avoid collision shall be taken in accordance with the Rules of this Part and shall, if the circumstances of the case admit, be positive, made in ample time and with due regard to the observance of good seamanship.*

(b) *Any alteration 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; a succession of small alterations of course and/or speed should be avoided.*

Rule 13: Overtaking

(b) *A vessel shall be deemed to be overtaking when coming up with another vessel from a direction more than 22.5 degrees abaft her beam.*

Rule 14: Head-on situation

(a) *When two power-driven vessels are meeting on reciprocal or nearly reciprocal courses so as to involve risk of collision each shall alter her course to starboard so that each shall pass on the port side of the other.*

Rule 15: Crossing situation

When two power-driven vessels are crossing so as to involve risk of collision, the vessel which has the other on her own starboard side shall keep out of the way and shall, if the circumstances of the case admit, avoid crossing ahead of the other vessel.

Rule 17: Action by stand-on vessel

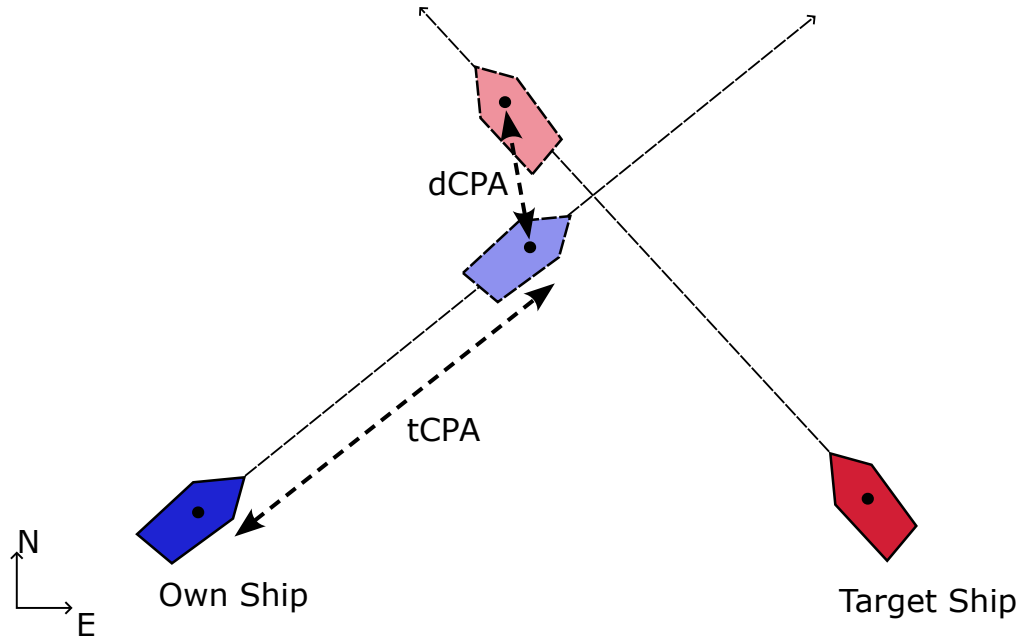


Figure 5: Visualizing dCPA and tCPA.

(a)(i) *Where one of two vessels is to keep out of the way the other shall keep her course and speed.*

(b) *When, from any cause, the vessel required to keep her course and speed finds herself so close that collision cannot be avoided by the action of the give-way vessel alone, she shall take such action as will best aid to avoid collision.*

These rules are not easily explained to a layperson, and even less easily to a computer algorithm. To formulate the constraints that will enforce COLREGs compliance it is sensible to start by considering rule 7; is there any risk of collision between the OS and any other vessel? A common risk assessment tool is to calculate the distance at Closest Point of Approach (dCPA) and time to Closest Point of Approach (tCPA) between two vessels as shown by (Kufoalor et al. 2018) and Figure (5):

$$t_{AB}^{CPA} = \begin{cases} \frac{\mathbf{P}_{BA} \cdot \mathbf{V}_{A|B}}{||\mathbf{V}_{A|B}||^2} & \text{if } ||\mathbf{V}_{A|B}|| > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.27a}$$

$$d_{AB}^{CPA} = ||(\mathbf{P}_A + t_{AB}^{CPA}\mathbf{V}_A) - (\mathbf{P}_B + t_{AB}^{CPA}\mathbf{V}_B)|| \tag{2.27b}$$

where $\mathbf{V}_{A|B} = \mathbf{V}_A - \mathbf{V}_B$ with $\mathbf{V}_A$, $\mathbf{V}_B$, $\mathbf{P}_A$ and $\mathbf{P}_B$ being the respective velocities and positions of two given vessels A and B parameterized in NED. To determine if the OS and a given TS should be considered to be in an active situation, the calculated dCPA can be compared to some lower threshold limit. If the dCPA is below the set threshold

the next step is to assert which COLREGs rule is currently in effect for the OS, this is what will be called 'active COLREGs situation' for the rest of this thesis, and asserting which COLREGs is currently in effect will be called 'COLREGs classification'.

(TODO: denne delen av masteroppgaven er veldig lik fordypningsprosjekt...) There have been multiple studies on COLREGs classification, (Woerner 2016) lays out an algorithmic approach based on the relative bearings between the OS and a given TS. In this algorithm numerical values from the COLREGs rules are used as the as the criteria for determining which COLREGs sitation the OS finds itself in. The algorithm yields the expected results, but it's a bit opaque and hard to follow. (Tam and Bucknall 2010) suggests a similar approach formulated in a more natural language that is easier and more intuitive to follow. This method first considers the relative bearing from the TS to the OS:

$$\phi = \text{atan2}((E_{TS} - E), (N_{TS} - N)) - \chi \tag{2.28}$$

where (N_{TS}, E_{TS}) and (N, E) are the positions of the TS and OS respectively, and χ is the course of the OS. With the OS as a centerpiece, 4 sectors can be defined by angles offset from the OS's course, where the relative bearing ϕ deciding which sector the TS is in. Similarly the relative bearing from the TS to the OS can be used to determine COLREGs situation. See Figure (6) for a visualization.

With COLREGs situation classified the last step is to determine the constraints so that the OS behaves compliant with the rules. One method, which was written about in the author's fordypningsproject is to add circular regions as consrtaints tied to the position and heading of the TS in which the OS is in an active situation with. An example of what a singular constraint placed like this would look like can be seen in Figure (7). To achieve this the trajectory of the TS must be discretized with the same time step size as the OS. At each control interval in the NLP, using the known values for the heading and position of the TS at that instance ψ_{TS_k}, (N_{TS_k}, E_{TS_k}), calculate an appropriate constraint origin:

$$\mathbf{o_{dc}} = \begin{bmatrix} N_{TS_k} & E_{TS_k} \end{bmatrix} + H * \begin{bmatrix} \cos(\phi_c) \\ sin(\phi_c) \end{bmatrix} \tag{2.29}$$

where H is the desired distance from the center of the TS to the constraint origin, and ϕ_c is the desired relative bearing from the TS to the constraint origin. The constraints

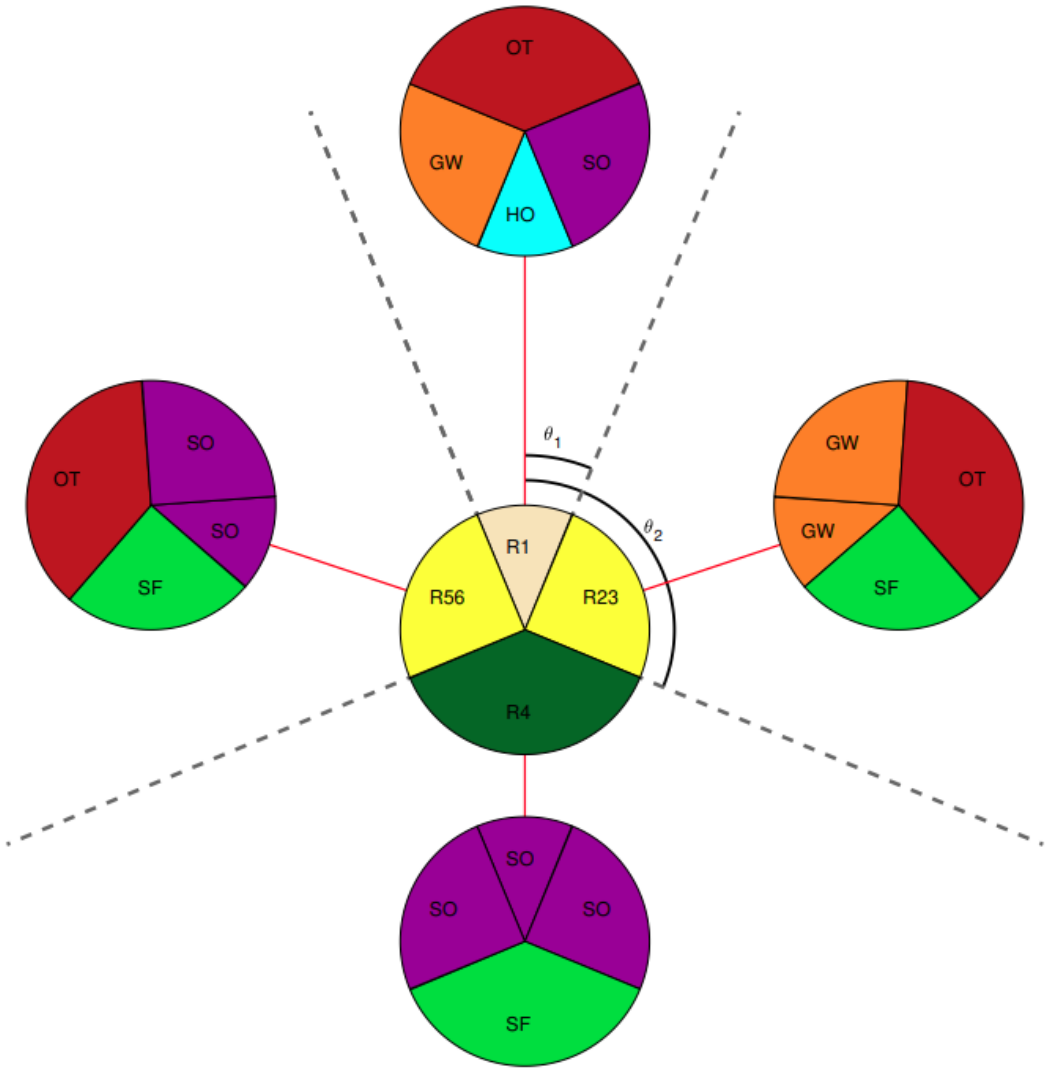


Figure 6: COLREGs classifcation; with OS in the center we can place the TS in one of four regions. Similarly the relative bearing from TS to OS can be assigned regions with region 1 pointed directly at the OS and the rest following in a clockwise rotation. Courtesy of Emil Thyri.

are added to $\mathbf{g}(\omega)$ the following way:

$$\mathbf{g}(\omega) = \begin{bmatrix} \vdots \\ ||\mathbf{X}_k - \mathbf{o_{dc}}|| \\ \vdots \end{bmatrix} \tag{2.30}$$

where $\mathbf{X}_k$ are the north and east positions in the decision variables $\boldsymbol{\omega}$. The square root of the lower bounds value for $\mathbf{g}(\omega)$ denotes the radius of the circle constructed by the consrtraint function, the upper bound value should be infinite.

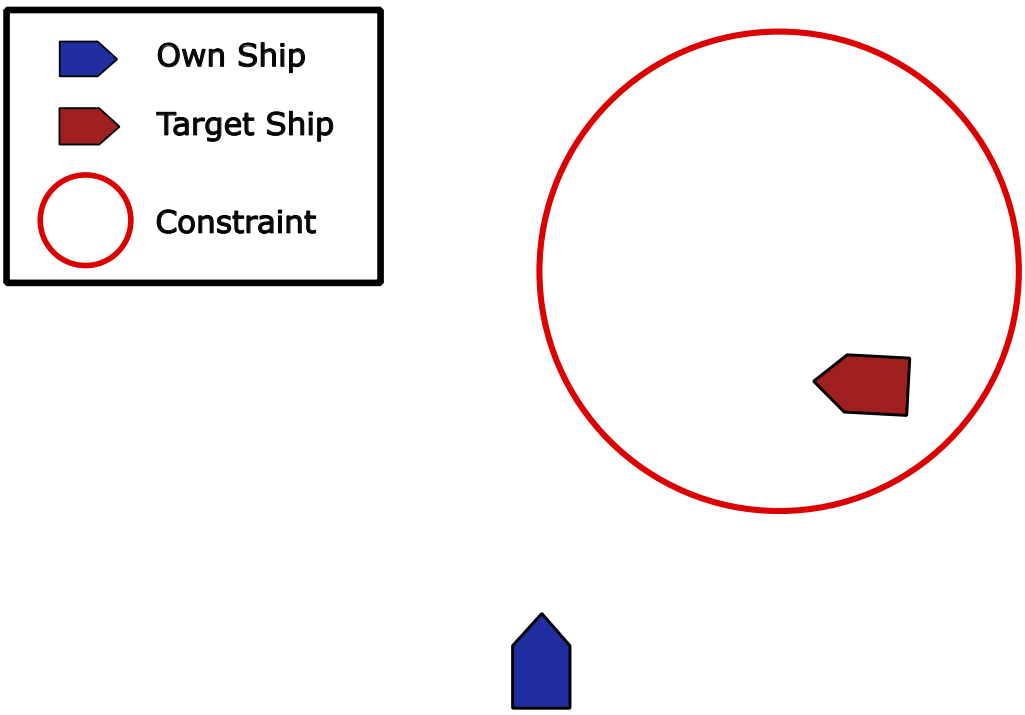


Figure 7: Example of a single placed constraint based on the position, heading, and COLREGs classification. Depicted would be a suggested placement for a Give way situation.

2.4 Target Ship Prediction

The apparent COLREGs compliance of the trajectory planner and collision avoidance algorithm can only be as good as it's ability to infer intent and predict the trajectories of other TSs. Infering intent and tracking other TS is an important task for human navigators, and a key part of collision avoidance. An ASV might have the instruments required to achieve full spatial and situational awareness, but it's ability to fully utilize these instruments is often underdeveloped.

(TODO: BRIDGE THIS GAP.)

To adress the issue of intent infering, (Cho et al. 2018) proposes a method that provides a decision-making procedure for safe navigation by predicting the maneuvering intent of TSs. In their work, a graphical model is constructed to infer intent by combining an intent model with a dynamic model. Each action of the TS influence the ship's assigned maneuvering intent probability, which denotes a probability that the ship is going to be compliant or non-compliant w.r.t COLREGs. In another study, (Schöller et al. 2021) attempts to predict the trajectory of TSs via an estimation scheme consisting of a Long Short-term Memory model in a Generative Adverserial Network configuration. The estimation scheme is backed by historical AIS data and outputs a probabilistic heat map of the future trajectory of TSs.

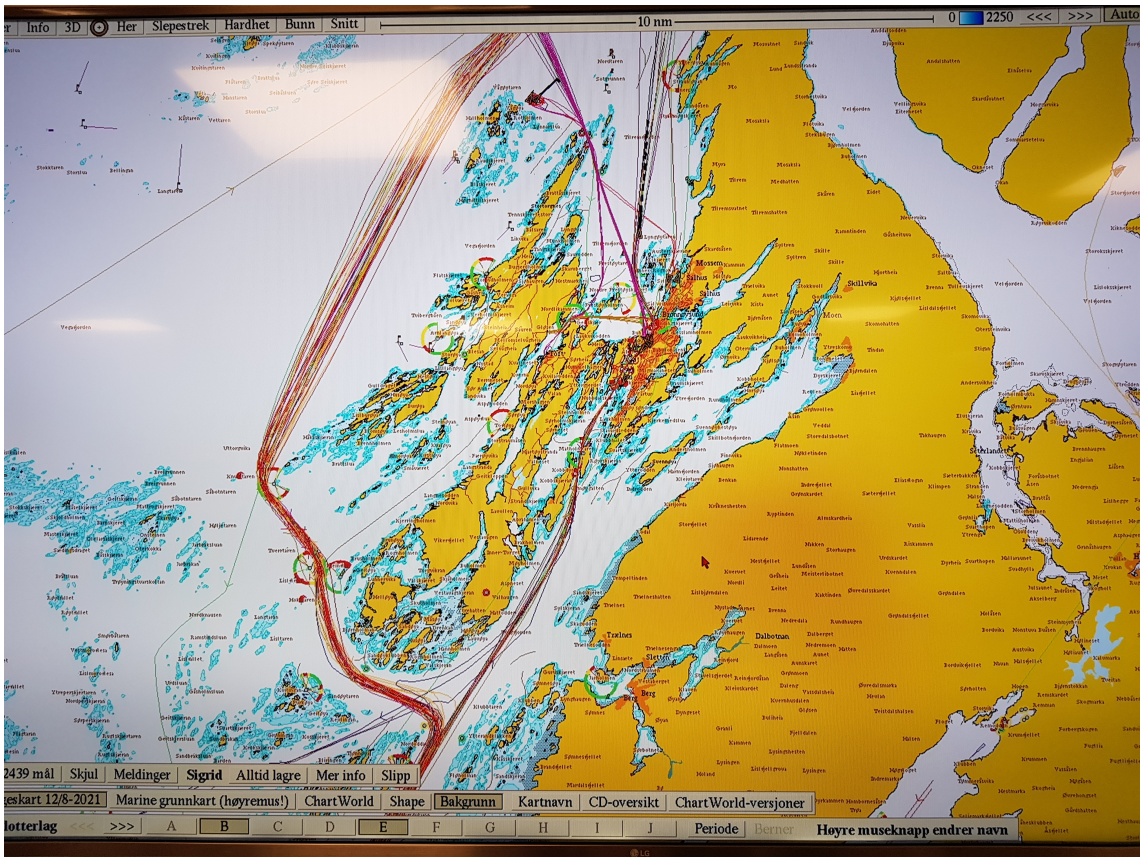


Figure 8: Photo of a typical ENC, here we can see the lines formed by saving AIS positional data over time. Image courtesy of Olex AS.

(TODO: BRIDGE THIS GAP.)

In a similar sense, (Zhang et al. 2021) notes in their survey how massive AIS data can be used for global route optimization by creating set waypoints based on the information extracted from massive AIS data. This AIS data can easily be compiled by storing the positional data of AIS transpoders over a period of time, as see in Figure (8), which is the result of saving data for about three days.

This thesis will make no attempts at implementing a prediction method for tracking and infering the intent of TSs. However it will operate under the assumption that the technology for more accurately predicting the trajectories of TSs is coming, and one of the key points of study for this thesis is the question of how improved prediction capacity will affect the behaviour of the proposed trajectory planning algorithm. In this thesis, the prediction capabilities afforded to the algorithm will be one of two options.

1. 'simple prediction' which is simply assuming that TS will maintain a steady speed and course over ground, a common method as noted by (Huang et al. 2020) in their review.

2. 'full prediction' which is some nebulous futuristic interaction and intent aware prediction method that is able to accurately predict the future trajectory of TS.

(TODO: siste avsnitt passer kanskje bedre i discussion.)

Even if the 'full prediction' capabilities are a far off possibility, an intermediate step could be extending the information sent via common AIS tde auto-navigation data or other details on the vessels's intended near-future maneuvers, perhaps a list of waypoints and the temporal constraints for reaching them or something similar. Having information is always better than predicting or infering intent, any aiding data that can assist in correctiransponders, as these are becoming more and more common in smaller vessel, and mandatory in bigger ones. The extended AIS data could inclung the predictions made by a computer would massively benefit the development of a 'full prediction' capable algorithm.

3 Trajectory Planner

This chapter presents a step-by-step walkthrough of the developed trajectory planning and collision avoidance algorithm, and explains some of the design decisions that were made during development. Additionally the chapter will include som analyzis of problems that arose during development, and the implementationts that were made to overcome them. First the general dataflow of the algorithm is presented so that an intuition can be gained as for how the individual parts of the algorithm are connected. Secondly each step of the algorithm is presented in the order of execution from top to bottom. Lastly a brief look at how the output from the algorithm is put to use.

The author wants to stress that while the trajectory planner is presented as a finished product, the algorithm was in active development until 12 days before the thesis deadline. The code will have bugs, there will be unreadable spaghetti code, and at times logical errors. The code should have be delivered as an attachment/appendix, or it can be found on github at: (TODO: LINK).

(TODO: enten i dette kapittelet eller i discussion må det skrives om tallverdi valg på funksjoner som dcpa grense etc.) (TODO: bør vel også nevne at koden er skrevet i MATLAB? er ikke det mest brukte kodespråket i verden).

3.1 Dataflow

(TODO: eller Overview?) The core of the design is to construct a path following trajectory that is simultaneously able to comply with COLREGs and avoid getting stuck on terrain or other obstacles. The dataflow of the algorithm is depicted in Figure 9, to avoid clutter the diagram does not include every subfunction and minor detail, it's a representative diagram, not a blueprint. On the left we begin with a higher level system, the algorithm relies on getting information about it's own vessel and information about other ships, in the diagram called "tracks". Additionally static obstacles and miscellaneous other settings for both debugging and behaviour tuning is to be supplied from said higher level system.

The algorithm is designed so that full prediction is the assumed standard prediction level. For testing and debugging purposes the tracks structure can be modified to emulate the formfactor of a simple prediction level, in a real implementation the simplification should not be neccessary as the data in the tracks struct should already be in the correct

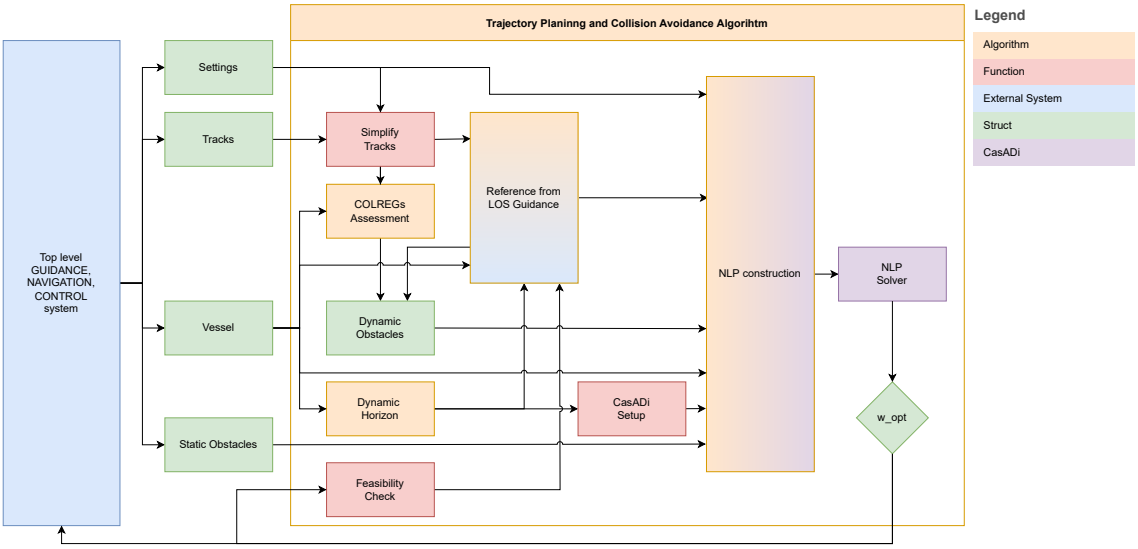


Figure 9: A simplified overview of the developed algorithm.

format for either prediction level. The data in the tracks struct is parsed through a COLREGs assessment algorithm to determine if any of the TSs need to be considered an active dynamic obstacle. If a TS is deemed to be active the tCPA and COLREGs sitation is determined and kept as a flag. The flag is stored in a persistent variable that the COLREGs assessment algorithm checks to avoid overwriting the classification of an active situation.

Next, the information stored in the Vessel struct, the struct dedicated to the OS, is used to calculate the desired time horizon for this call's MPC. Horizon distance and discretization step length are then needed by the CasADi setup function to create the RK4 method that discretizes the vessel dynamics. On the very first call of the algorithm the feasibility check is skipped, because there is no previous trajctory to parse, on all subsequent calls of the function the previously calculated optimal trajectory is checked for infeasibility. Feasibility, time horizon and discretization step information is then used to generate a reference trajectory for the OS, a trajectory is also generated for all the TS in the tracks struct, which are used to place dynamic constraints later.

The last step of the setup is to initialize the NLP using the OS's initial conditions, which creates the first six decision variables of $\boldsymbol{\omega_0}$ and the first six elements of the constraint function $\mathbf{g}(\boldsymbol{\omega})$. Then the algorithm iterates through all the control intervals, NP, from $K = 0 : NP - 1$, as decided by discretization step length and time horizon, constructing the NLP piece by piece in the following order:

(TODO: Denne kan kanskje gjøres om til sånn fin algorithm type)

First three new decision variables $\boldsymbol{\tau_k}$ are made, the appropriate reference states are

extracted from the reference trajectory, making sure that the heading reference doesn't wrap the wrong way about $[0, 2\pi]$. Then the discretized dynamics are used to integrate one control interval forward using $\boldsymbol{\omega_0}$, $\boldsymbol{\tau_k}$, and the reference values. Six new decision variables, $\boldsymbol{\omega_k + 1}$, and their shooting constraints are made. Lastly dynamic and static obstacles are placed in $\mathbf{g}$ and k is incremented by one.

After the NLP is constructed the initial guess for $\boldsymbol{\omega}$ is replaced by the previous optimal trajectory if it was feasible. When the NLP is solved the time it took is recorded, and if it didn't take too long to solve we save the solution to be used as the previous optimal trajectory for the next time the algorithm runs. Some plots for debugging can then be made if desired, and the resulting optimal states for the next control interval returned as the output of the algorithm.

3.2 Setup

The setup is all of the code that is run from when the trajectory planning algorithm is called, up until the construction of the NLP. This is the modular part of the code, where functionality can easily be added or removed without having to refactor the rest of the algorithm. Anything from new and improved situational awareness models to reference trajectory creation and COLREGs compliance ideas slot right into the setup. Of course these mentioned systems could just as well exist outside of the trajectory planning algorithm, but for this thesis it's designed as an all-in-one package.

In the current version of the trajectory planning algorithm there are four major and five minor tasks to get through in the setup. First the major tasks:

- Conduct COLREGs assessment.
- Calcualte dynamic horizon.
- Run CasADi initialization.
- Generate reference trajectories for OS and TSs

and the 5 minor tasks:

- Declare and initialize persistent variables.
- Initialize dynamic obstacles, and simplify tracks if needed.

- Conduct feasibility check on the previous optimal trajectory if it exists.
- Sanitize initial position to make sure there are no problems with the heading.
- fetch static obstacles, this is only a task because of how the MATLAB simulator is set up.

Persistent Variables

(TODO: vet ikke helt hvorfor jeg går så grundig til verks...) In MATLAB, persistent variables are stored in memory when a script has terminated, and is loaded back as they were the next time the script runs. This can be used to create rudimentary state machines, or to check the outcome of the previous iteration for anomalies. Persistent variables are declared without an initialization, and the method for initializing them without overwriting every time is shown in Algorithm [1].

Algorithm 1 Function: Initalize persistent variable

```
persistent Var
if isempty(Var) then
    Var ← Initial value
end if
```

Because persistent variables persist it is advised to manually clear the script when starting the MATLAB simulator, otherwise residual perisstent variables from unrelated scenario files can cause debugging problems. In the algorithm there are seven persistent variables. Two are used to store the optimal trajectory between iterations. One to store the discretized function F, so it doesn't have to be remade every time the algorithm runs. one for storing COLREGs flag to act as a state machine. One for storing a variable called firsttime, used to execute code only the first time the algorithm is called. One that enables or disables obstacles, only left intact as a debugging tool. And the last one to store the previous iterations heading reference, which is used to prevent a Wrap-to-2-Pi problem. (TODO: SKriv om hva Wrap-to-2-pi problem er for noe).

The reason there are two persistent variables to store the previous optimal trajectory is poor planning: the original optimal trajectory was dumped if it took too long to solve the NLP. But later when I implemented a feasiblity check this would cause issues since feasiblity and time-to-solve for the NLP are not neccessarily linked. Saving the previous optimal trajectory twice so that one is use for feasiblity check, while the other is the initial guess substitution candidate, was a very quick hack solution that worked well enough that it survived until the final version of the code.

Simplify Prediction

This part of the setup is only neccessary in simulations, the idea is to prepare the tracks struct so that it can be parsed by the COLREGs assessment algorithm regardless of desired prediction level. Since it is much easire to truncate excess waypoints and 'step down' from full prediction to simple, than the other way around, the algorithm was developed with full prediction level as the standard. If it's desired to step down to a simple prediction level the tracks simply need to be parsed though algorithm [2].

Algorithm 2 Function: Simplify TS prediction

for $i = 1$:size(tracks,2) **do**
  if simple **then**
    tracks(i).wp(1:2) $\leftarrow$ tracks(i).eta(1:2)
    tracks(i).wp(3:4) $\leftarrow$ tracks(i).eta(1:2)
      + 1 nmi * [cos(tracks(i).eta(3)) , sin(tracks(i).eta(3))]$^{\mathrm{T}}$
    tracks(i).wp = [tracks(i).wp(1:2)$^{\mathrm{T}}$, tracks(i).wp(3:4)$^{\mathrm{T}}$]
    tracks(i).current_wp $\leftarrow$ 1
  end if
end for

COLREGs assessment

The COLREGs assessment function solves two problems; figuring out ifwhen a TS vessel will be in close enough proximity that evasive maneuvers might be considered, and deciding which of the COLREGs rules will apply for the encounter. The design idea is to first find what the distance at closest point (dCPA) of approach with the TS is, and then time until cloest point of approach (tCPA) occurs. If both dCPA and tCPA values are under a set threshold we consider the encounter an active event and run the COLREGs situation assessment shown by Thyri and Breivik 2022.

Finding the dCPA and tCPA between two vessels with constant velocity and course is easily done with a formula as shown by(eller in?) Kufoalor et al. 2018.

TODO: Ikke ferdig

to get a list of all dCPA and tCPAs between two agents, as well as the corresponding positions of both agents as they are when the euqations (2.27) are used. getCPAlist

Dynamic Horizon

The dynamic horizon algorithm could also theoretically calculate a desired discretization step length, but in the current version of the algorithm that functionality is hardcoded.

CasADi setup

Feasibility check

Reference from LOS

3.3 NLP Construction and Solver

(TODO: ??? Dette kan ikke være bare et kapittel.)

3.3.1 NLP initialization

3.3.2 Integration step

3.3.3 Dynamic Obstacles constraints

3.3.4 Static Obstacles constraints

3.3.5 Solver

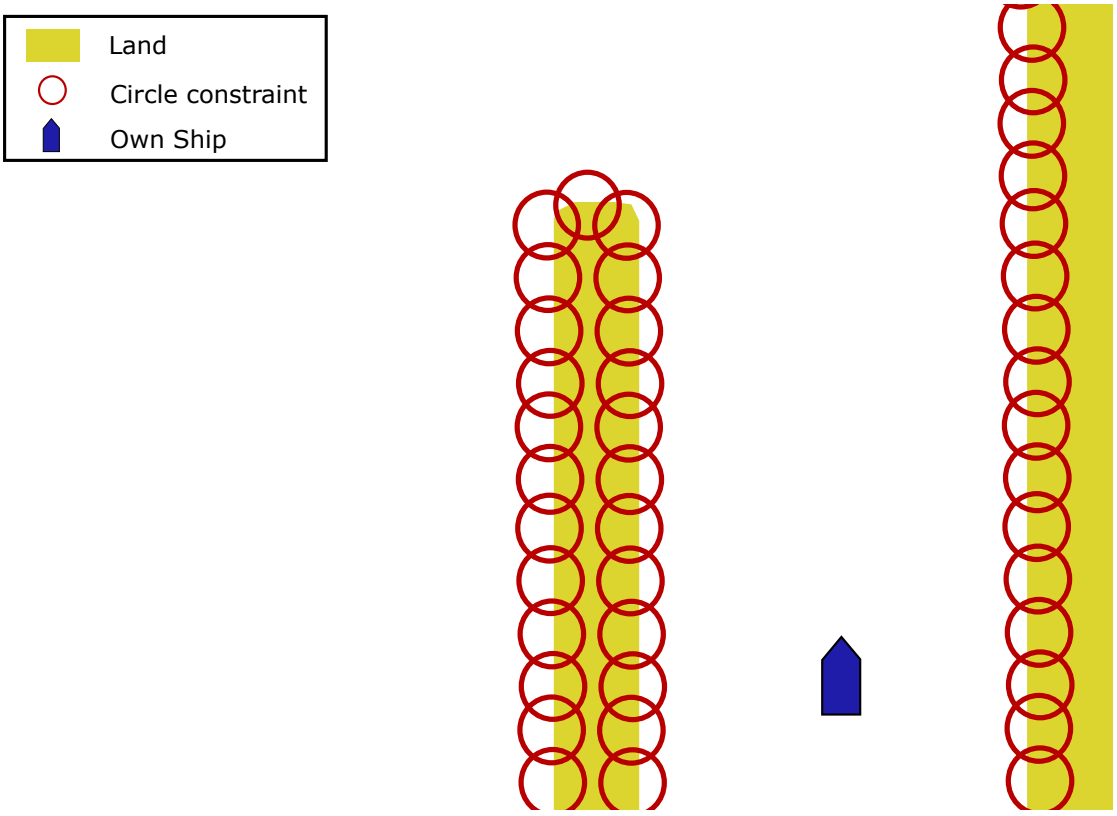


Figure 10: TODO: SKRIV OG REFERER. Naiv approach 1

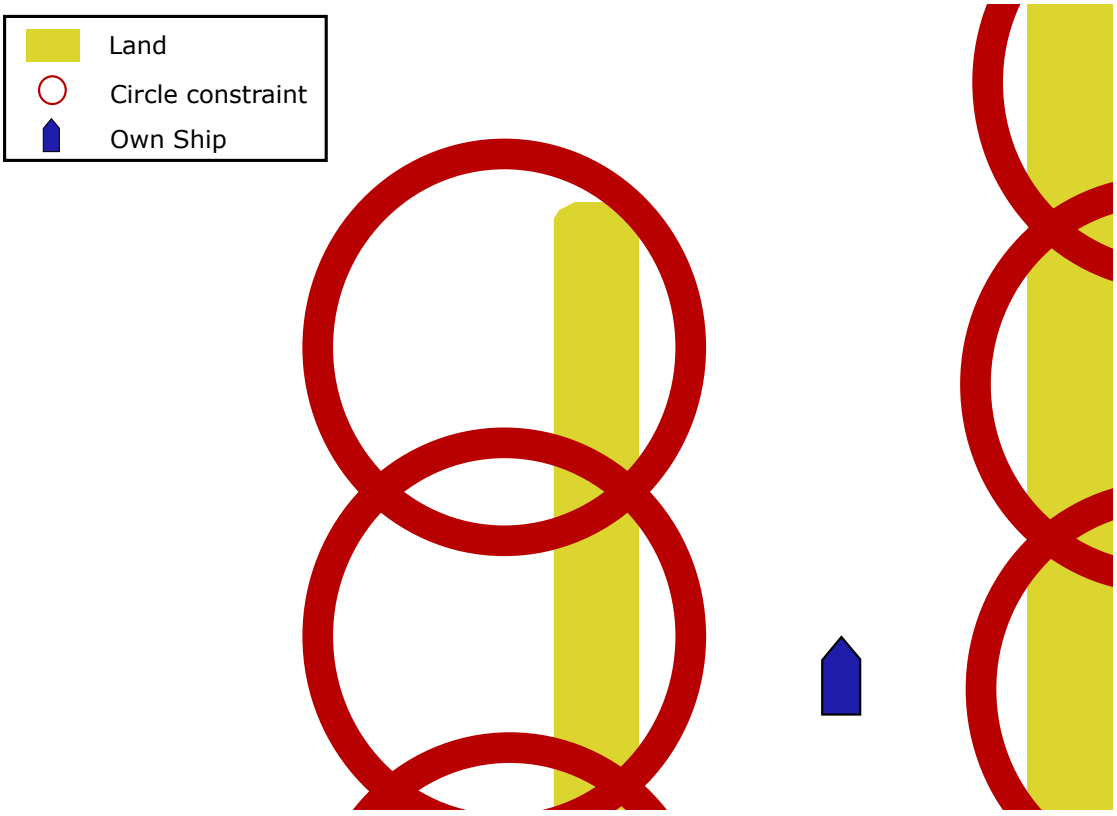


Figure 11: TODO: SKRIV OG REFERER. Naiv approach 2

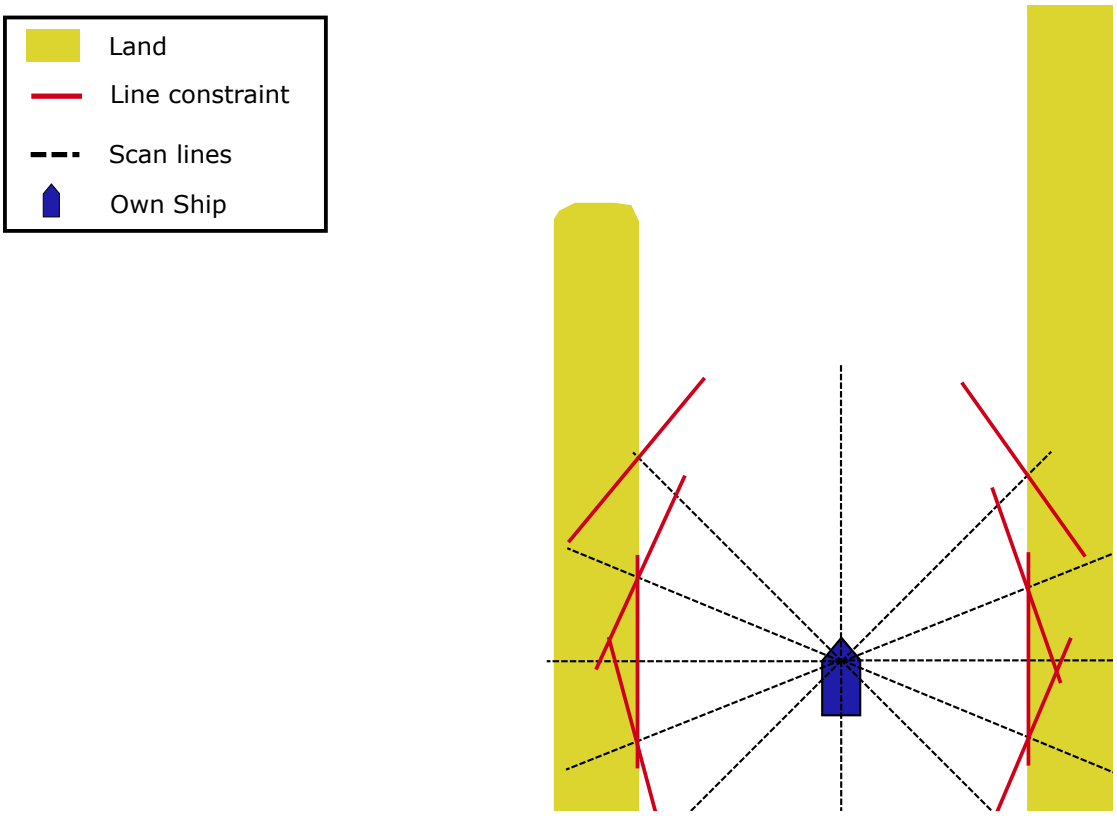


Figure 12: TODO: SKRIV OG REFERER. Convex free set

4 Simulation Results

To test the capabilities of the trjectory planning algorithm it is useful to conduct simulations of various scenarios. With a simulator it is possible to cover a wide assortment of scenarios in a timely fashion, this helps explore the full range of the algorithm's behaviour without having to conduct time consuming full scale tests. NTNU also has a full-scale functional prototype of an autonomous ferry that could be used to conduct real life tests. However during the period of working on this thesis the ferry was out of commission due to a thruster failure. The MATLAB simulator emlpoyed for this thesis was developed by Emil Thyri and is used with permission. In this chapter the results are presented with figures to show the developemnt of the scenario over time, in addition to these figures there exists a youtube video compiling all the results in video format, the video can be found as an attachment to the thesis, or by following this link: (TODO: sett in link).

All the simulations are conducted under the assumption that the OS has perfect vision for spotting and tracking dynamic obstacles. disturbances are also largely ignored, the simulation features no current or wind induced sideslip, crab angle is also not considered.

4.1 Scenario Overview

The scnearios used for this thesis are constructed to test both trajectory planning and collision avoidance capabilities through a combination of both trivial and complex situations. The scenarios are also designed so that behaviour differences between full and simple TS prediction can be observed. Any time we encounter a TS that maintains a steady course and velocity there will not be any observable difference, therefor most of the scenarios are constructed so that encounters occur when ships are turning. The first set of scenarios are simple situations to establish baseline behaviour in the various COLREGs situations. In these scenarios there are only two agents and there are mostly no meaningful differences observed between simle and full prediction of TSs. The second set of scenarios are more complex by featuring more agents and longer paths to follow. These scenarios often feature multiple COLREGs situations that can even overlap, additionally TSs will not be considerate of the OS and will exhibit reckless behaviour in order to test a sort of worst case scenario. The complex scenarios also incorporate static obstacles to show how the algorithm handles both types of obstacles at the same time.

Simple COLREGs Situations

These scenarios feature two agents, the OS and the TS, each entering a fully open space while maintaining a steady course and fixed speed. The agents then cross in manners as described by the COLREGs rules discussed in prior chapters.

Turning COLREGs Situations

Similar to the simple COLREGs situations these scenarios all feature two agents who enter a fully open space. The difference is as the name implies that these scenarios feature a turn by the TS. Shortly after both agents are in motion the TS will alter it's course, changing the COLREGs situation from one apparent situation to another.

Canals

This scenario features a set of canals that form a T-junction as well as a choke point on one of the junction points that restricts the traversable space. There are three agents present and they all meet roughly at the choke point, the scenario is set up so that the dynamic constraints of the TSs completely block the path of the OS if full prediction is

used.

Fjord

The fjord is construct as a miniature version of the Trondheimsfjord, this scenario is designed as a stress test of COLREGs situations. With multiple TSs crossing, turning and overtaking the OS simultaneously this scenario will show how the trajctory plannign algorithm differs with prediction level.

Helloya

The situation in this scenario is specifically modelled after a spot near Brønnøysund and is not an entirely uncommon situation when in transit along the coast of Norway. Traffic that wishes to avoid the narrow pass leading in to or out of Brønnøysund's will elect to take a wider path on the outside of the local archapeligo. The result is a path with a very prominent turn that is invisible at a glance, but very obvious to any experienced navigator. The simulation is conducted with the OS arriving from both the north and south direction with both full and simple prediction enabled.

Skjærgård With Traffic

Skjærgård is a Norwegian term for a section of ocean where there are many small islands and skerries, while the term translates to archapeligo a skjærgård is generally small in scale. This scenario puts a lot of stress on the trajectory planner which has to deal with both moving dynamic obstacles as well as the static obstacles that are sometimes blocking the reference path.

Skjærgård Without Traffic

A simpler version of the previous scnario, this time with no traffic but with more skerries near the reference path.

Miscellaneous

These scenarios are not meant to simulate any specific situation, rather these are meant to showcase quirks, features, and bugs encountered while developing and testing the al-

gorithm. While some of the problems shown here were taken care of and are no longer present in the current iteration of the algorithm they are nonetheless important to showcase and discuss.

4.2 Results

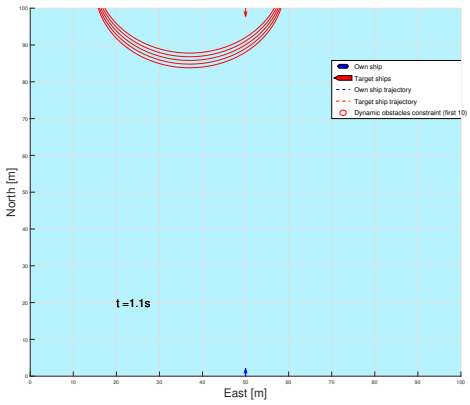
- 'Dårlig' resultat er fortsatt resultat
- Computational efficiency is also a topic
- First a brief discussion about each scenario result individually, taking a look at both the simple and full prediction level results.
- Then a closer examination of specific behaviours, problems, and observations that are not neccessary situation specific.
- Then a qualitative disucssion on the results as a whole, are theese the exected result? why or why not. etc.
- The scale is not uniform across all simulations, sometimes the boats are scaled up to make the figures easier to read.
- The results are accompanied by MALTAB figures, as well as a youtube video that compiles all the results into a video which sampled the simulations every second.

4.2.1 Simple Head On

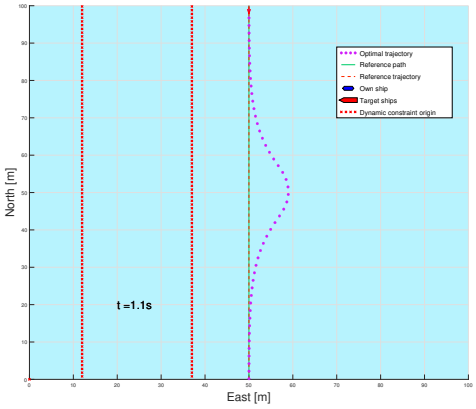
- Very straight forward result, behaviour is exactly as one would expect given the placement of the constraint.
- No difference between full and simple prediction because target ship holds steady course and velocity.
- This is the behaviour we can expect every time a target ship is met directly head on and there are no other disturbances.

4.2.2 Simple Give Way

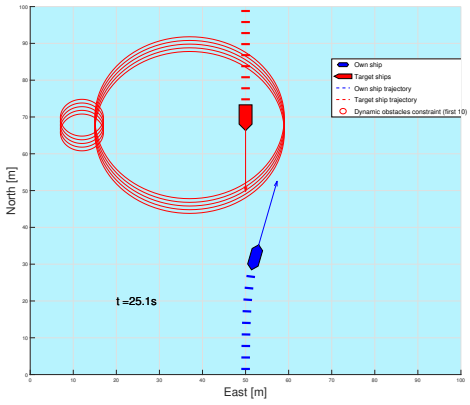
- It would be highly unusual to start using the trajectory planner when alredy this close to a situation



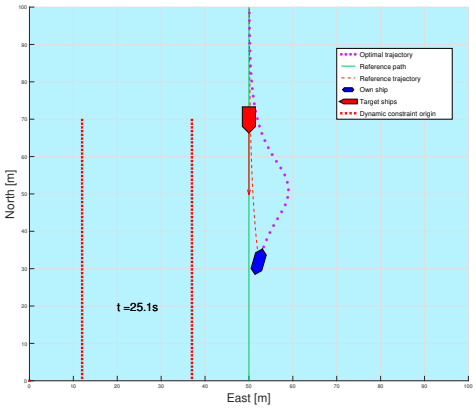
(a) caption



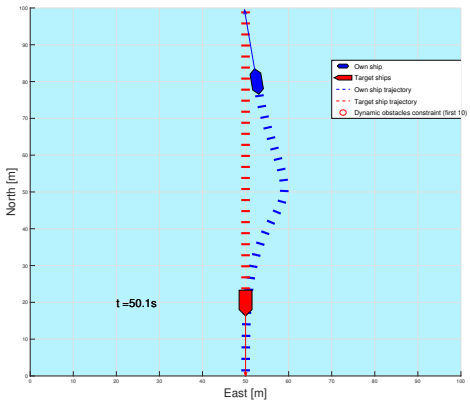
(b) mhm



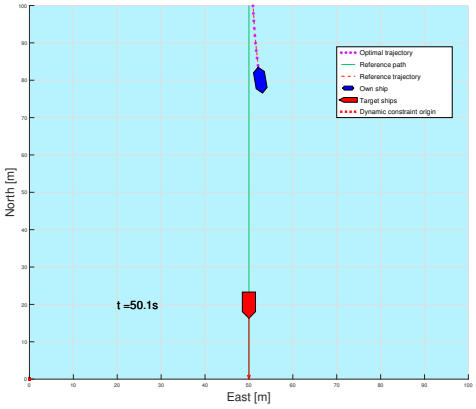
(c) caption



(d) mhm



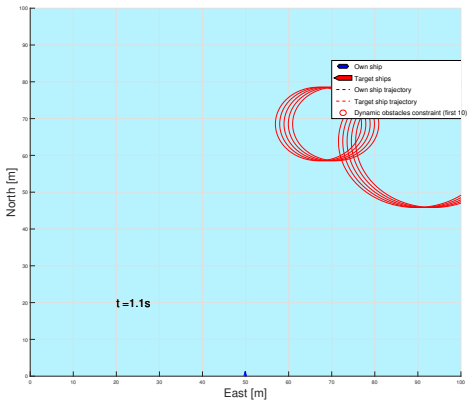
(e) caption



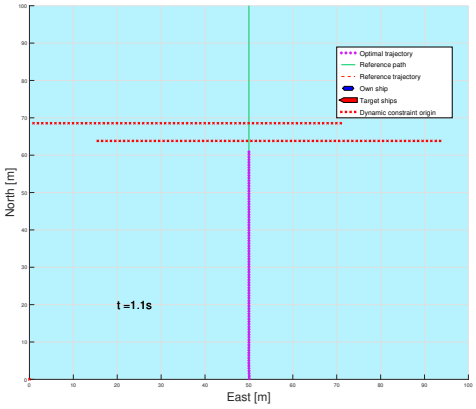
(f) mhm

Figure 13: Simple Head on situation. Result independent of prediction level.

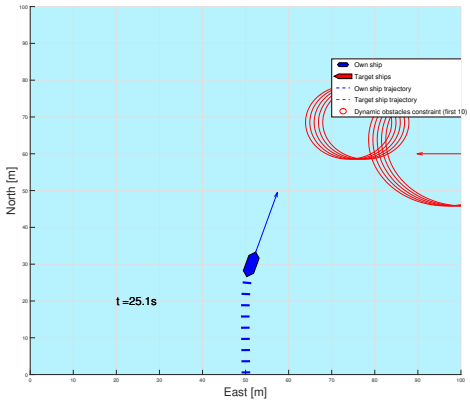
- This is reflected in the strange behaviour where our path is completely blocked when dynamic obstacle constrains are enabled.
- otherwise the behaviour ends up being exactly as expected considering the placements of the constraints.



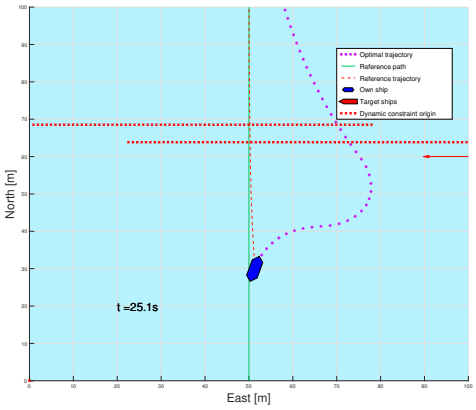
(a) caption



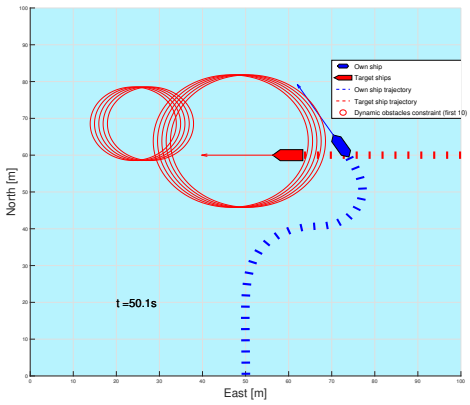
(b) mhm



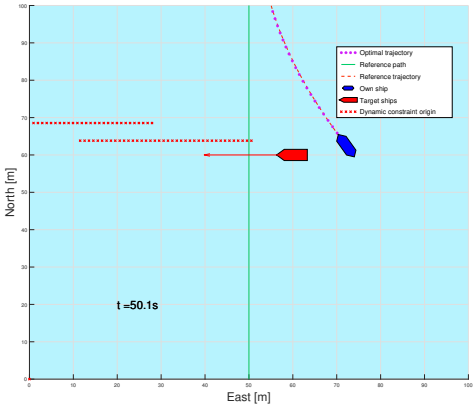
(c) caption



(d) mhm



(e) caption

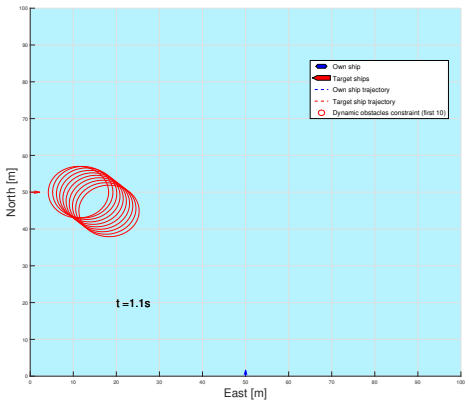


(f) mhm

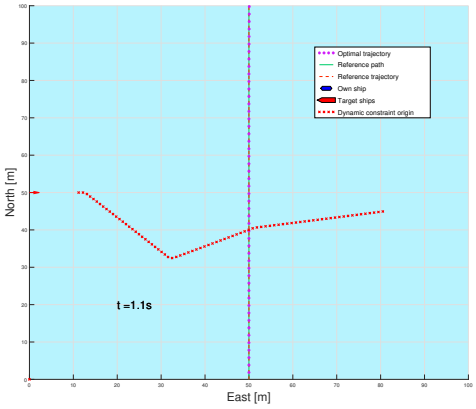
Figure 14: Simple Give Way situation. Result independent of prediction level.

4.2.3 Simple Stand On

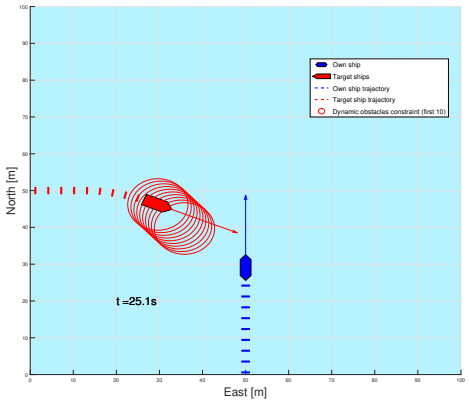
- This scenario assumes that the TS plays nice and attempts to follow the COLREGs rules.
- When using simple prediction the optimal path is pushed towads port side behind the TS,



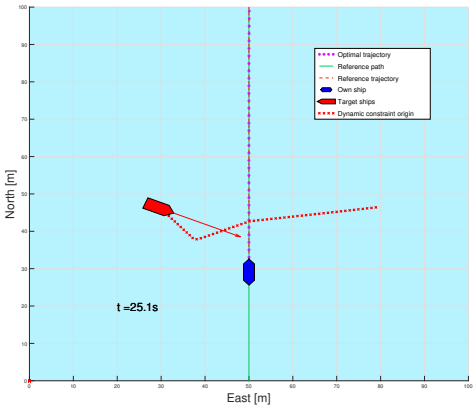
(a) caption



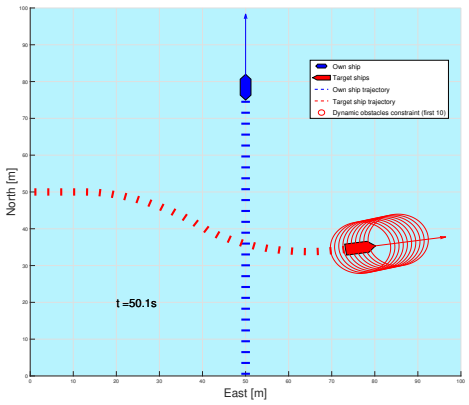
(b) mhm



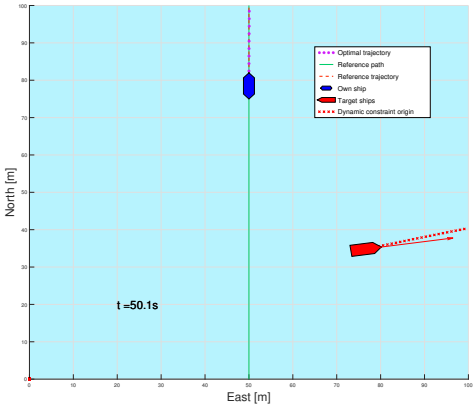
(c) caption



(d) mhm



(e) caption

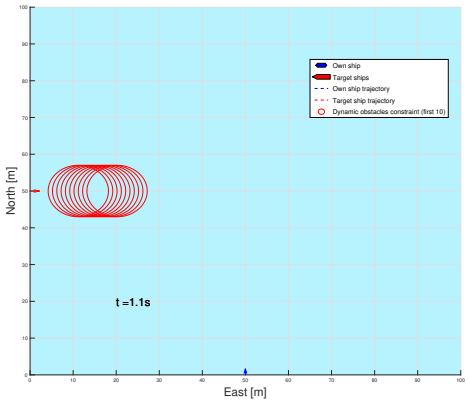


(f) mhm

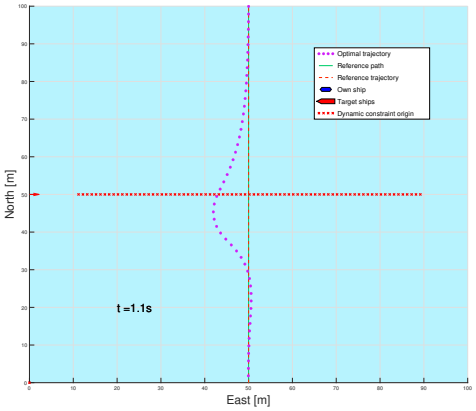
Figure 15: Simple stand on situation. Here shown with full prediction, OS correctly stands on.

and as the TS begins to turn the OS is dragged along by the movement of the constraints.

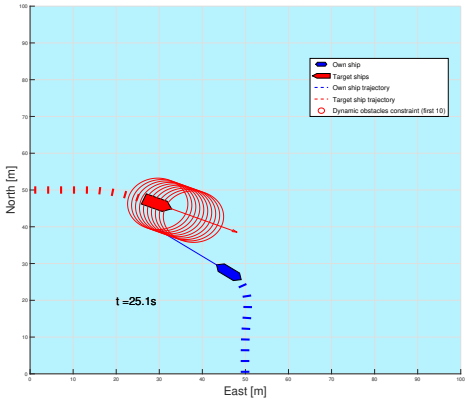
- This is actually quite unrealistic, there is no reason to assume the TS would continue to yield after observing the OS change course like this. It's also a bad result for simple prediction because the behaviour can cause confusion



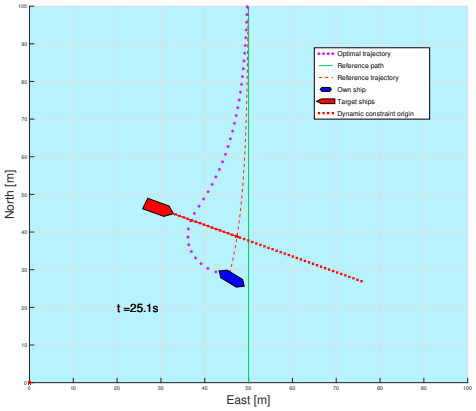
(a) caption



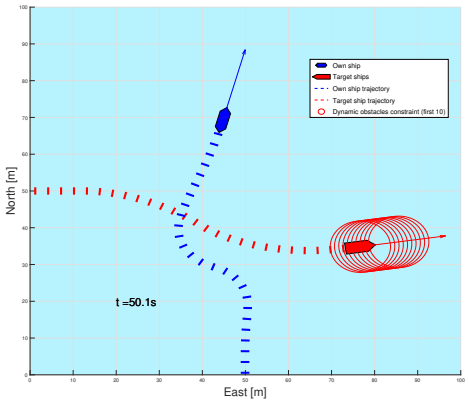
(b) mhm



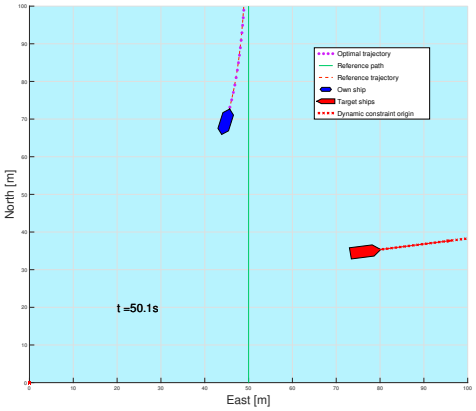
(c) caption



(d) mhm



(e) caption



(f) mhm

Figure 16: Simple stand on situation. Here shown with simple prediction, the OS can be observed to yield when it shouldn't.

for other navigator.

- The author realized late that the way prediction is done of the TS is not entirely consistent with the way prediction is handled internally in the algorithm. In the algorithm waypoints

are used to predict TSs trajectories, which is not neccessary
the exact same trajectory as the TS ends up following.

- In order to get the TS to comply with expected COLREGs behaviour a waypoint was placed some meters south of the OS position at the would be tCPA. This waypoint would obviously not exist in a normal transit situation and so
  this scenario is actually a bit of a cheat in regards to how prediction is argued for in prior chapters.

- The result is still interesting, for this scenario only we can exchange the advanced prediction result with a fully accurate prediction, and the result for simple prediction would hold for the current algorithm in all cases.

4.2.4 Turn Head On

- THIS SCENARIO SHOULD BE MIRRORED HORIZONTALLY TO INCREASE THE CHANCES OF AN OBSERVABLE DIFFERENCE BETWEEN PREDICTIONS.

- otherwise business as usual, turns aren't predicted quite right.

4.2.5 Turn Give Way

- Here we finally observe a big differene between full and simple prediction.

- due to how the situation plays out the OS is dragged along by the constraints of the TS.

- being dragged by constraints is not unique to this situation, and is one of the problems that can occur with this method of collision avoidance.

4.2.6 Turn Stand On

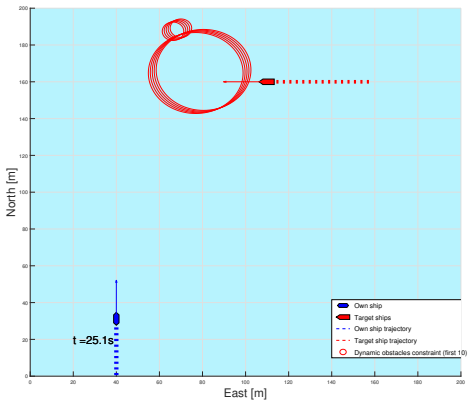
- Absolutely nothing happens here. move on.

4.2.7 Canals

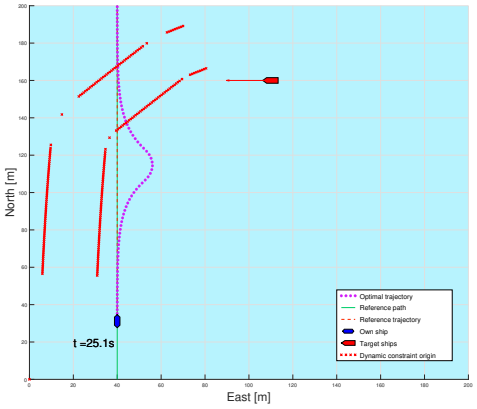
- blocked path

- feasibility check

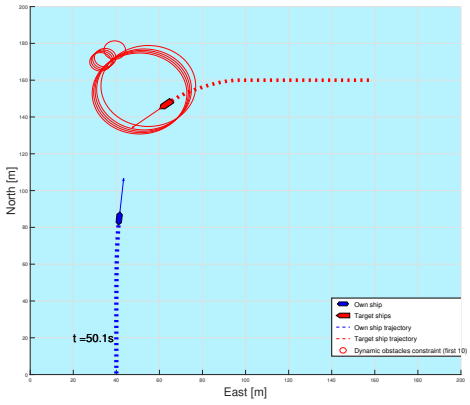
- forskjell mellom prediction metoder



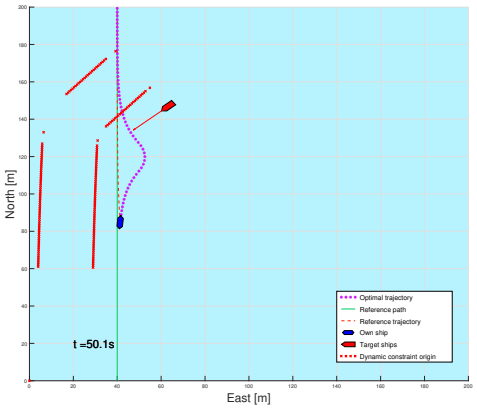
(a) caption



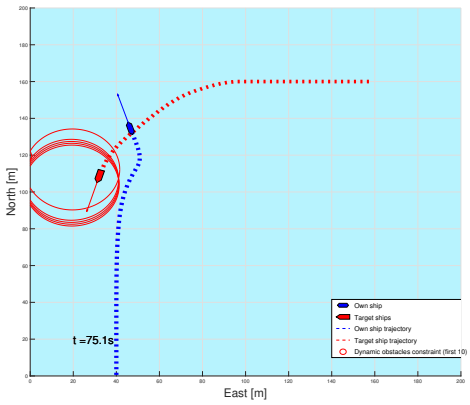
(b) mhm



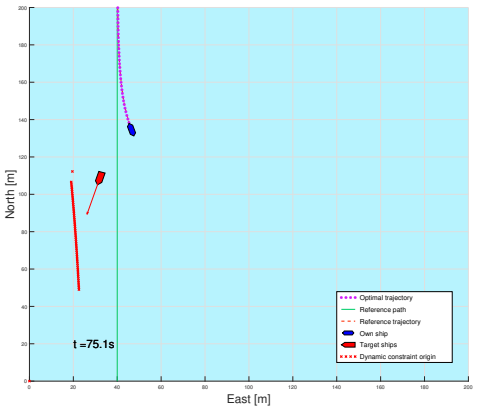
(c) caption



(d) mhm



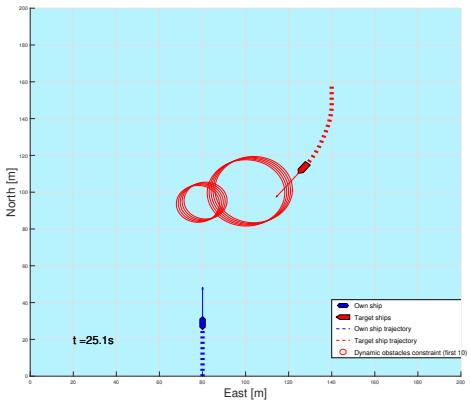
(e) caption



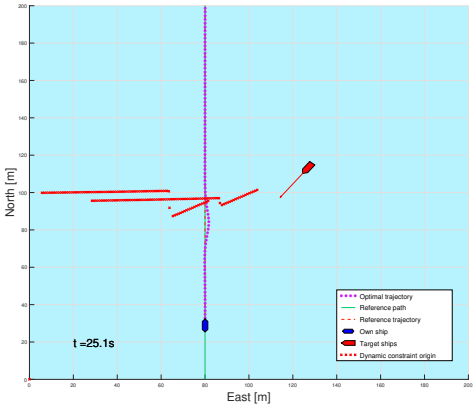
(f) mhm

Figure 17: Head on situation with a turn, Result for this were the same regardless of prediction level.

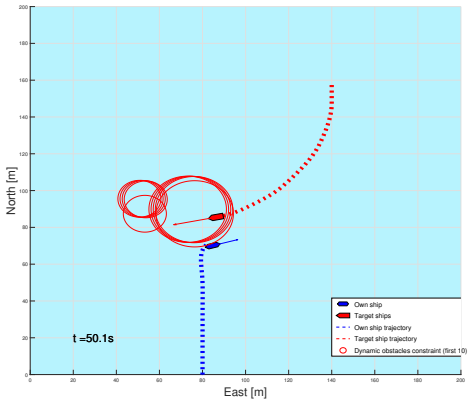
- When the path is blocked it takes a very very very long time to solve the NLP, users beware.
- Get to see static obstacles in effect.



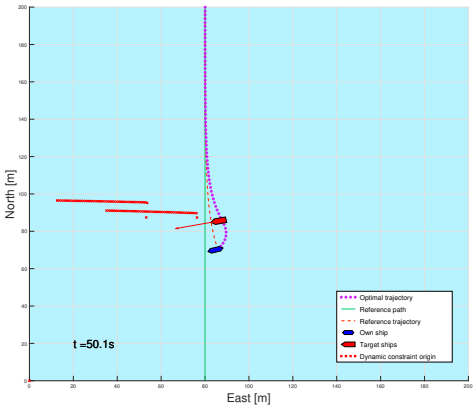
(a) caption



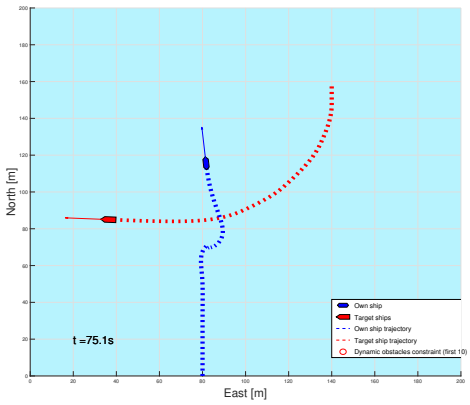
(b) mhm



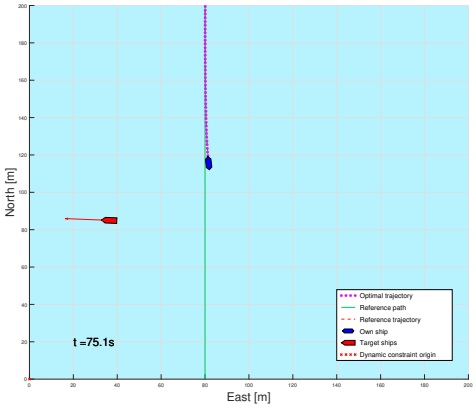
(c) caption



(d) mhm



(e) caption

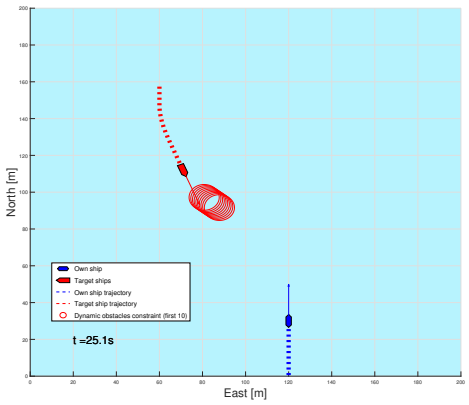


(f) mhm

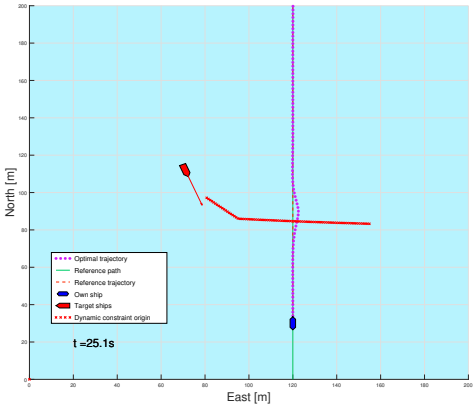
Figure 18: Give way with a turn, here with full prediction. Observe the OS not execting to have to yield until it's almost too late.

4.2.8 Trondheimsfjord

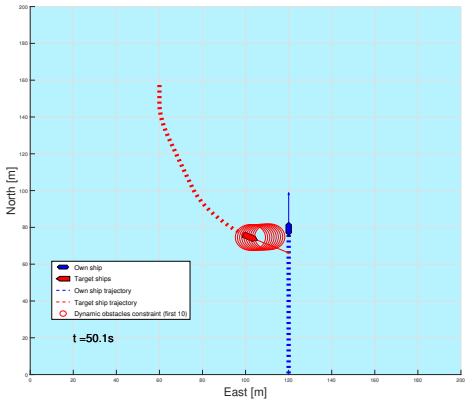
- COLREGs stress test
- Full prediction behaves much 'calmer', which is better in the author's opinion.



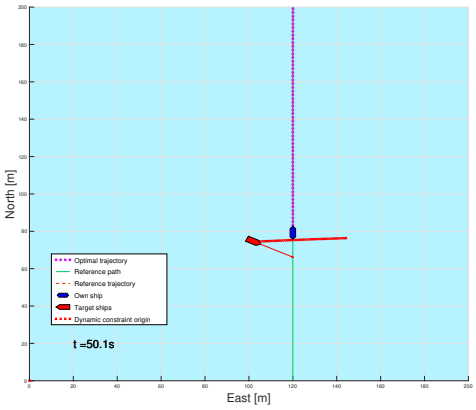
(a) caption



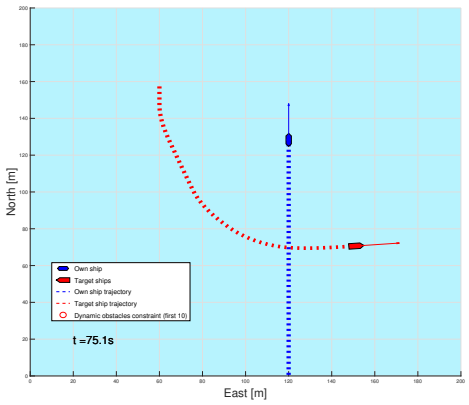
(b) mhm



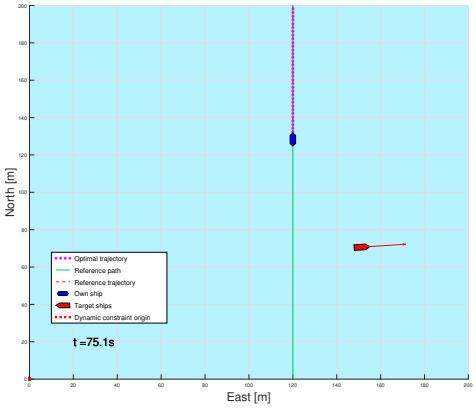
(c) caption



(d) mhm



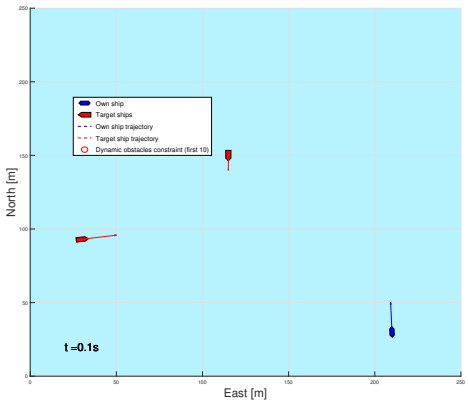
(e) caption



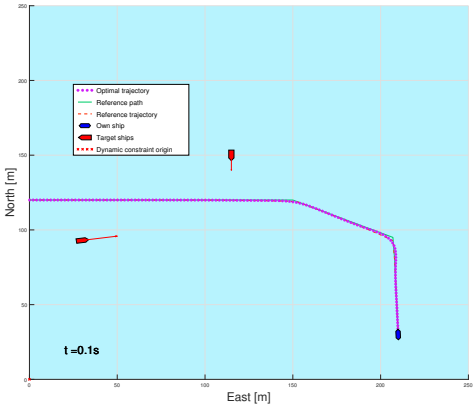
(f) mhm

Figure 19: Stand on situation with turn. Result independent of prediction level.

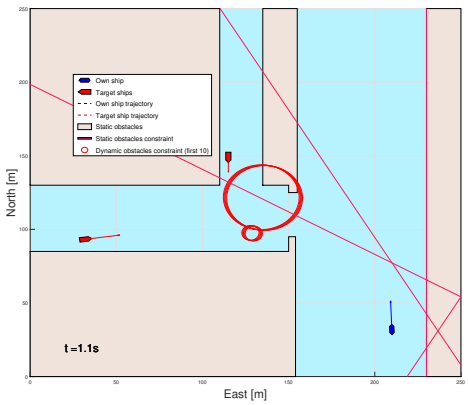
- Full prediction is also much more computationally efficient, for reasons that will be discussed later.



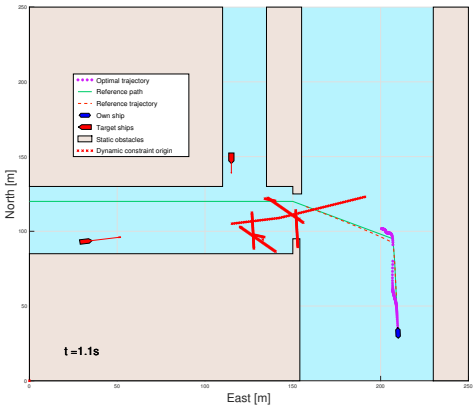
(a) caption



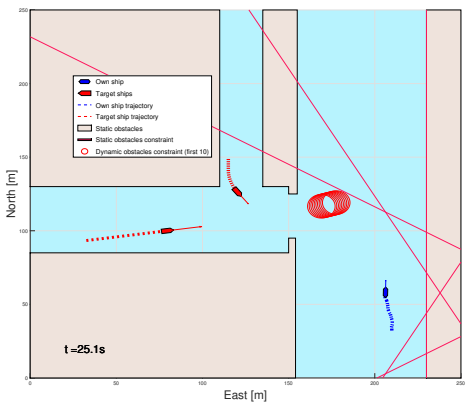
(b) mhm



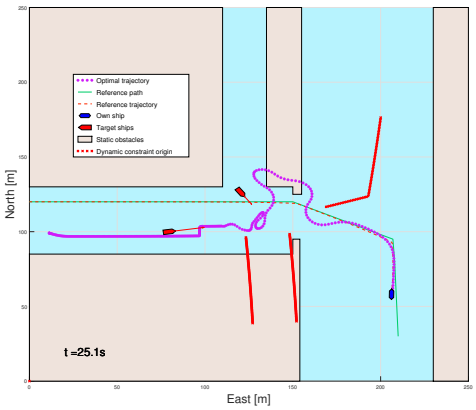
(c) caption



(d) mhm



(e) caption



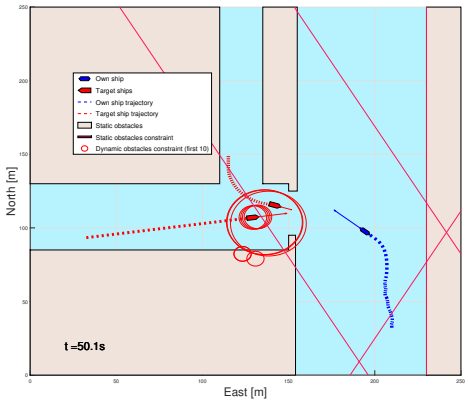
(f) mhm

4.2.9 Helløya

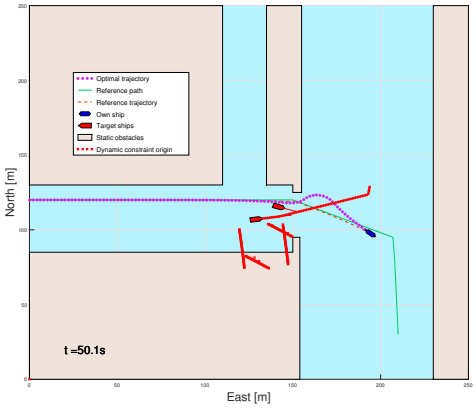
- Both simple and full prediction actually navigate the 'invisible' turn quite well
- even in the reverse direction it's not really a problem, though simple prediction cuts the turn, which could be considered bad.

- when being overtaken the full prediction exhibits much better behaviour.
- This is one of the scenarios where the scale is very exaggerated.

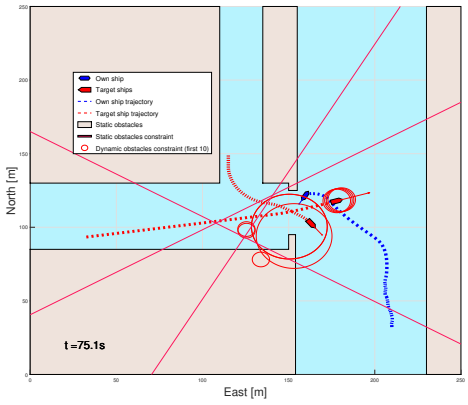
(TODO: Skriv og figurer)



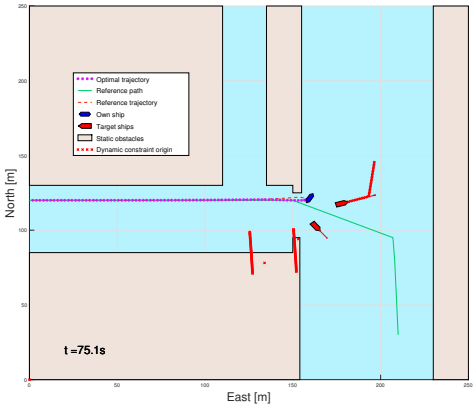
(g) caption



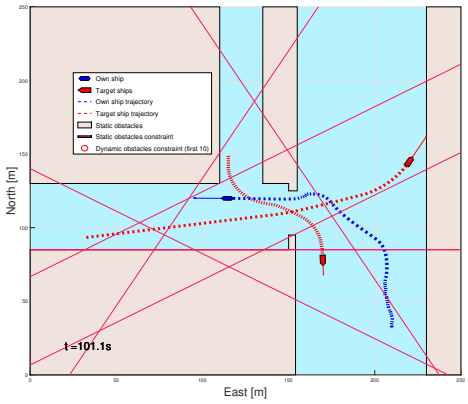
(h) mhm



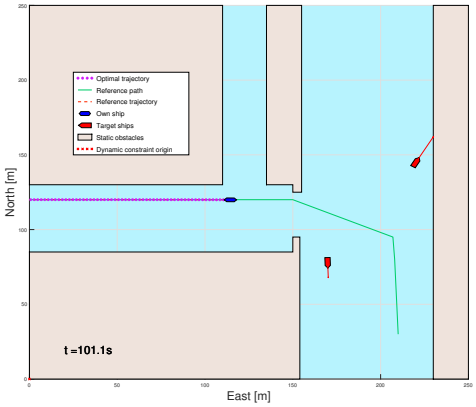
(i) caption



(j) mhm

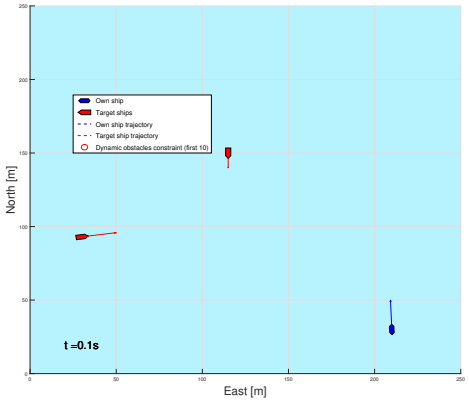


(k) caption

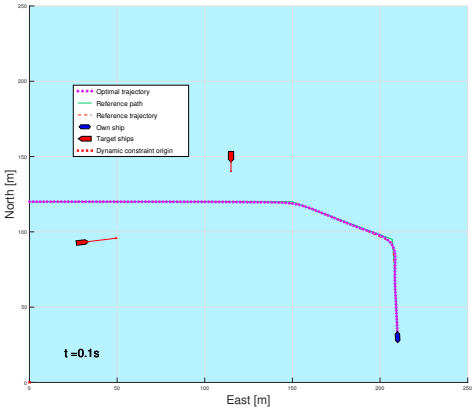


(l) mhm

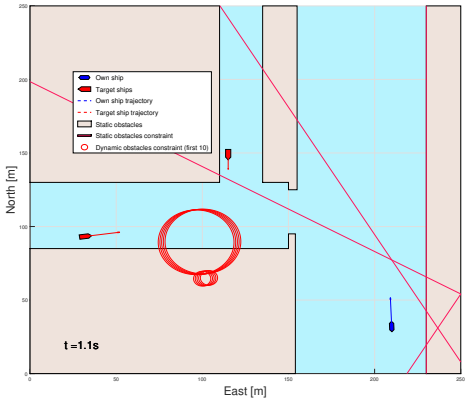
Figure 20: Canals situation. Here shown with full prediction.



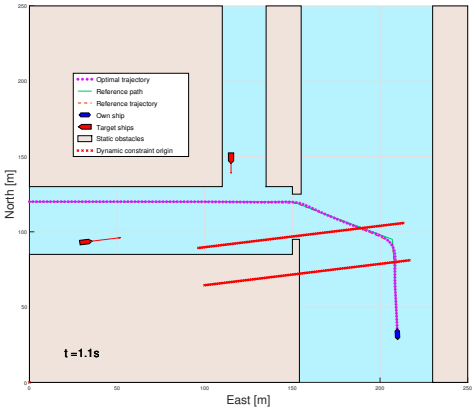
(a) caption



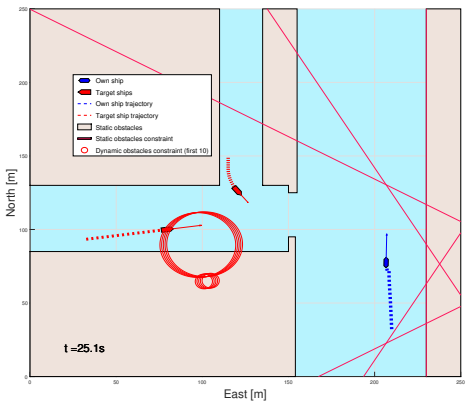
(b) mhm



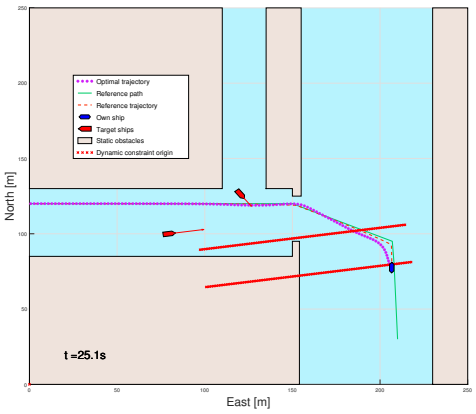
(c) caption



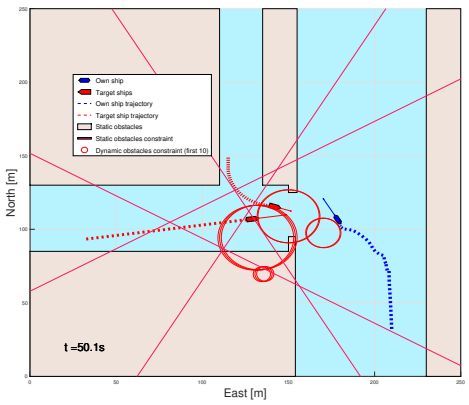
(d) mhm



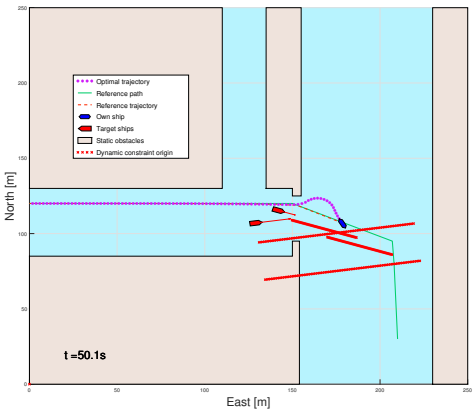
(e) caption



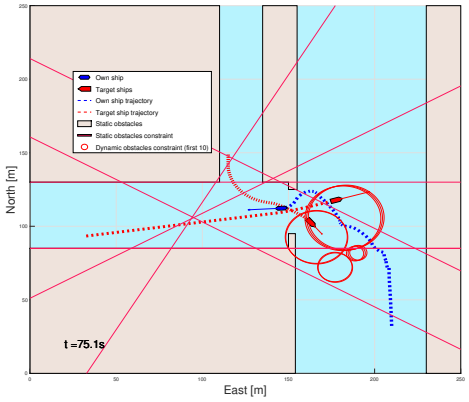
(f) mhm



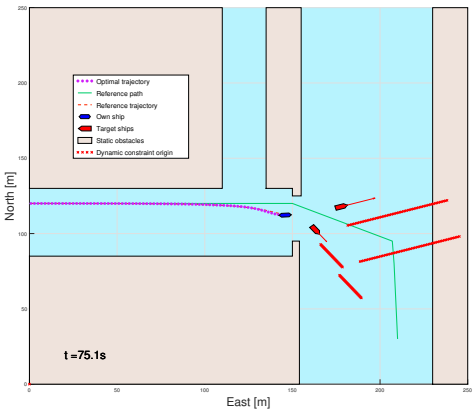
(g) dsadada



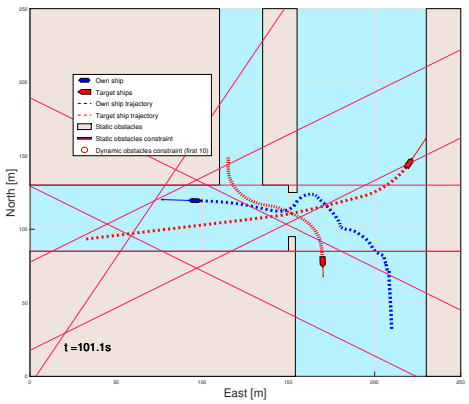
(h) mhm



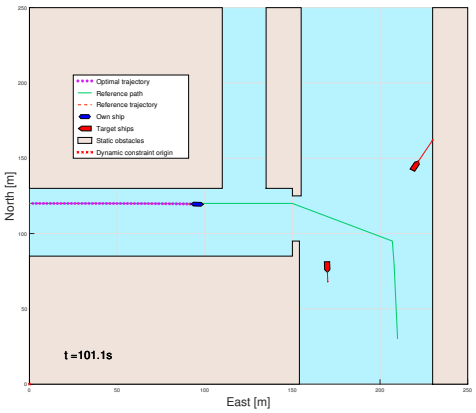
(i) caption



(j) mhm

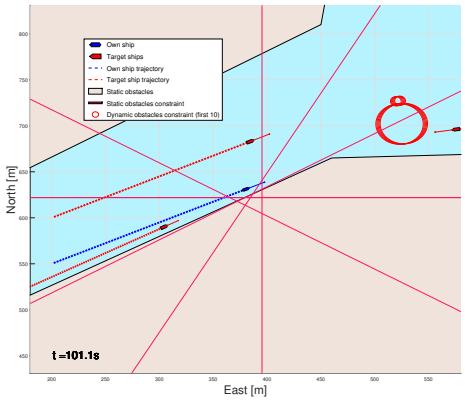


(k) caption

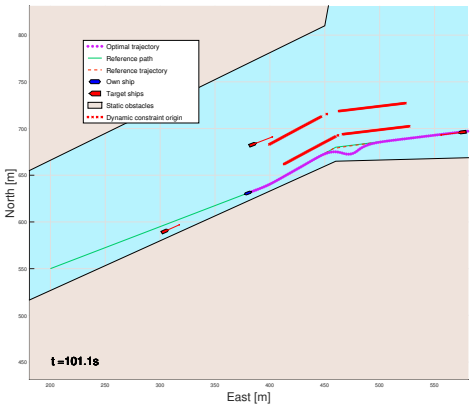


(l) mhm

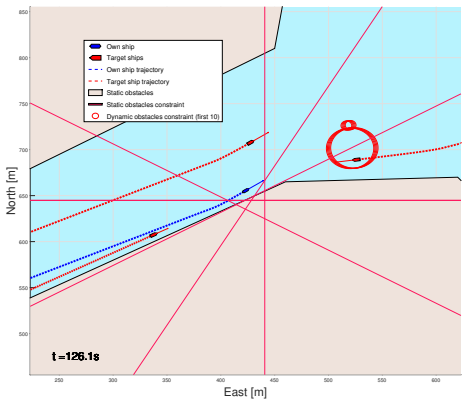
Figure 21: Canals situation. Here shown with simple prediction.



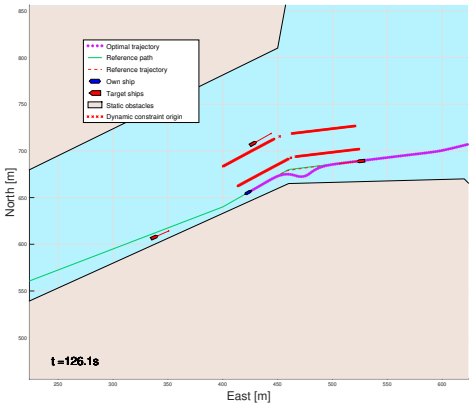
(a) caption



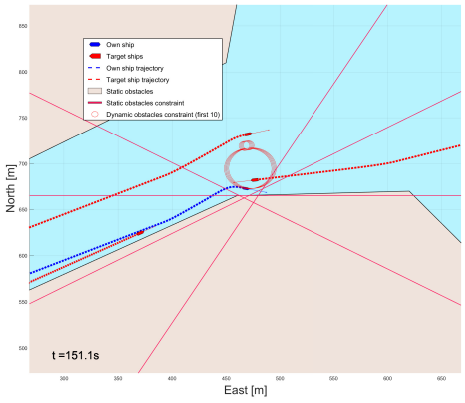
(b) mhm



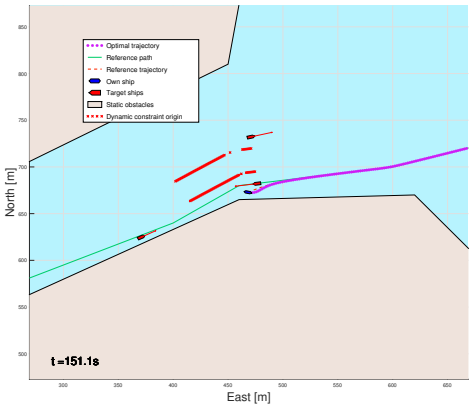
(c) caption



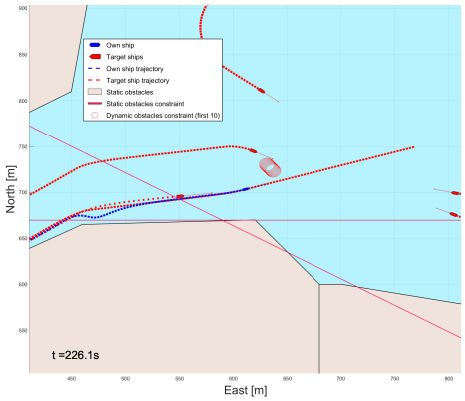
(d) mhm



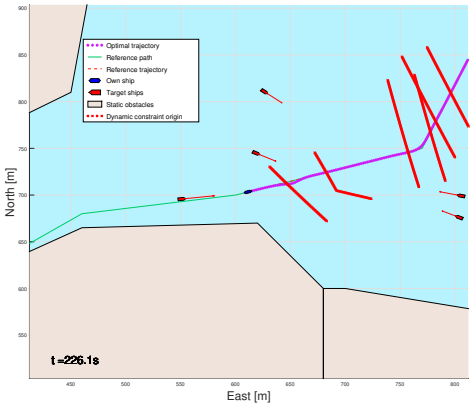
(e) caption



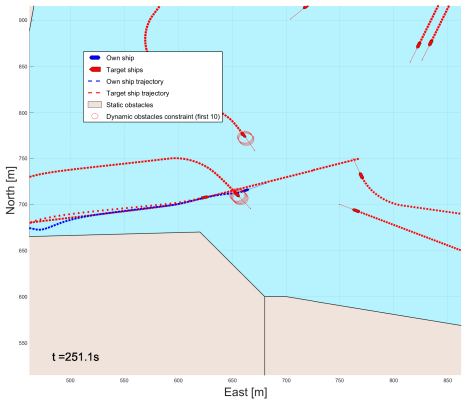
(f) mhm



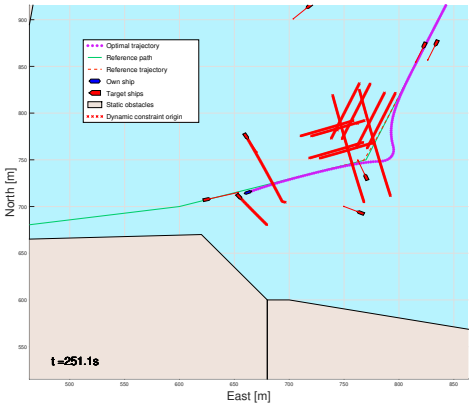
(g) caption



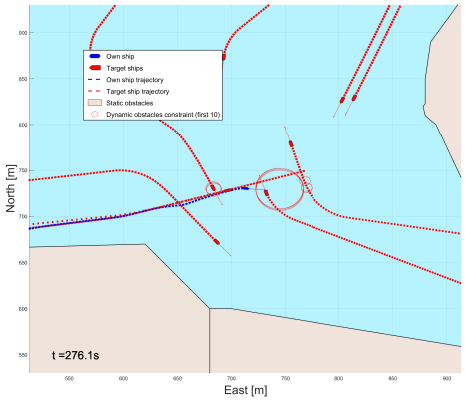
(h) mhm



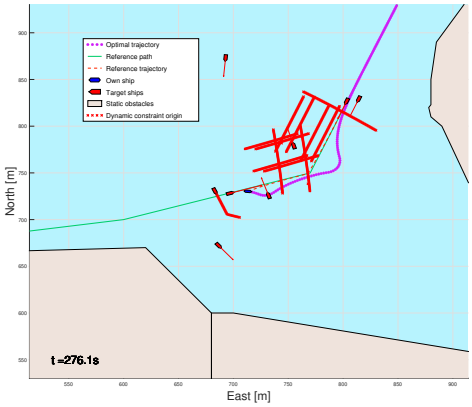
(i) caption



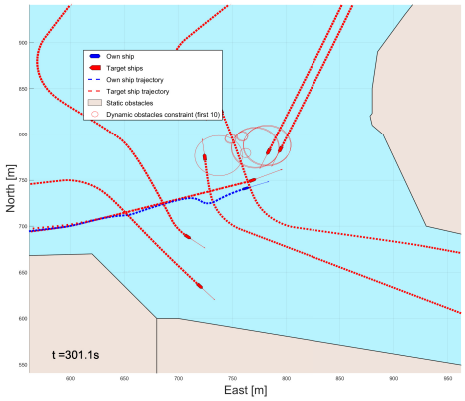
(j) mhm



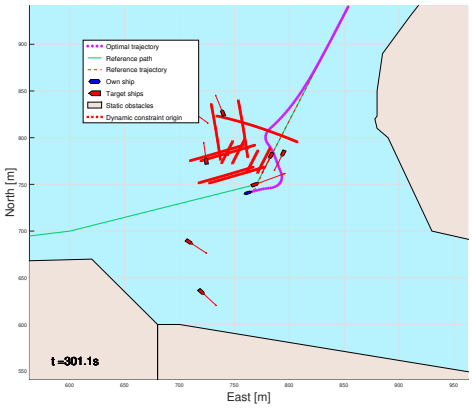
(k) caption



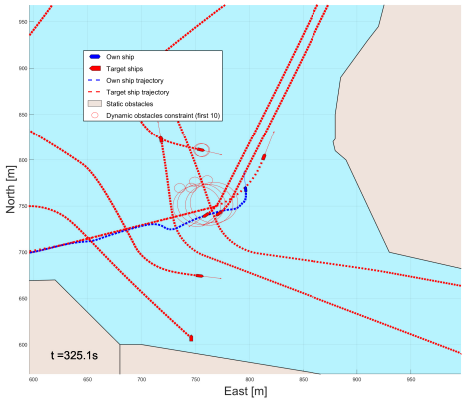
(l) mhm



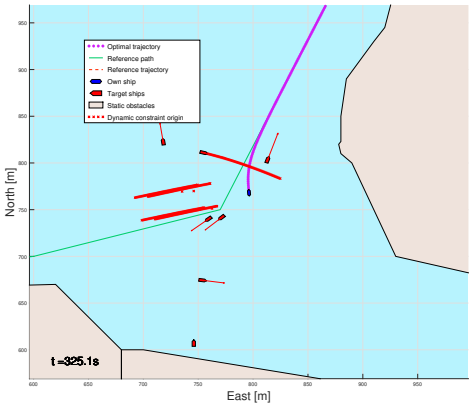
(m) t = 301s.



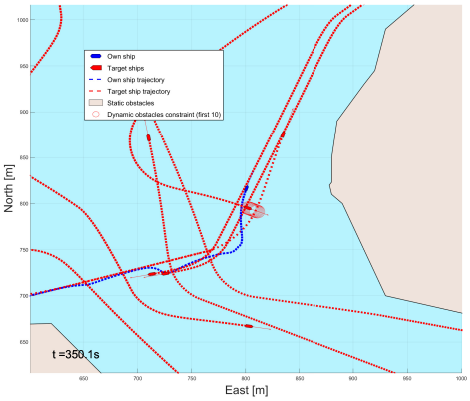
(n) t = 301s.



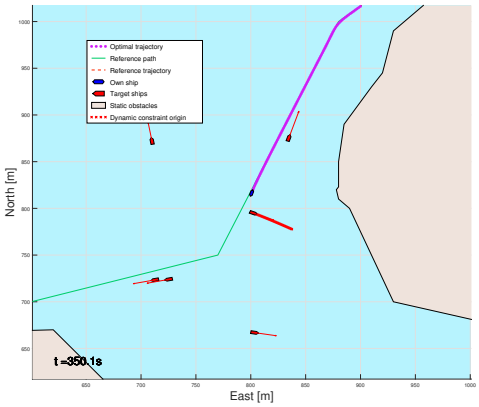
(o) t = 325s.



(p) t = 325s.

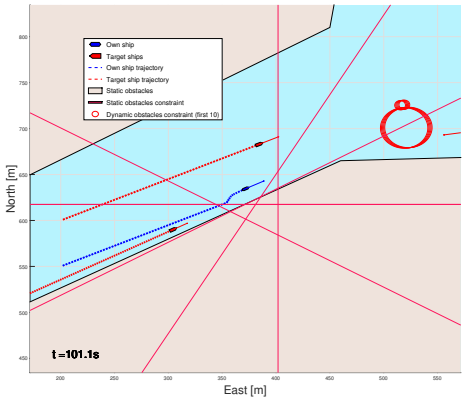


(q) t = 350s.

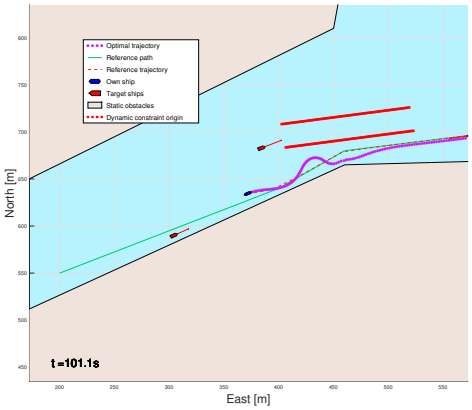


(r) t = 350s.

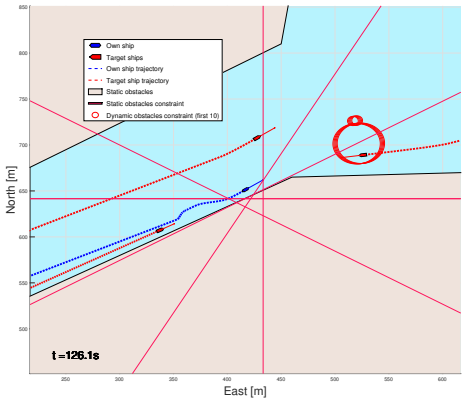
Figure 22: Fjord situation. Here shown with full prediction. Observe the OS handles the stress test pretty well.



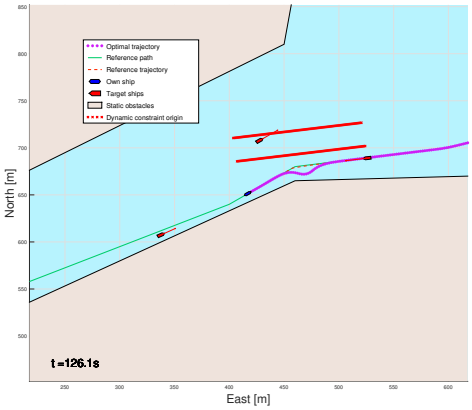
(a) caption



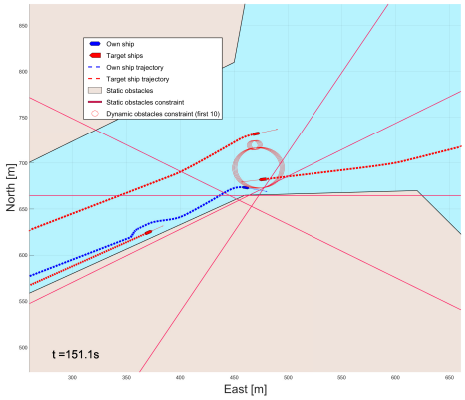
(b) mhm



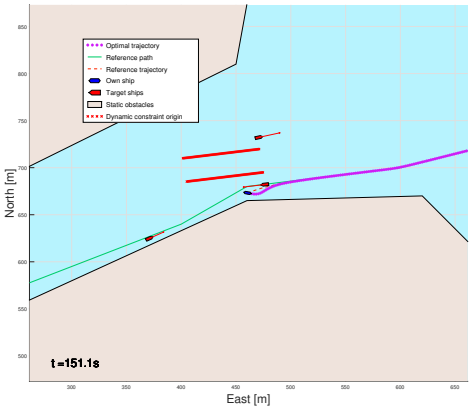
(c) caption



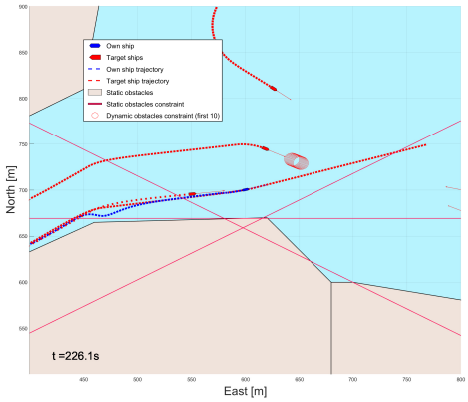
(d) mhm



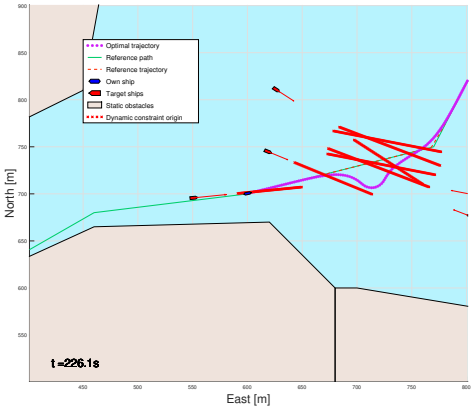
(e) caption



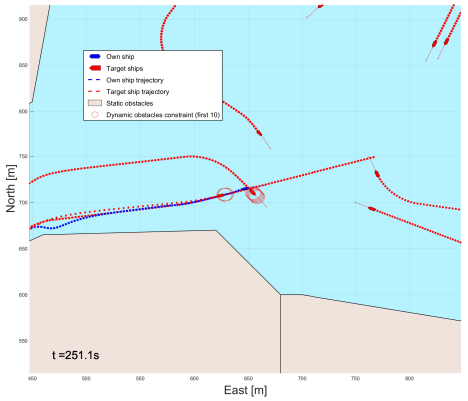
(f) mhm



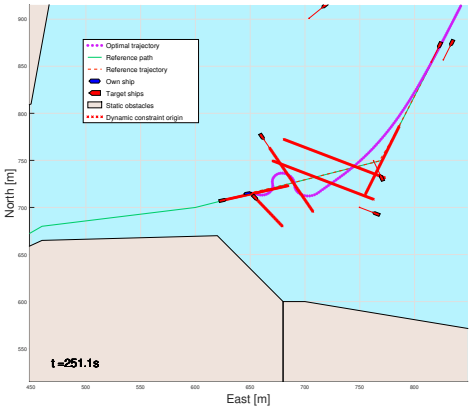
(g) caption



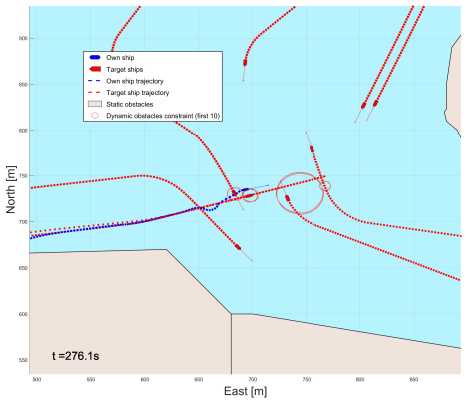
(h) mhm



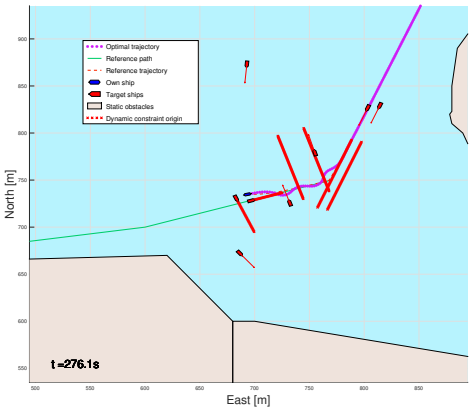
(i) caption



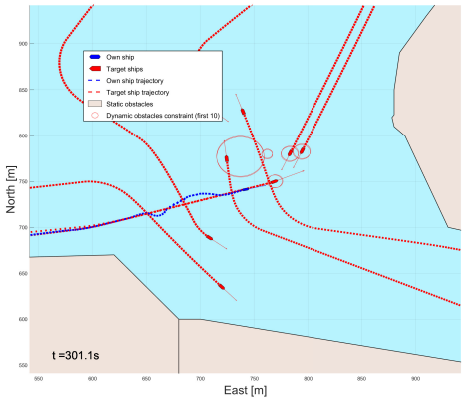
(j) mhm



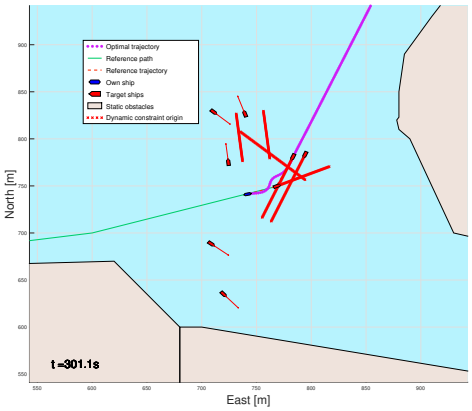
(k) caption



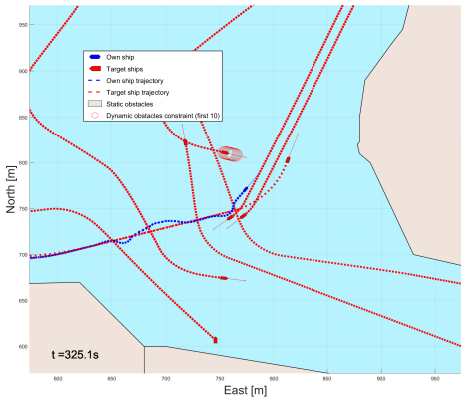
(l) mhm



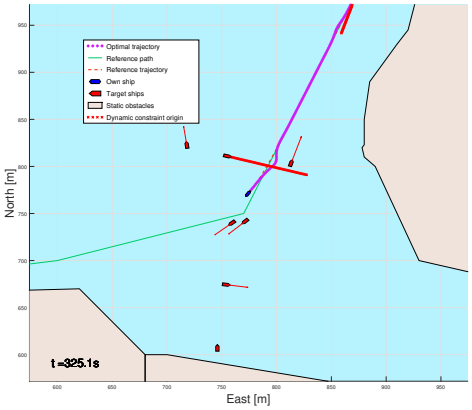
(m) caption



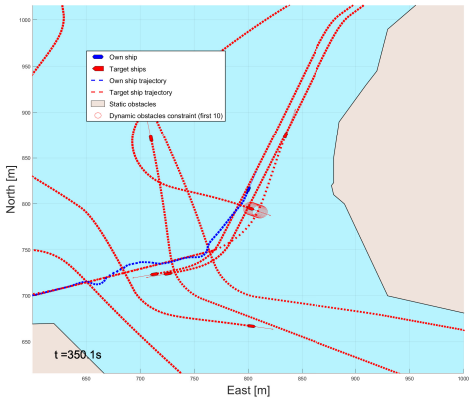
(n) mhm



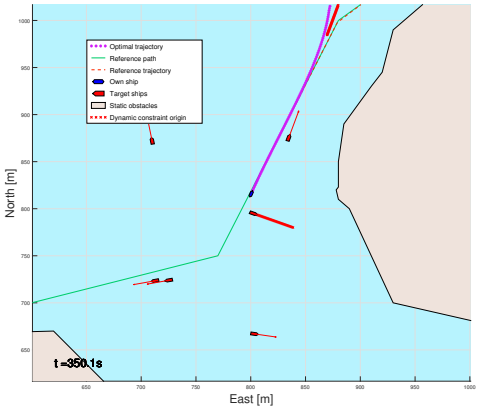
(o) caption



(p) mhm

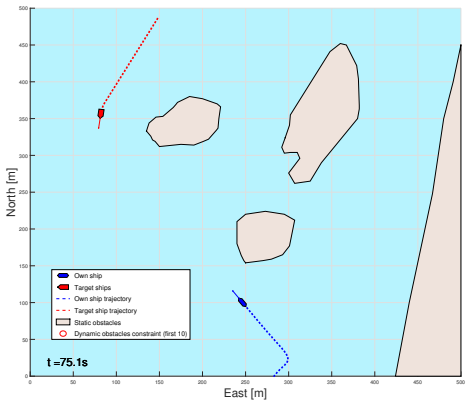


(q) caption

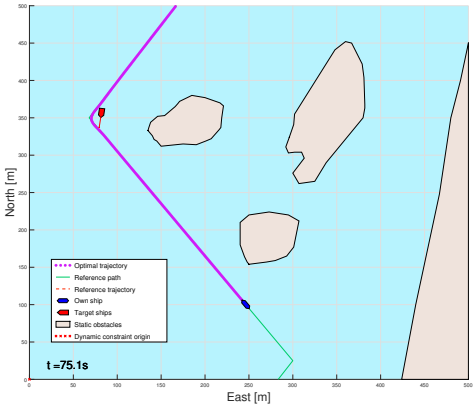


(r) mhm

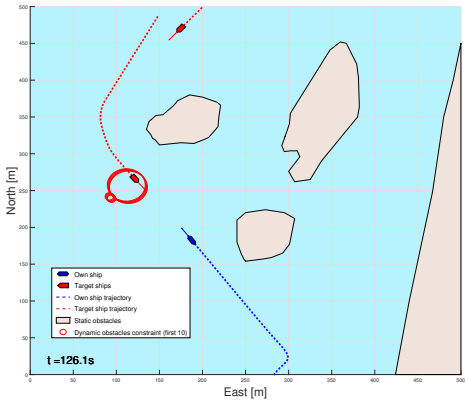
Figure 23: Fjord situation. Here shown with simple prediction. Observe the OS behaves much more erratically compared to Figure 22.



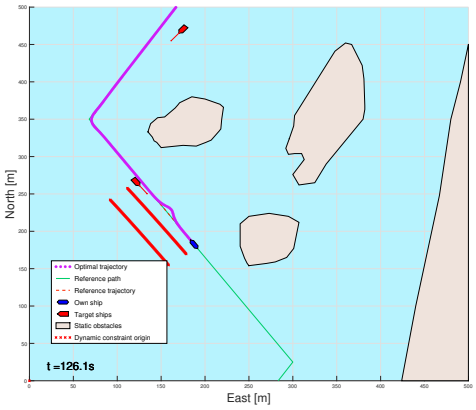
(a) caption



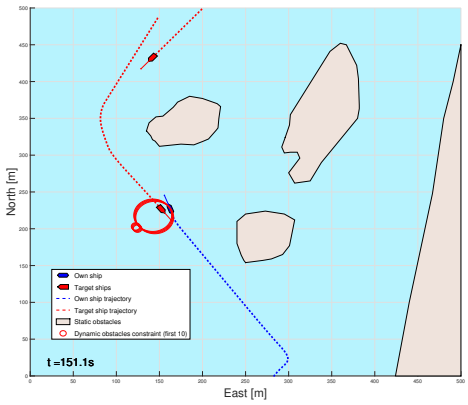
(b) mhm



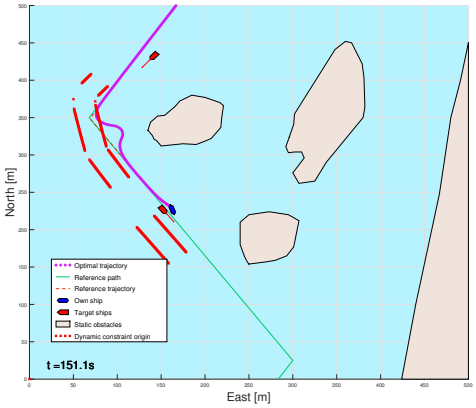
(c) caption



(d) mhm

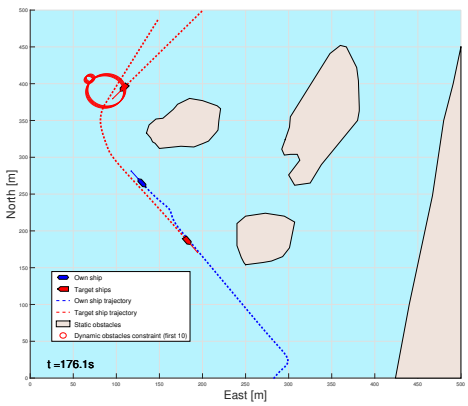


(e) caption

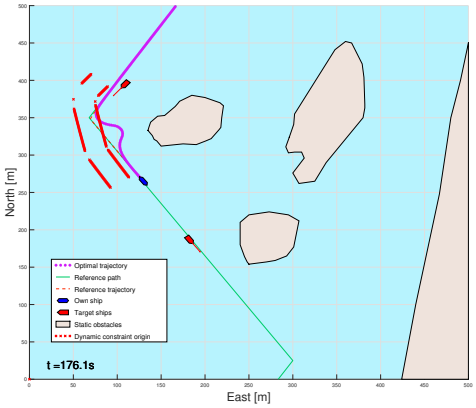


(f) mhm

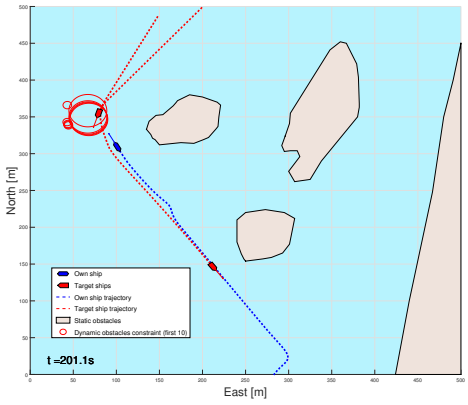
4.2.10 Helløya Reversed



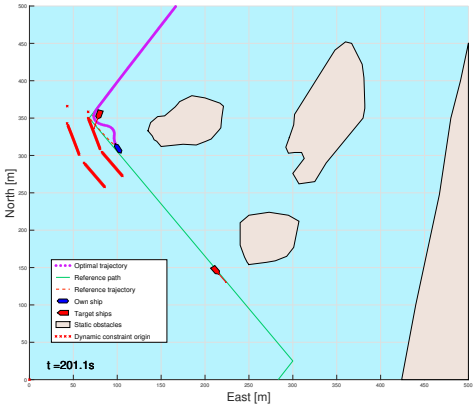
(g) caption



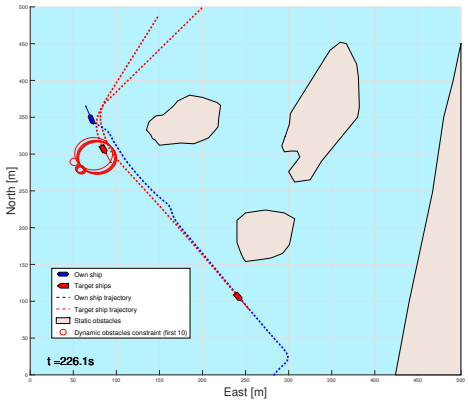
(h) mhm



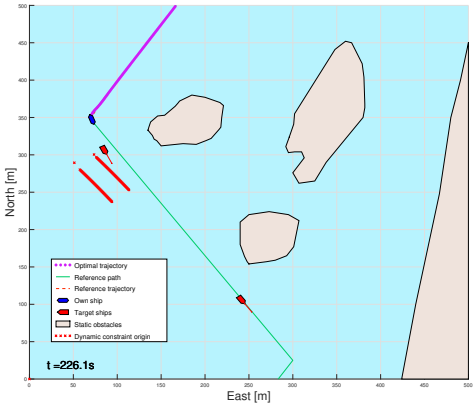
(i) caption



(j) mhm

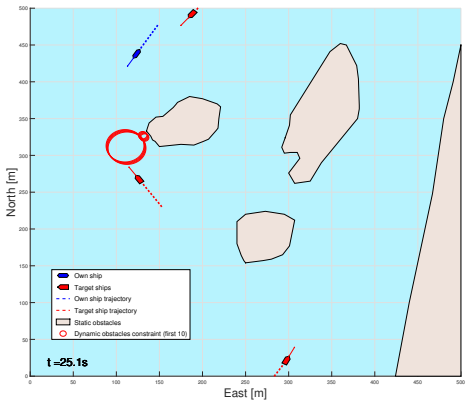


(k) caption

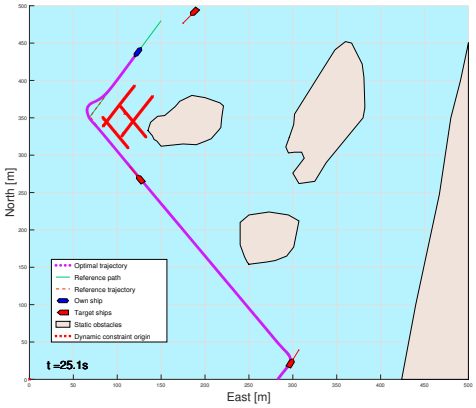


(l) mhm

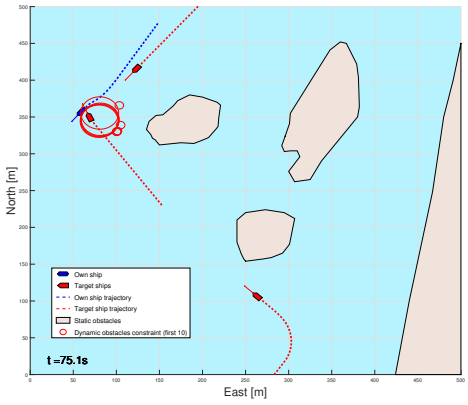
Figure 24: Helloya Situation. Here, OS behaves to expectations independently of prediction level.



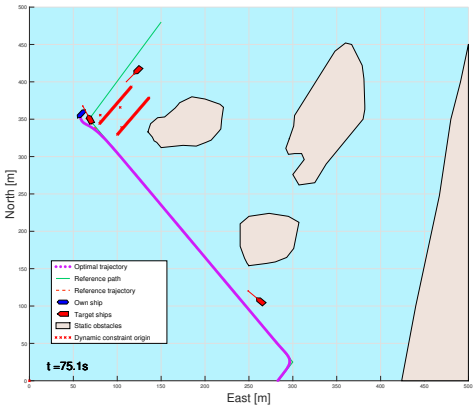
(a) caption



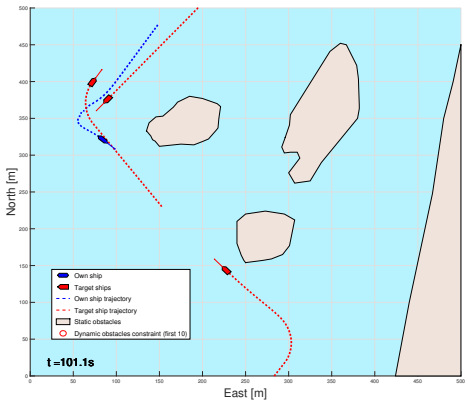
(b) mhm



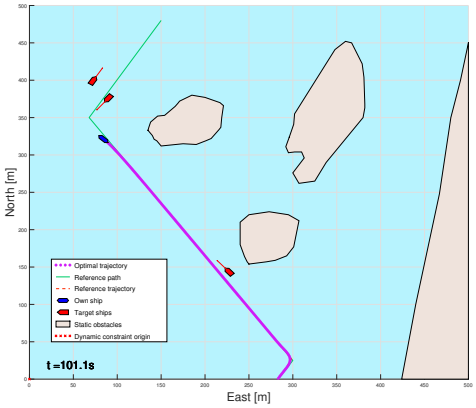
(c) caption



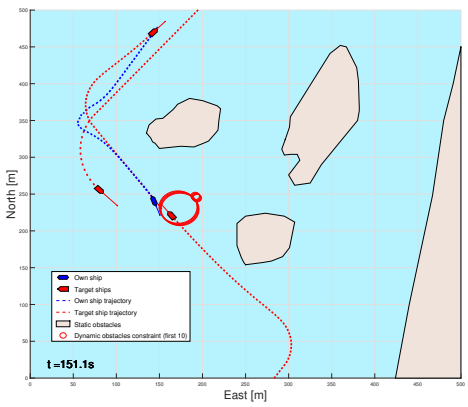
(d) mhm



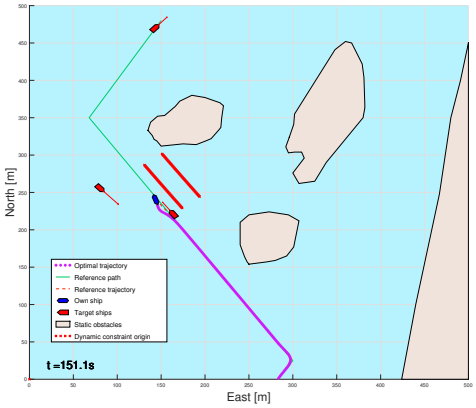
(e) caption



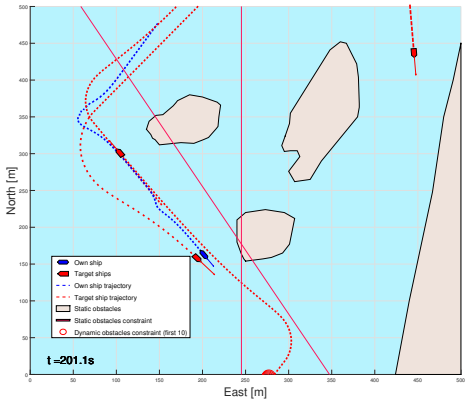
(f) mhm



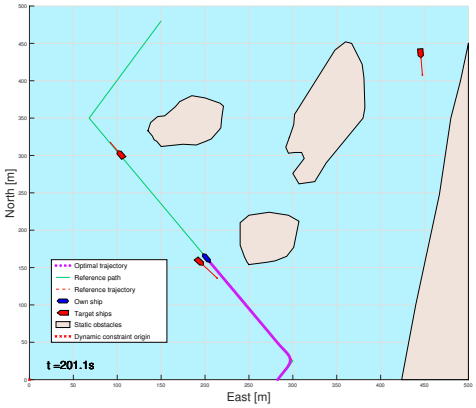
(g) caption



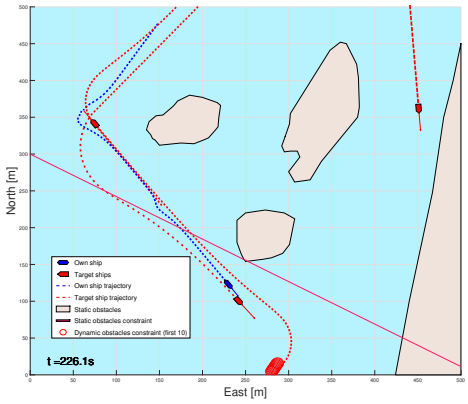
(h) mhm



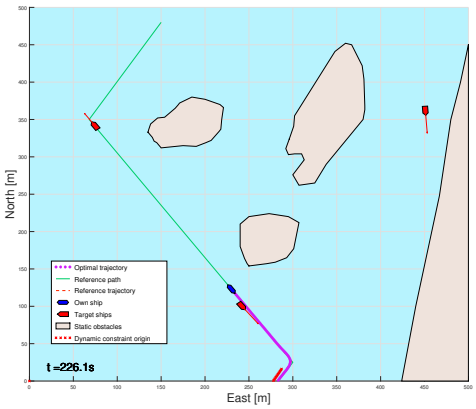
(i) caption



(j) mhm

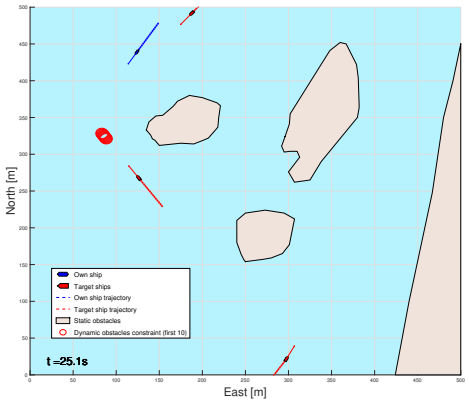


(k) caption

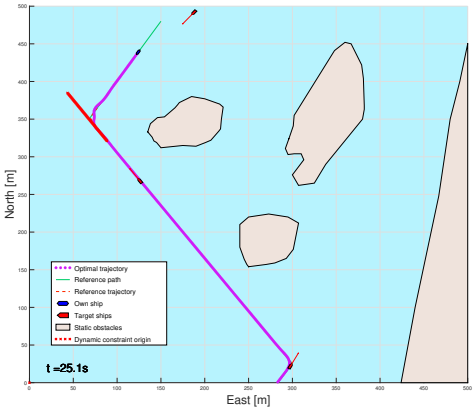


(l) mhm

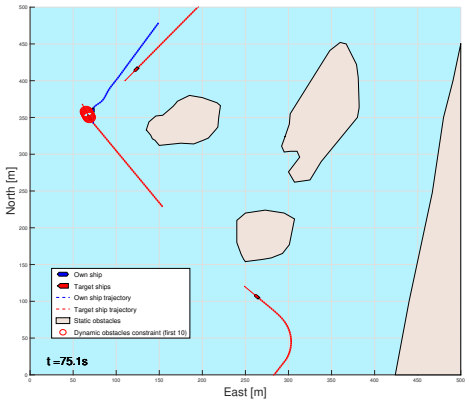
Figure 25: Helloya situation in reverse. Here with full prediction, OS behaves to expectations.



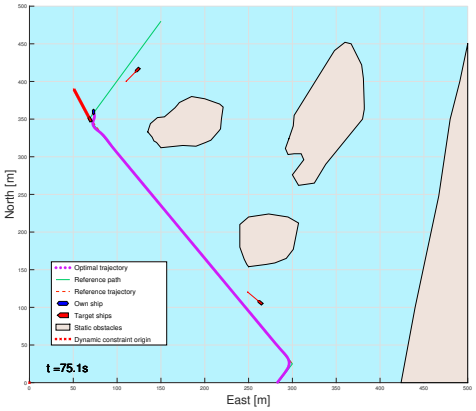
(a) caption



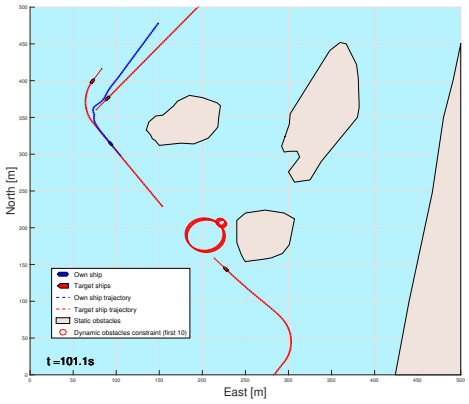
(b) mhm



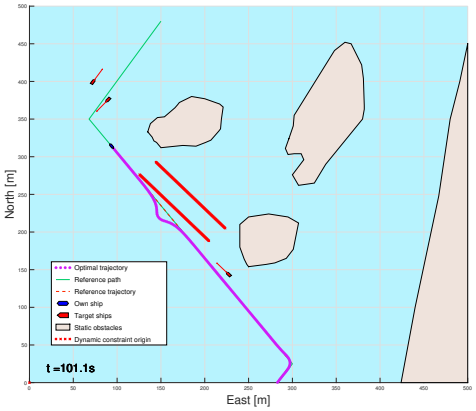
(c) caption



(d) mhm



(e) caption

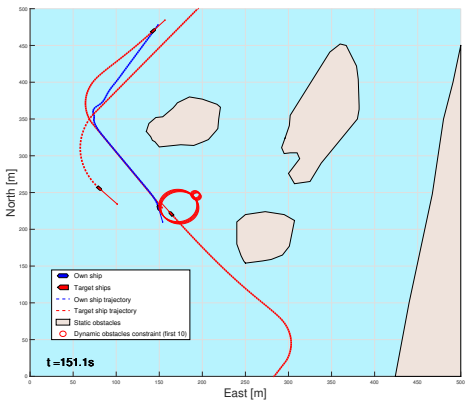


(f) mhm

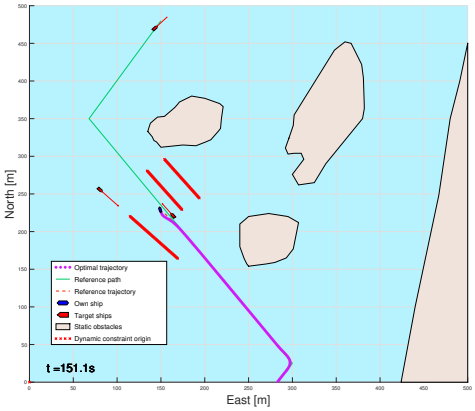
4.2.11 Skjærgård with Traffic

- Both prediction levels successfully navigate this scenario
- with simple prediction the algorithm has an absolute nightmare trying to get past the third TS, this sim took hours to run.

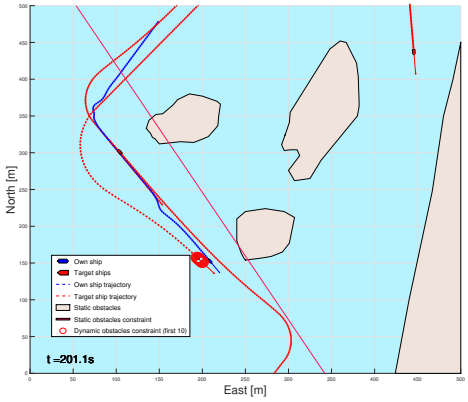
- an interesting problem with 'islands' is that the optimal trajectory could get stuck inside one.
- Tried experimenting with placing bigger and bigger islands on top of the reference path, did not go too well.



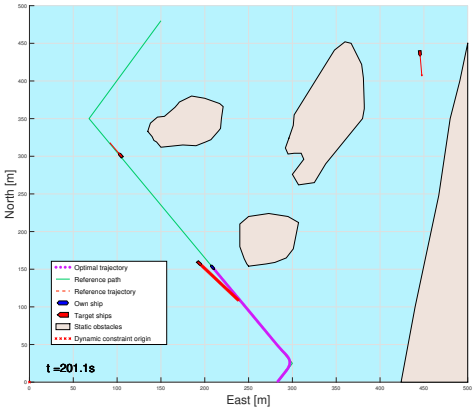
(g) caption



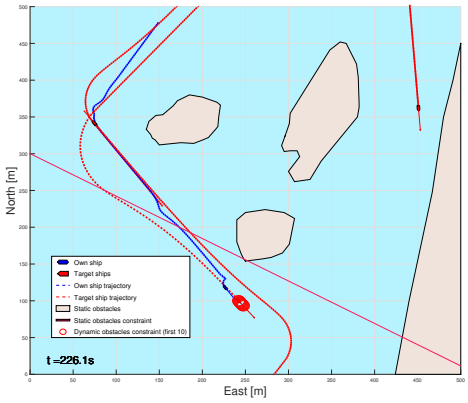
(h) mhm



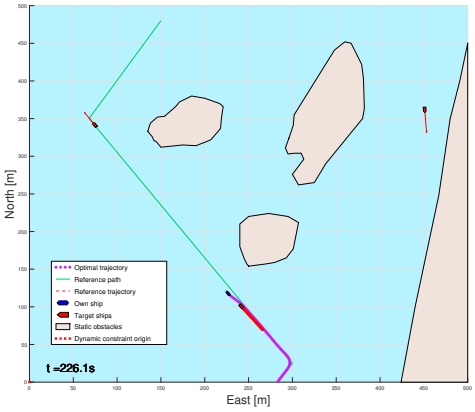
(i) caption



(j) mhm

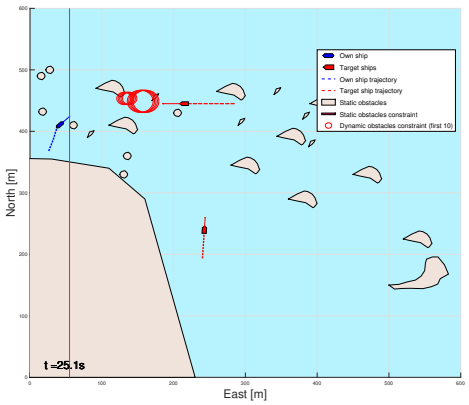


(k) caption

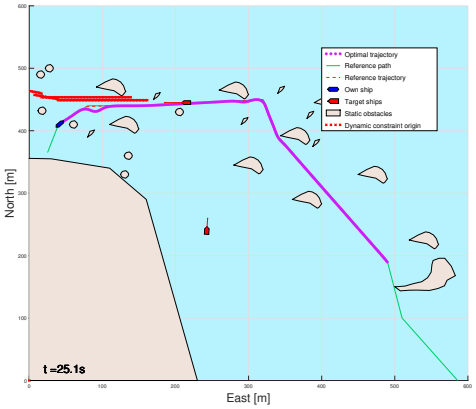


(l) mhm

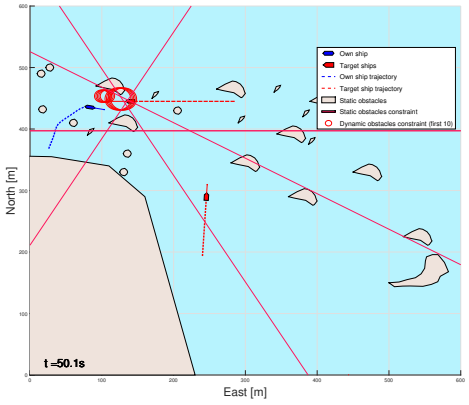
Figure 26: Helloya situation in reverse. Here with simple prediction, OS behaves slightly erratically



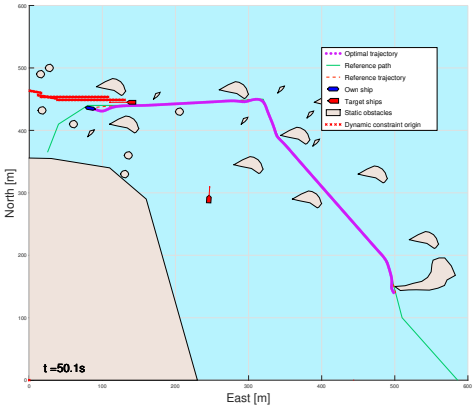
(a) caption



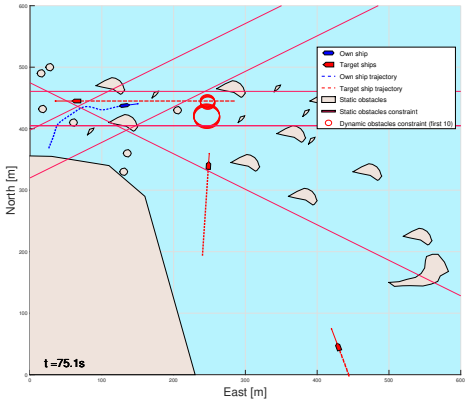
(b) mhm



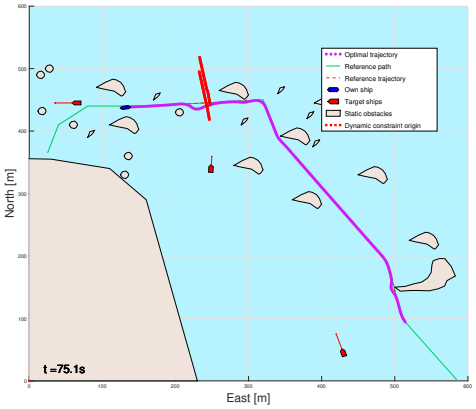
(c) caption



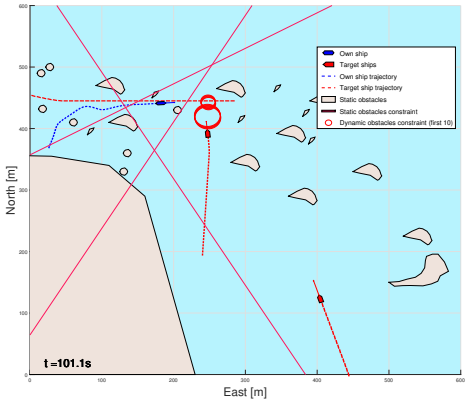
(d) mhm



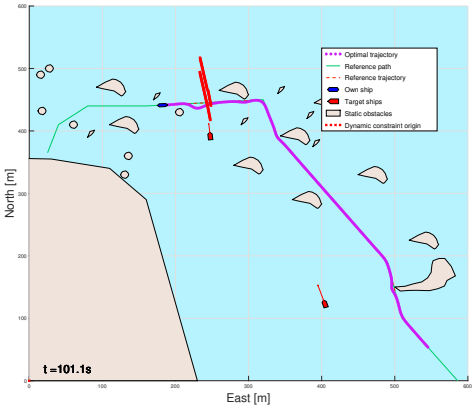
(e) caption



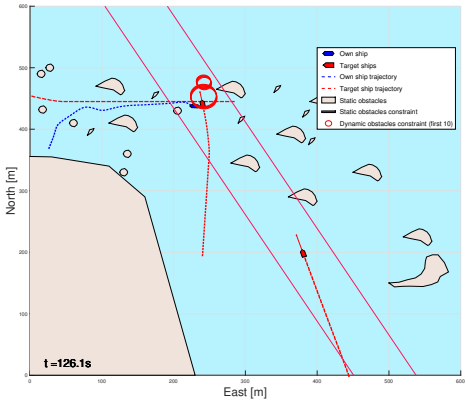
(f) mhm



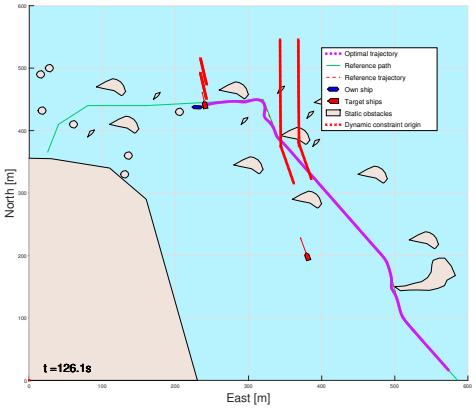
(g) caption



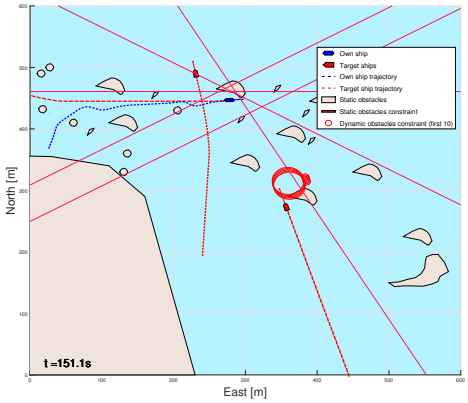
(h) mhm



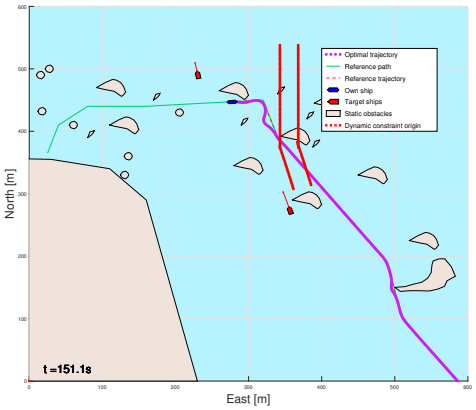
(i) caption



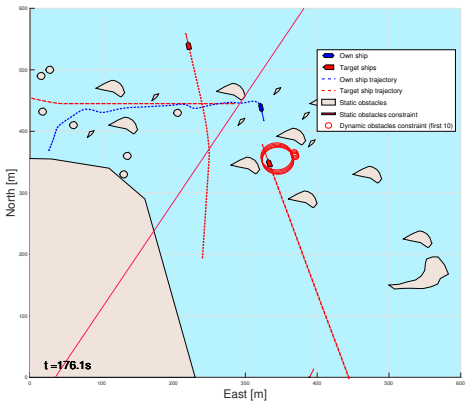
(j) mhm



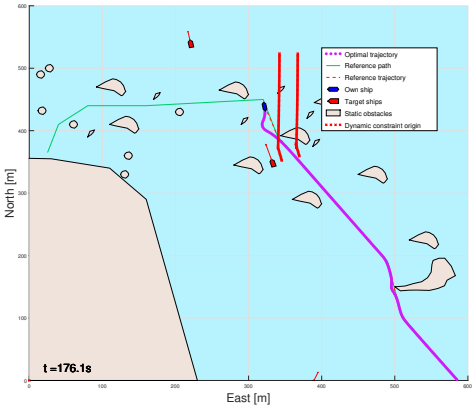
(k) caption



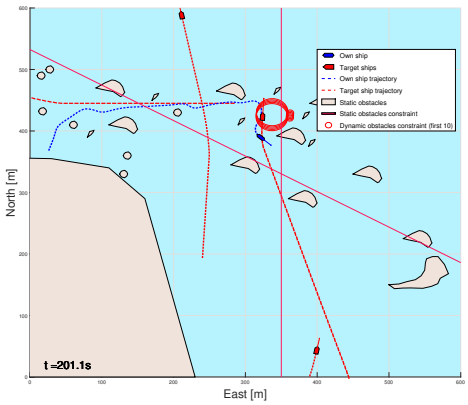
(l) mhm



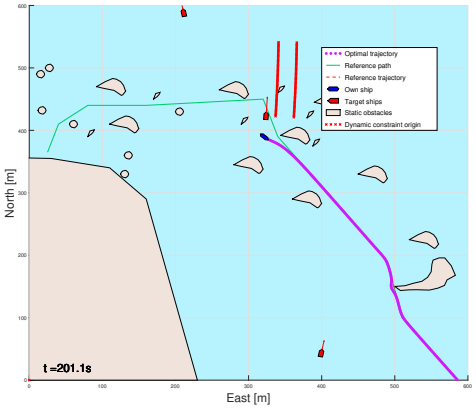
(m) caption



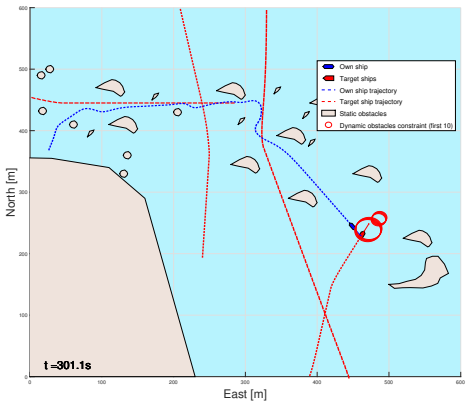
(n) mhm



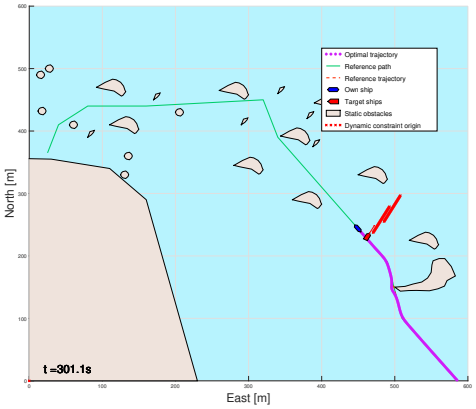
(o) caption



(p) mhm

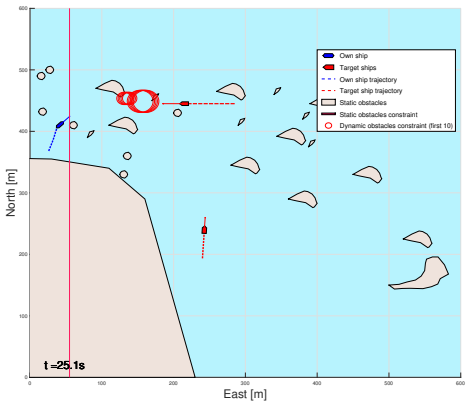


(q) caption

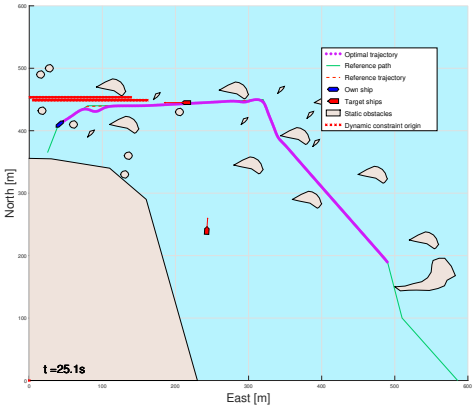


(r) mhm

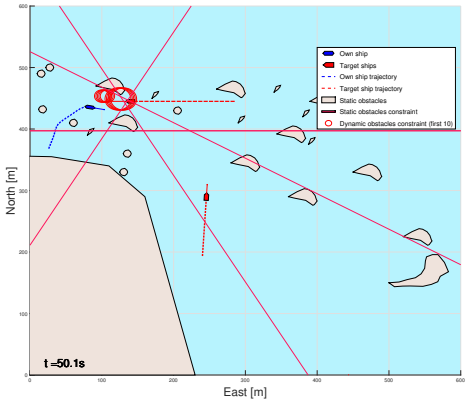
Figure 27: Skjærgård with traffic situation. Here with full prediction.



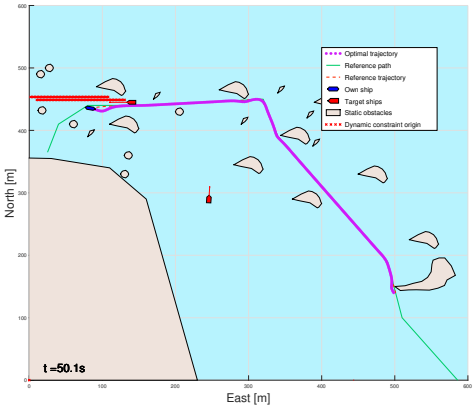
(a) caption



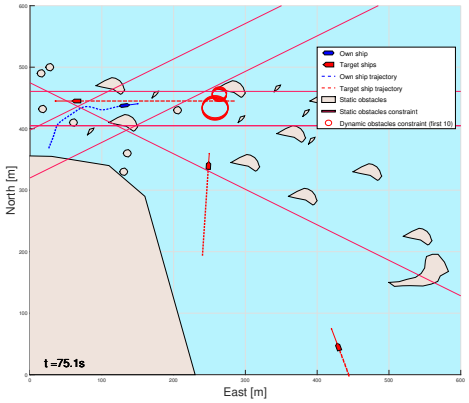
(b) mhm



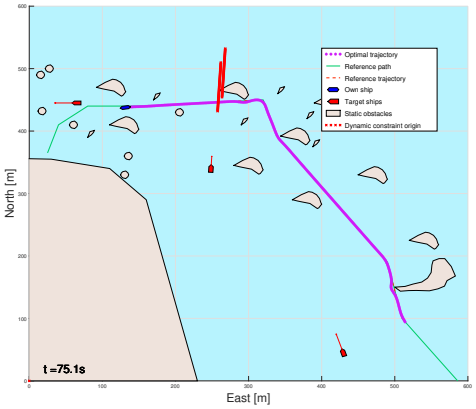
(c) caption



(d) mhm



(e) caption

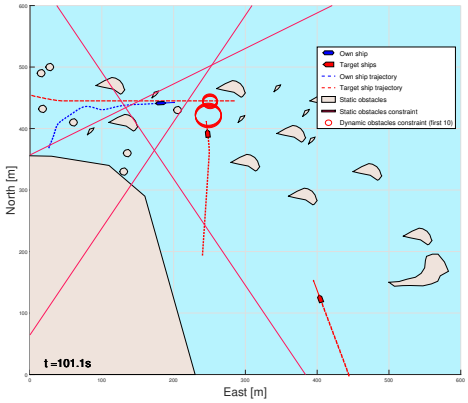


(f) mhm

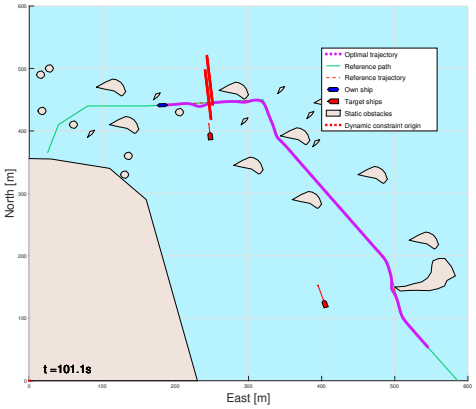
4.2.12 skjærgård without Traffic

- Good path tracking.
- this is where I discovered a heading reference problem.
- able to dodge islands that are in the way.

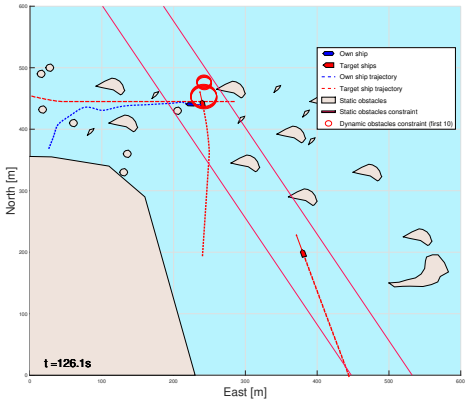
- computationally not too difficult either, algorithm exhibit reasonable execution times.



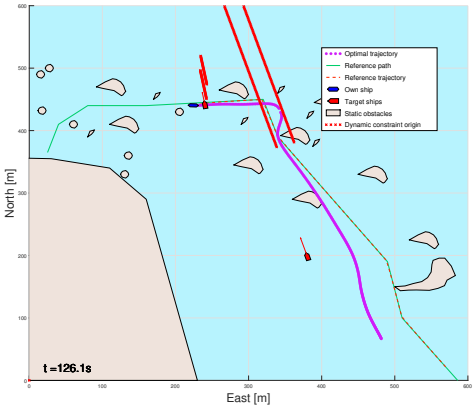
(g) caption



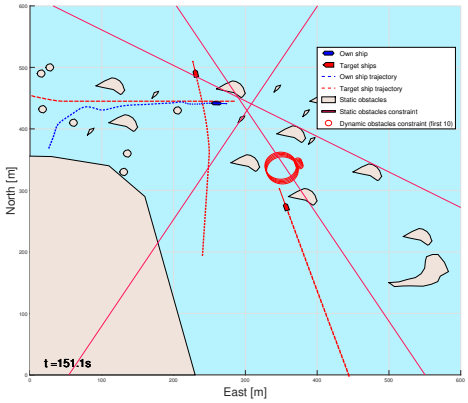
(h) mhm



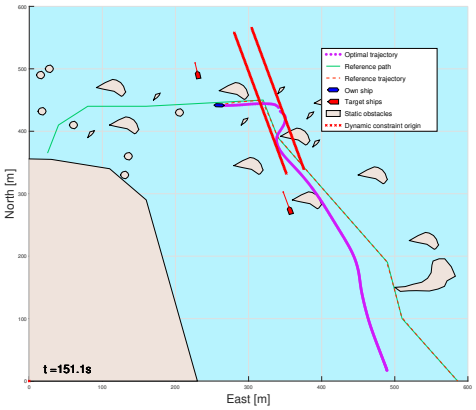
(i) caption



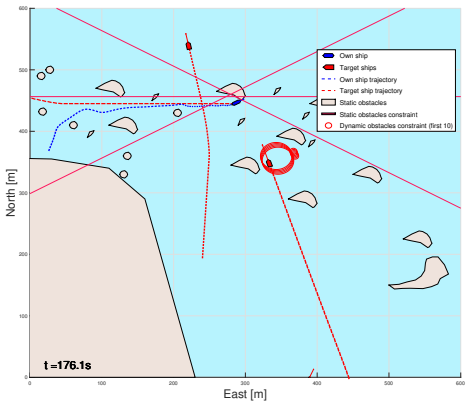
(j) mhm



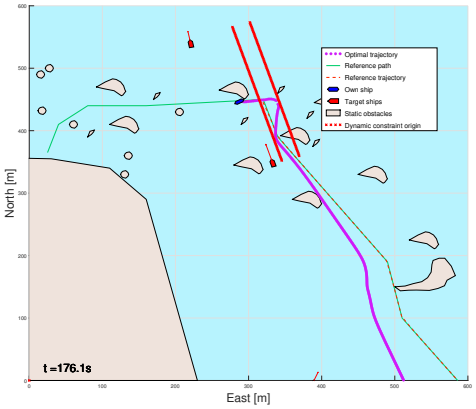
(k) caption



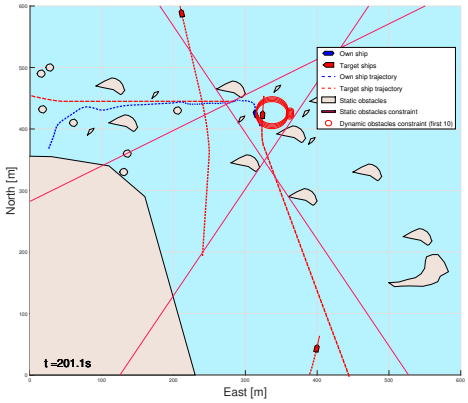
(l) mhm



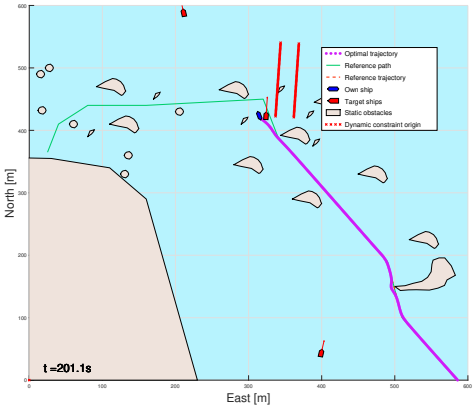
(m) caption



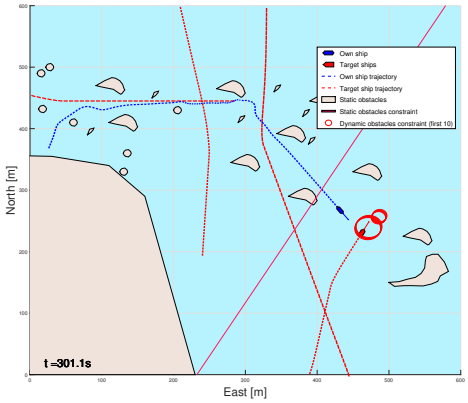
(n) mhm



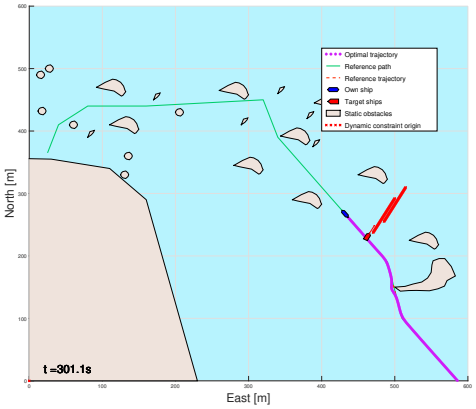
(o) caption



(p) mhm

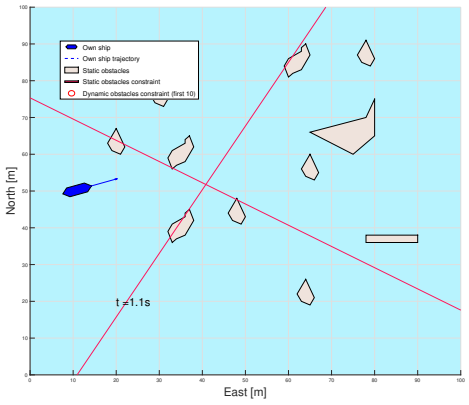


(q) caption

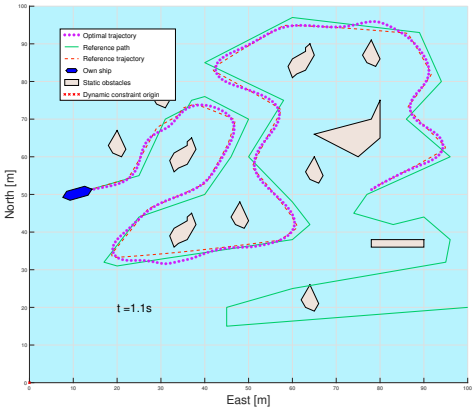


(r) mhm

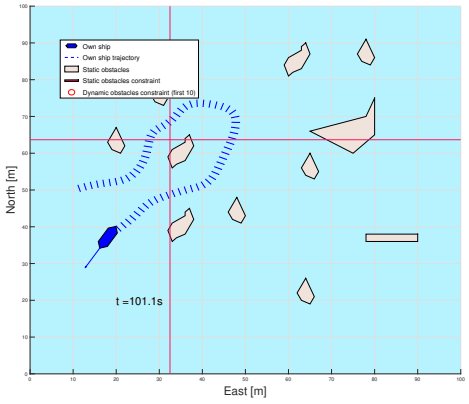
Figure 28: Skjærgård with traffic situation. Here with simple prediction.



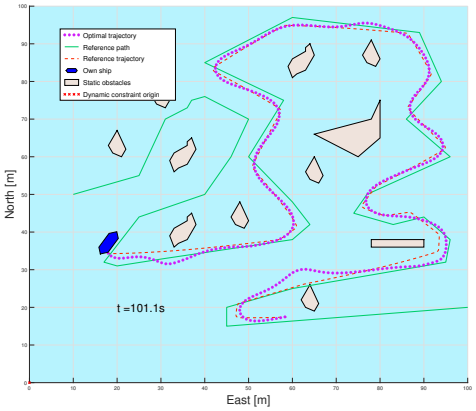
(a) caption



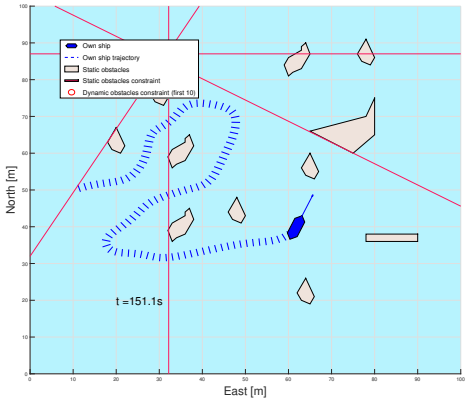
(b) mhm



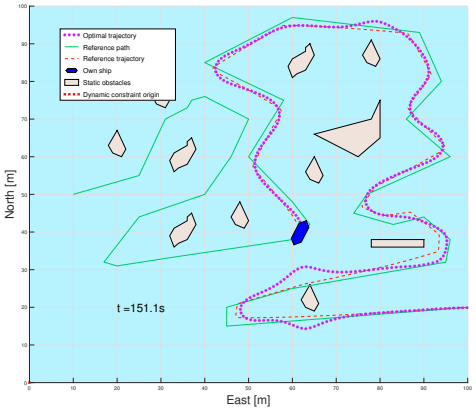
(c) caption



(d) mhm



(e) caption

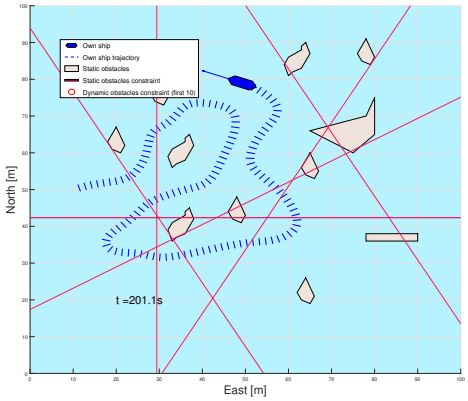


(f) mhm

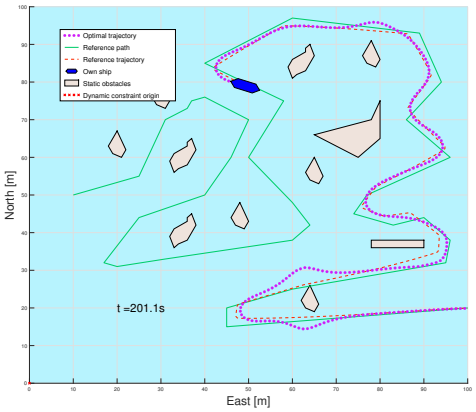
4.2.13 Miscellaneous

- Bad Prediction, what happens when the target ship does not follow the predicted path
- Blocked Path, a closer look at what happens when the path we intend to take is fully blocked.

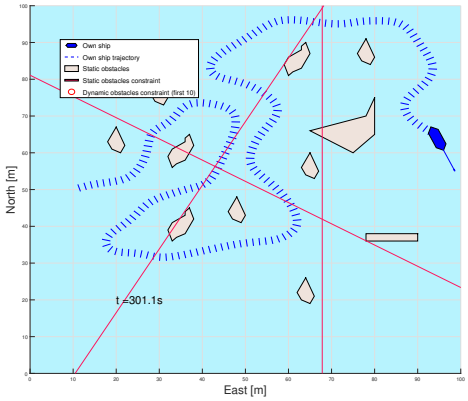
- Wrong turn, observe that the optimal trajectory is often to turn the wrong direction slightly when changing course.
- WrapTo2Pi problems, how to explain to an algorithm how course works.
- 'Dragged' along by Target Ships. When does it happen.
- When overtaking or being overtaken the constraints can really mess with the IPOPT solver.
- the optimal solution could get trapped inside static obstacles with the way the constraints are active 'both ways'. please provide picture.



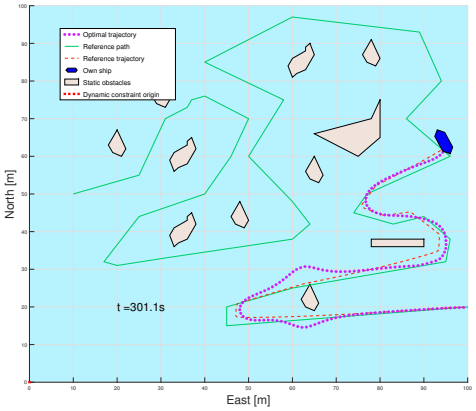
(g) caption



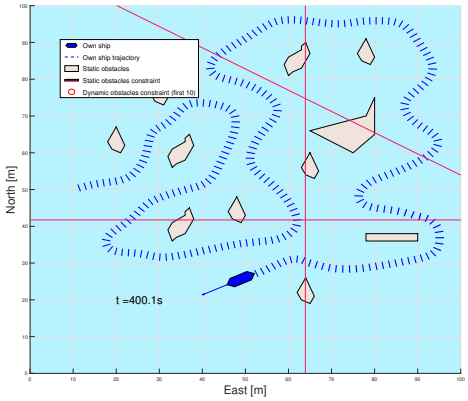
(h) mhm



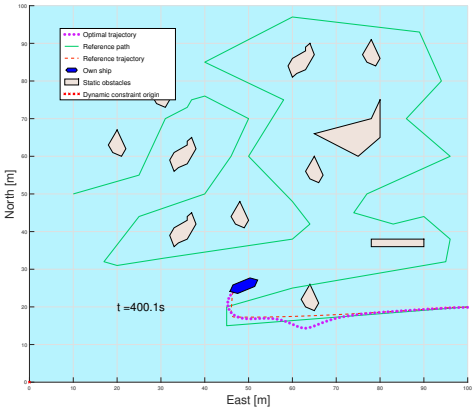
(i) caption



(j) mhm

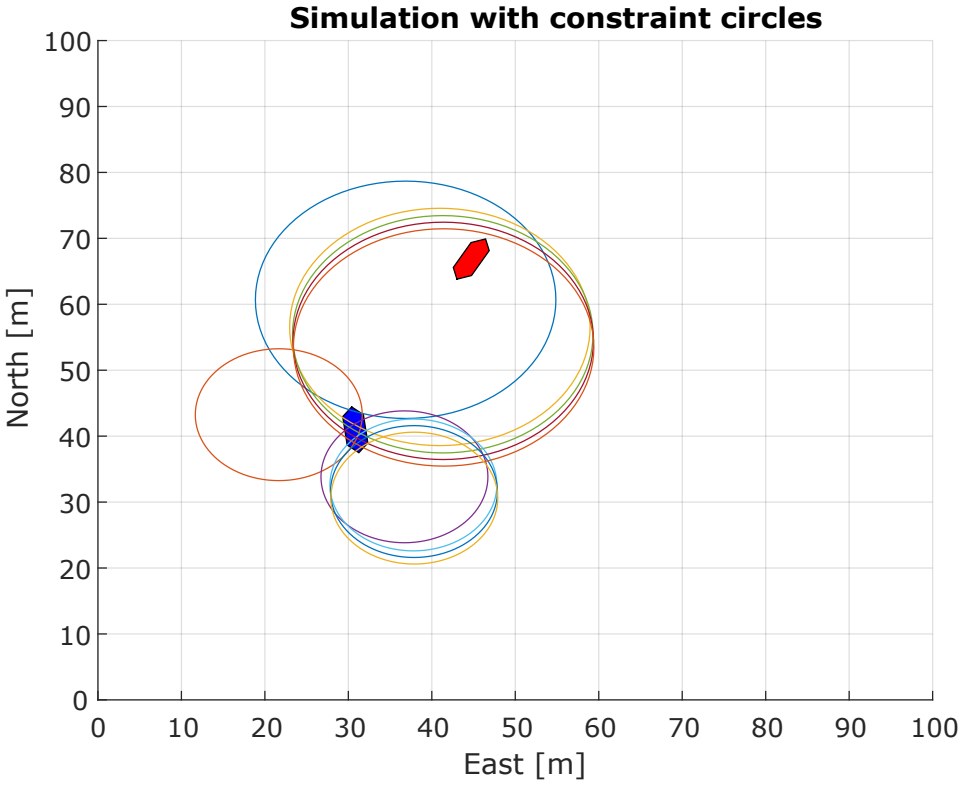


(k) caption

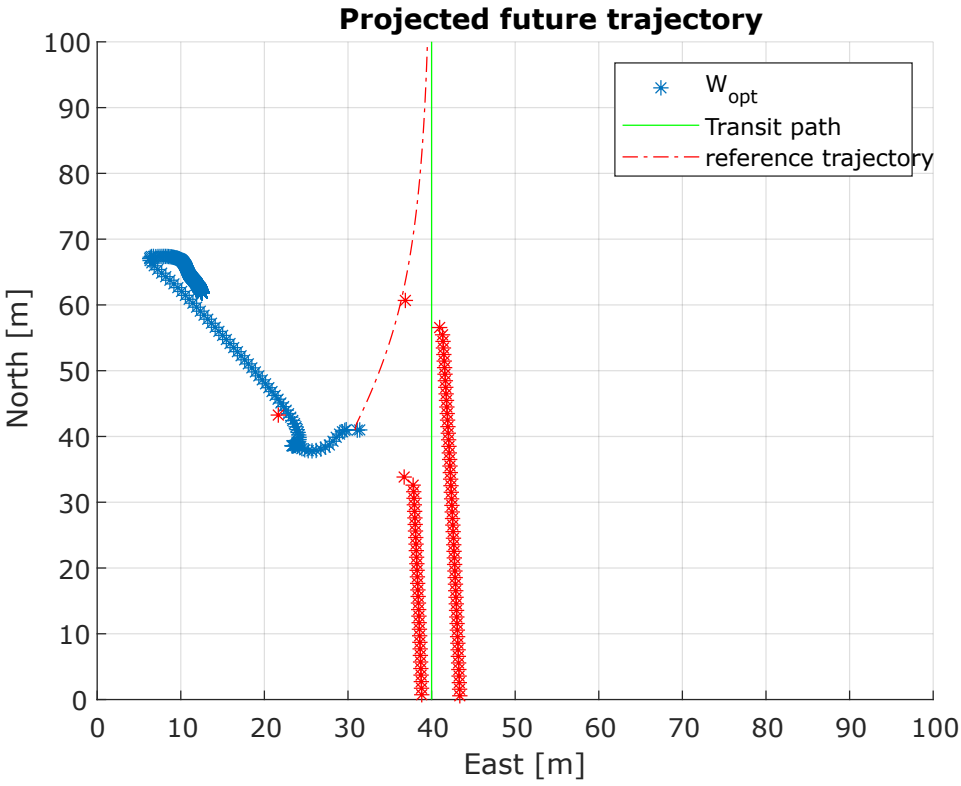


(l) mhm

Figure 29: Skjærgård without traffic simulation. Result independent of prediction level due to no TSs.

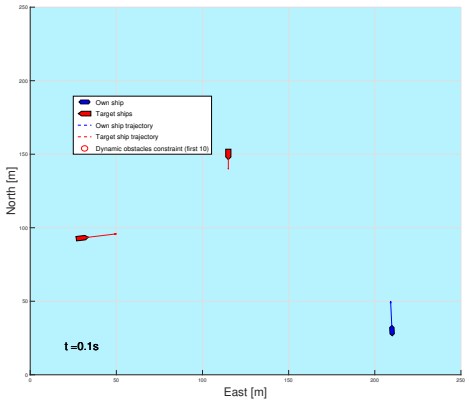


(a) When prediction goes wrong, the OS can get caught by moving constraints. (Old style figure).

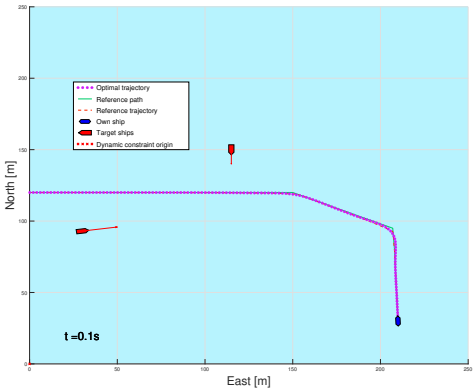


(b) When caught inside an active constraint, the solver is unable to find a feasible solution. (Old style figure).

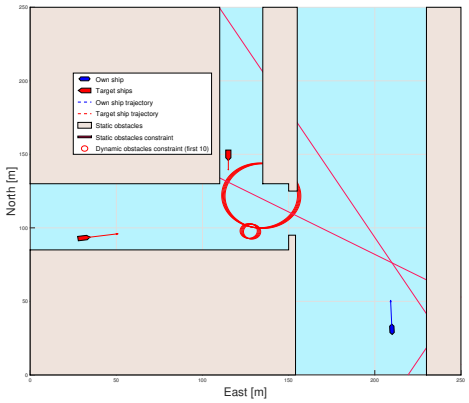
Figure 30: This is what can happen when the prediction does not match the actual trajectory of TSs.



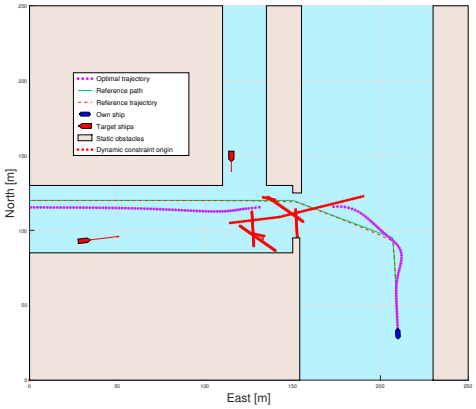
(a) Start of simulation, no active obstacles.



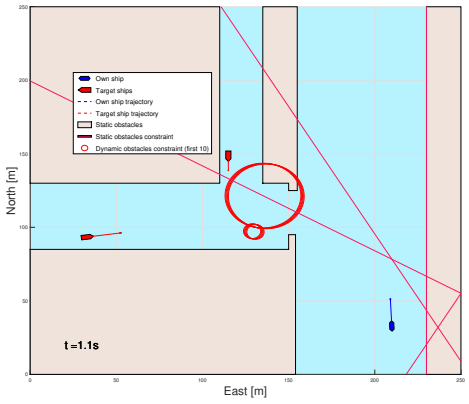
(b) Start of simulation, no active obstacles.



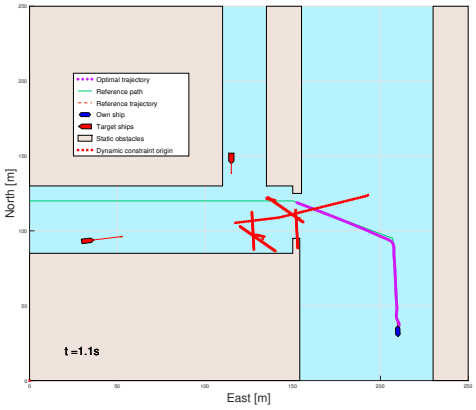
(c) Obstacles activate, breaking the optimal trajectory.



(d) Obstacles activate, breaking the optimal trajectory.

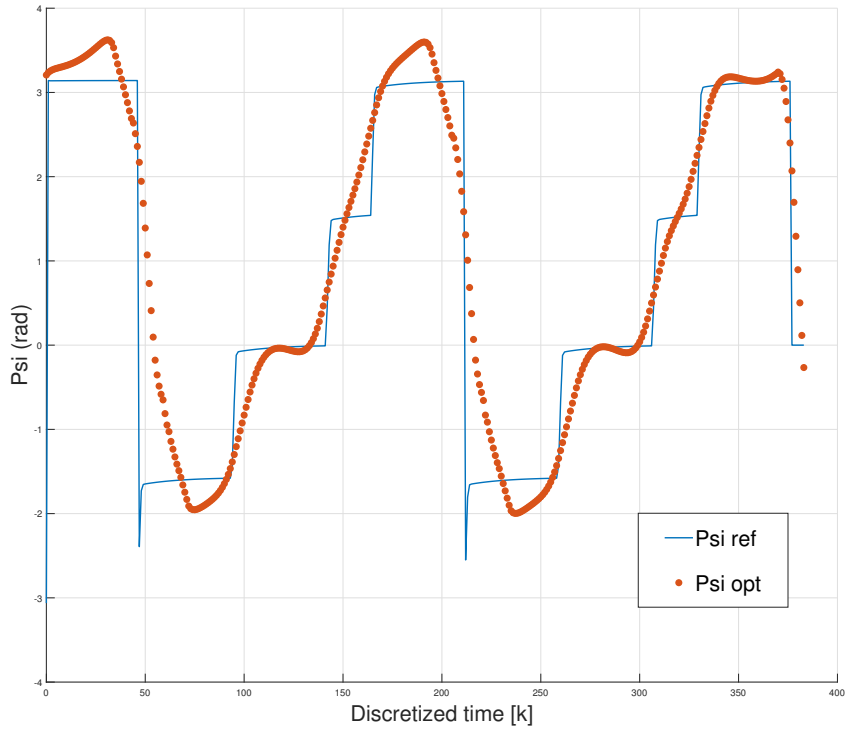


(e) Speed is reduced, resulting in a shorter optimal trajectory.

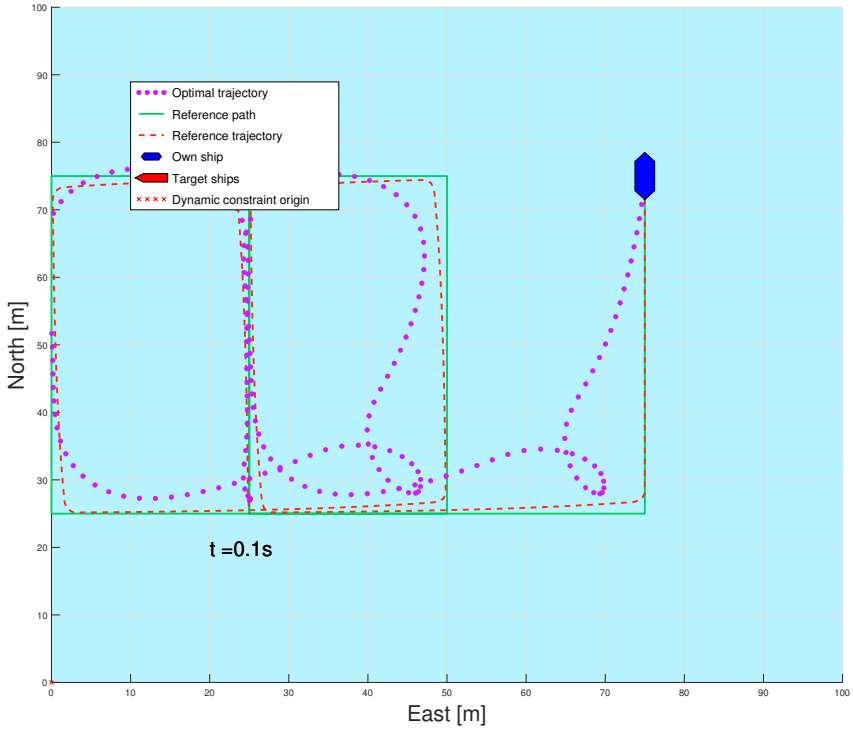


(f) Speed is reduced, resulting in a shorter optimal trajectory.

Figure 31: How optimal path is calculated with lower speed when infeasibility is detected.



(a) ref



(b) wopt

Figure 32: Without proper course reference, this sometimes happens.

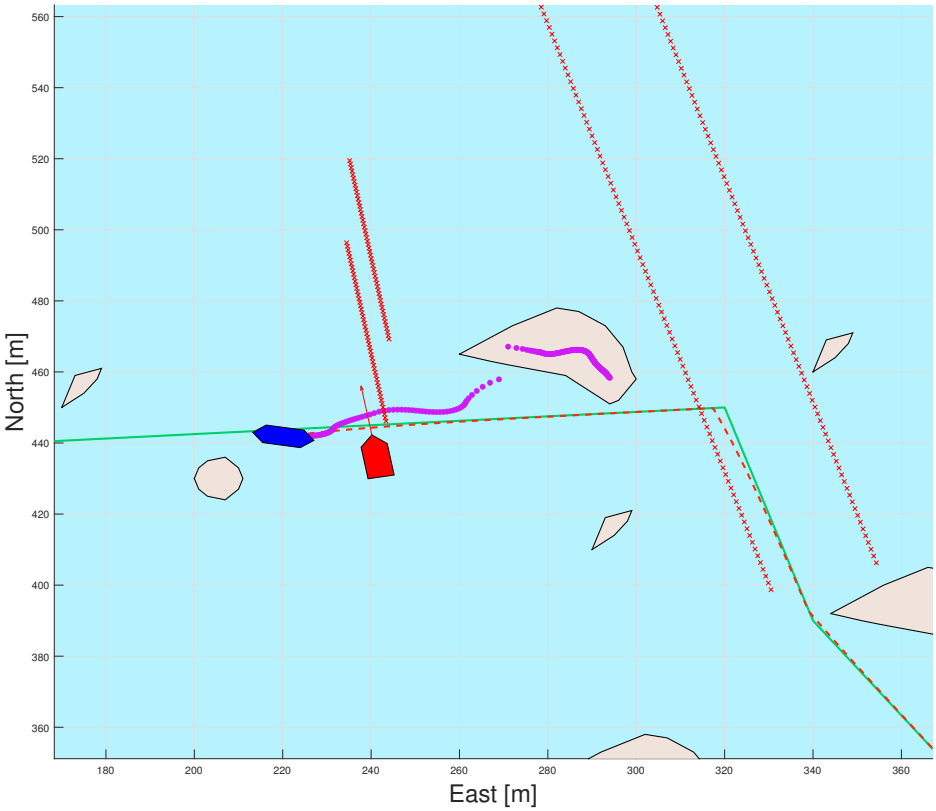
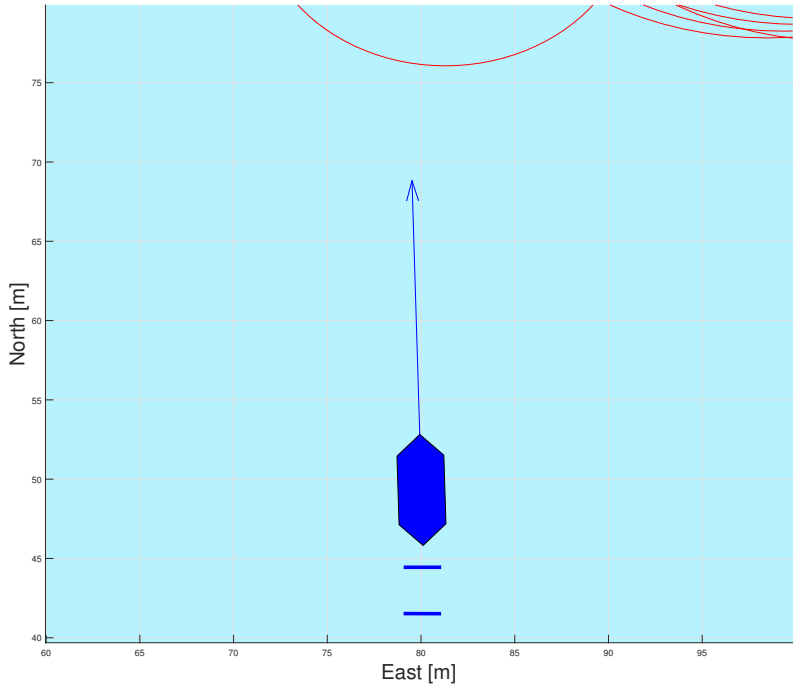
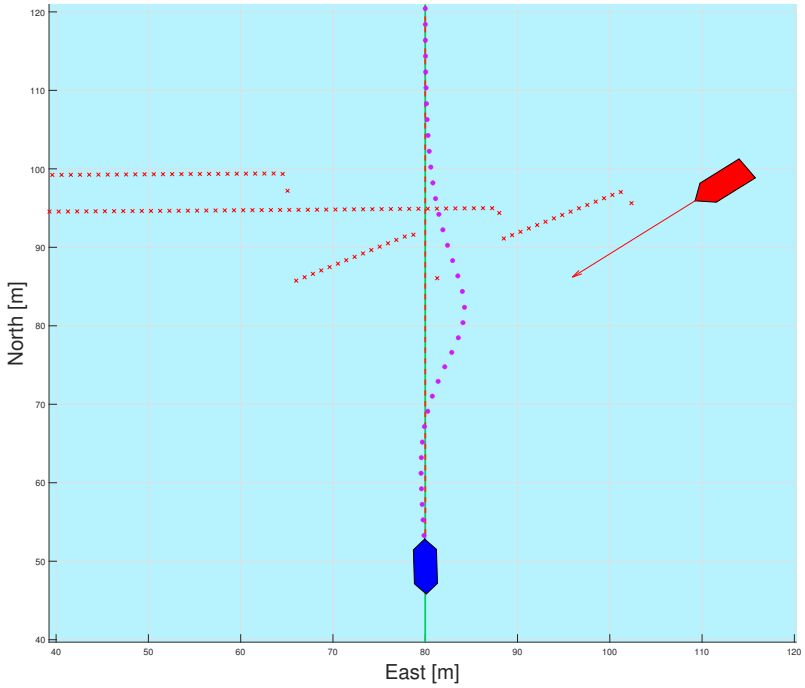


Figure 33: Stuck inside a static obstacle.



(a) By zooming in it is observed that the OS turns slightly to port side.



(b) Meanwhile the optimal trajectory clearly is clearly a turn to starboard.

Figure 34: A quirk of numerical optimization, sometimes turning to the wrong side leads to a 'smoother' curve.

4.3 Discussion

- Hvorfor er viktigere en hva
- ikke overanalyser resultat, ikke dra ville konklusjoner.
- Hvis et resultat er mye værre enn forventet kan det godt være det er bugs.
- i tillegg til det resultatene viser kan jeg også skrive om det jeg kan se med debugging.
- WrapTo2Pi problems (shortest signed angle stuff)
- Turning the wrong way to get a more even turn, Optimization leads to this problem.
- if(ĩsempty(previous_w_opt)) && feasibility ==
  previous_feasibility && feasibility skaper problemer
- We really don't want to put a cost on heading reference more than neccessary, heading will often not be correct due to disturbances. heading is also just plain wrong any time we deviate from the reference trajecotry.
- With 'full' prediction solving the NLP is often computationally more efficient due to a better previous_w_opt.
- WHEN BEING OVERTAKEN: needs a better method for Standing On, putting constraints on overtaking vessel is not sufficient.
  See helloya rev without pred.
- Standing on in general needs a better way of handling constraints.
- Perhaps an alternative way of achieving COLREGs compliance would be to asess whether the OS is in 'Stand on' or 'Give way' mode, and modifying the cost function as well as dynamically placing constraints based on the situation. instead of just placing constraints that lead to 'immitating' COLREGs compliance.
- I wanted to test the algorithm agasint itself, by having all vessels in a situation be controlled by it's own instance. However the logic rewrite needed to conduct this test proved to be a bit too much

4.4 Improvements over Previous Version

- Definite improvements in terms of computational efficiency. This greatly increases the likelyhood of finding an optimal solution

- Because of the better efficiency the algorithm is also able to handle more control intervals, This means it is better at handling both greater time horizon and shorter control interval steps.
- The new method for handling static obstacles is much less prone to misplaced or inefficnet constraints. (her ta gjerne med figuren som viser problemer med sirkel constraints for statiske hindringer).
- The new way of handling dynamic constraints should in theory make the algorithm better suited for more complex situations with more agents, however the placement of dynamic constraints remains largely unchanged. Dynamic Constraint placement is bigger 'bottleneck' than agent culling for how complex situations are handled.
- More robust when an encounter leads to an infeasible solution.
- Improved COLREGs assessment
- But does it behave *noticeably* better? Yes.

5 Conclusion and Future Work

Conclusion

- Summary of results, obviously.
- Compare with the 'problem description', have I succesfully contributed to anything?
- Final thoughts on my own work

Future Work

- More work needed for optimizing placement of constraint, as well as when constraints need to be active.
- More work needed for situational awareness adjustion of parameters
- more work needed to examine scenarios with and without full TS prediction; more velocities, more variation of vessel sizes, more diverse enviornments.
- More work related to a variable cost function, a more adaptable cost function could for example yield better COLREGs compliance.
- More work related to Optimizing runtime of algorithm, tuning the IPOPT tolerances to balance computation speed and desired behaviour.
- More work related to tuning COLREGs compliance, testing more COLREGs situations and quantifying what constitutes good behaviour.
- Extracting the Algorithm from the MATLAB simulator it's built into and making a more stand-alone / generic software or algorithm.
- And more that I will think about as I write.

References

Andersson, Joel AE, Joris Gillis, Greg Horn, James B Rawlings and Moritz Diehl (2019). 'CasADi: a software framework for nonlinear optimization and optimal control'. In: *Mathematical Programming Computation* 11.1, pp. 1–36.

Cho, Yonghoon, Jungwook Han and Jinwhan Kim (2018). 'Intent inference of ship maneuvering for automatic ship collision avoidance'. In: *IFAC-PapersOnLine* 51.29, pp. 384–388.

Cockcroft, AN and JNF Lameijer (2012). *Guide to the Collision Avoidance Rules*. Oxford: Butterworth-Heinemann. Cockcroft, AN and Lameijer, JNF.

Eriksen, H. Bjørn-Olav and Morten Breivik (2017). 'MPC-based mid-level collision avoidance for ASVs using nonlinear programming'. In: *2017 IEEE Conference on Control Technology and Applications (CCTA)* (Mauna Lani Bay Hotel). IEEE. Hawaii, USA, pp. 766–772.

Fossen, Thor I (2011). *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons.

Gros, Sebastien (2017). *Numerical optimal control, lecture 4: Shooting methods*. Video lecture. URL: https://www.youtube.com/watch?v=UqWRcbdwPP8.

Huang, Yamin, Linying Chen, Pengfei Chen, Rudy R Negenborn and PHAJM Van Gelder (2020). 'Ship collision avoidance methods: State-of-the-art'. In: *Safety science* 121, pp. 451–473.

IMO (1972). *International Regulations for Preventing Collisions at Sea*. Wikisource Archive. URL: https://en.wikisource.org/wiki/International_Regulations_for_Preventing_Collisions_at_Sea.

Kufoalor, D. K.M., E. F. Brekke and T. A. Johansen (2018). 'Proactive Collision Avoidance for ASVs using A Dynamic Reciprocal Velocity Obstacles Method'. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2402–2409. DOI: 10.1109/IROS.2018.8594382.

Lekkas, Anastasios M and Thor I Fossen (2013). 'Line-of-sight guidance for path following of marine vehicles'. In: *Advanced in marine robotics*, pp. 63–92.

Loe, Øivind (2007). 'Collision avoidance concepts for marine surface craft'. In: *Specialization project report. Norwegian University of Science and Technology (NTNU), Trondheim, Norway*.

Park, Shinkyu, Michal Cap, Javier Alonso-Mora, Carlo Ratti and Daniela Rus (2020). 'Social Trajectory Planning for Urban Autonomous Surface Vessels'. In: *IEEE Transactions on Robotics* 37.2, pp. 452–465.

Pedersen, Anders Aglen (2019). 'Optimization based system identification for the milliAmpere ferry'. MA thesis. NTNU.

Qin, S Joe and Thomas A Badgwell (1997). 'An overview of industrial model predictive control technology'. In: *AIche symposium series*. Vol. 93. 316. New York, NY: American Institute of Chemical Engineers, 1971-c2002., pp. 232–256.

Schöller, Frederik ET, Thomas T Enevoldsen, Jonathan B Becktor and Peter N Hansen (2021). 'Trajectory prediction for marine vessels using historical AIS heatmaps and long short-term memory networks'. In: *Proceedings of 13th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles Elsevier*. Vol. 54. 16. IFAC. Oldenburg, Germany: Elsevier, pp. 83–89.

Tam, CheeKuang and Richard Bucknall (2010). 'Collision risk assessment for ships'. In: *Journal of Marine Science and Technology* 15.3, pp. 257–270.

Thyri, Emil Hjelseth and Morten Breivik (2022). 'A domain-based and reactive COLAV method with a partially COLREGs-compliant domain for ASVs operating in confined waters'. In: *Field Robotics* 2, pp. 632–677. DOI: https://doi.org/10.55417/fr.2022022.

Vagale, Anete, Rachid Oucheikh, Robin T Bye, Ottar L Osen and Thor I Fossen (2021). 'Path planning and collision avoidance for autonomous surface vehicles I: A review'. In: *Journal of Marine Science and Technology* 26, pp. 1292–1306.

Wächter, Andreas and Lorenz T Biegler (2006). 'On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming'. In: *Mathematical programming* 106.1, pp. 25–57.

Woerner, Kyle (2016). 'Multi-contact protocol-constrained collision avoidance for autonomous marine vehicles'. PhD thesis. Massachusetts Institute of Technology, USA.

Wright, Stephen, Jorge Nocedal et al. (1999). 'Numerical optimization'. In: *Springer Science* 35.67-68, p. 7.

Zhang, Xinyu, Chengbo Wang, Lingling Jiang, Lanxuan An and Rui Yang (2021). 'Collision-avoidance navigation systems for Maritime Autonomous Surface Ships: A state of the art survey'. In: *Ocean Engineering* 235, p. 109380.