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.

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

COLREGs Convention on the International Regulations for Preventing Collisions at Sea

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:

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.

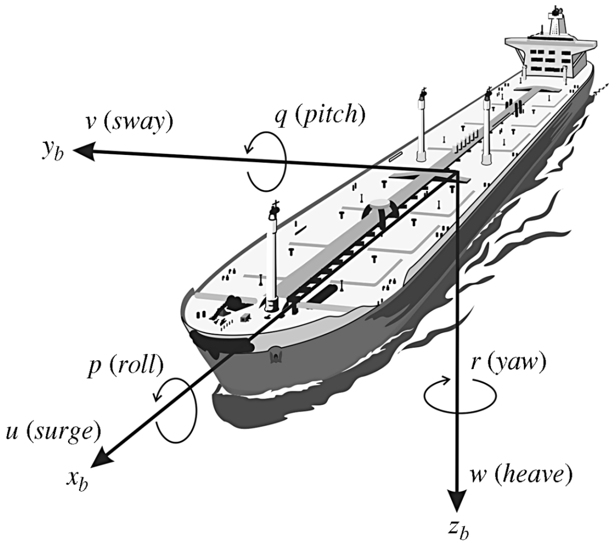


Figure 1: A ships 6 degrees of Freedom, from Fossen 2011

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 that the reader will have 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 seperate 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 seperate out and model the dynamics of is often what seperates 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 tangental plane North East Down (NED) where the x-axis points towards true north, the y-axis east and the z-axis down towards the center of the planet. The NED frame can be said to be inertial for short distance control objectives. The vector $\boldsymbol{\eta}$ is a column vector with the vessel's North, East and Heading values, which are the three DOF of the system. The $\boldsymbol{\nu}$ vector is a column vector containing the vessel's velocities parameterized in the BODY frame, namely surge, sway, and yaw rate. 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. $\mathbf{R}$ is a rotational matrix about the heading (ψ) of the vessel and it transform the BODY velocities into NED movement. The Rotation matrix, as well as pose, velocity, and the thruster vector $\boldsymbol{\tau}$ are:

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

$$\boldsymbol{\eta} = \begin{bmatrix} x & y & \psi \end{bmatrix}^T \tag{2.4}$$

$$\boldsymbol{\nu} = \begin{bmatrix} u & v & r \end{bmatrix}^T \tag{2.5}$$

$$\boldsymbol{\tau} = \begin{bmatrix} F_X & F_Y & F_N \end{bmatrix}^T \tag{2.6}$$

In 2.2 the **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 **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 **D**. That is because a trajectory planning algorithm needs to work regardless of vessel parameters, Pedersen 2019 explains more in-depth how the 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:

$$\mathbf{M} = \begin{bmatrix} m11 & m12 & m13 \\ m12 & m22 & m23 \\ m31 & m32 & m33 \end{bmatrix} \tag{2.7}$$

$$\mathbf{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & c13(\boldsymbol{\nu}) \\ 0 & 0 & c23(\boldsymbol{\nu}) \\ c31(\boldsymbol{\nu}) & c32(\boldsymbol{\nu}) & 0 \end{bmatrix} \tag{2.8}$$

$$\mathbf{D}(\boldsymbol{\nu}) = \begin{bmatrix} d11(\boldsymbol{\nu}) & 0 & 0 \\ 0 & d22(\boldsymbol{\nu}) & d23(\boldsymbol{\nu}) \\ 0 & d32(\boldsymbol{\nu}) & d33(\boldsymbol{\nu}) \end{bmatrix} \tag{2.9}$$

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.

$$\mathbf{D}(\boldsymbol{\nu}) = \begin{bmatrix} d11 & 0 & 0 \\ 0 & d22 & 0 \\ 0 & 0 & d33 \end{bmatrix} \tag{2.10}$$

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(TODO: Citation needed?). 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

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 vessel 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. Or the method can be incredibly complex and and computationally involved, 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 controlled vessel 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 controlled vessel's position (x,y), the cross track and along track errors from the straigth line as defined by $\mathrm{WP_k}$ $(\mathrm{x_k,y_k})$ and $\mathrm{WP_{k+1}}$ $(\mathrm{x_{k+1}}, \mathrm{y_{k+1}})$ are:

$$\begin{bmatrix} x_e \\ ye \end{bmatrix} = \mathbf{R}^T(\gamma_p) \begin{bmatrix} x - x_k \\ y - y_k \end{bmatrix} \tag{2.11}$$

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. The rotation matrix $\mathbf{R}$ is given by:

$$\mathbf{R}(\gamma_p) = \begin{bmatrix} cos(\gamma_p) & -sin(\gamma_p) \\ sin(\gamma_p) & cos(\gamma_p) \end{bmatrix} \tag{2.12}$$

with γ_p:

$$\gamma_p = \text{atan2}(y_{k+1} - y_k, x_{k+1} - x_k) \tag{2.13}$$

The control objective is to drive $y_e(t) \rightarrow 0.$ as t trends towards infinity. Assuming a steady velocity this is done by selecting a course that steers the controlled vessel in the direction that reduces y_e. How fast the error y_e is suppressed is tuned by a proportional gain factor, Δ, that is often called look ahead distance. The desired heading is given by:

$$\psi_d = \gamma_p + \arctan(\frac{-y_e}{\Delta}) \tag{2.14}$$

and consequently desired course:

$$\chi_d = \psi_d + \beta \tag{2.15}$$

Where β is the sideslip of the controlled vessel. 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:

$$\dot{y}_{int} = y_e \tag{2.16}$$

$$\psi_d = \gamma_p - \arctan(K_p y_e + K_i \dot{y}_{int}) \tag{2.17}$$

Where $\mathrm{K_p}$ and $\mathrm{K_i}$ 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 Own Ship (OS) 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 OS is within a certain distance from it's current target waypoint. The reference trajectory from t_0 to N iterations of LOS applications is of the form:

$$\overline{\boldsymbol{\eta}}_{ref} = [\boldsymbol{\eta}_{t0}\ ,\ \boldsymbol{\eta}_{t+1}\ ,\ \ldots\ ,\boldsymbol{\eta}_N] \tag{2.18a}$$

$$\overline{\boldsymbol{\nu}}_{ref} = [\boldsymbol{\nu}_{t0}\ ,\ \boldsymbol{\nu}_{t+1}\ ,\ \ldots\ ,\boldsymbol{\nu}_N] \tag{2.18b}$$

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:

$$\underset{x \in \mathbf{R}^n}{\text{Minimize}} \quad f(x) \tag{2.19a}$$

$$\text{Subject to:} \quad c_i(x) = 0, \quad i \in \mathcal{E} \tag{2.19b}$$

$$c_i(x) \geq 0, \quad i \in \mathcal{I} \tag{2.19c}$$

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:

$$\text{Minimize} \quad L(\boldsymbol{\theta}(t), \boldsymbol{\theta}_{ref}(t), \boldsymbol{\tau}(t)) \tag{2.20a}$$

$$\text{Subject to:} \quad \dot{\boldsymbol{\theta}}(t) = \mathbf{J}(\boldsymbol{\theta}, \boldsymbol{\tau}) \tag{2.20b}$$

$$\mathbf{h}(\boldsymbol{\theta}(t), \boldsymbol{\tau}(t)) \leq 0 \tag{2.20c}$$

$$\boldsymbol{\theta}(t_0) - \overline{\boldsymbol{\theta}}_0 = 0 \tag{2.20d}$$

where L is the cost function, $\boldsymbol{\theta} = [\boldsymbol{\eta}^T \ , \ \boldsymbol{\nu}^T]^T$ and $\boldsymbol{\tau}$ is the same as in 2.6. $\mathbf{J}(\boldsymbol{\theta}, \boldsymbol{\tau})$ 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 $\mathrm{t}_{\mathrm{end}}$. And L has the form of a quadratic function akin to a weighted least squares: (TODO: Eh, grei formulering?)

$$L(\boldsymbol{\theta}(t), \boldsymbol{\theta}_{ref}(t), \boldsymbol{\tau}(t)) = (\boldsymbol{\theta}(t) - \boldsymbol{\theta}_{ref}(t))^T \mathbf{Q} (\boldsymbol{\theta}(t) - \boldsymbol{\theta}_{ref}(t)) + \mathbf{K}_\tau \boldsymbol{\tau}^2 \tag{2.21}$$

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 2.20 where L is redefined as a discretized cost function with N_p control intervals steps:

$$\Phi(\boldsymbol{\omega}, \boldsymbol{\omega}_{ref_{1:N_p}}) = \sum_{k=0}^{N_p-1} ((\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_{ref_{k+1}})^T Q(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_{ref_{k+1}}) + K_{\boldsymbol{\tau}} \boldsymbol{\tau}_{\boldsymbol{k}}^2) \tag{2.22}$$

where $\boldsymbol{\omega} = [\boldsymbol{\theta}_0^T \ , \ \boldsymbol{\tau}_0^T \ , \ \ldots \ , \ \boldsymbol{\theta}_{N_{p-1}}^T \ , \ \boldsymbol{\tau}_{N_{p-1}}^T \ , \ \boldsymbol{\theta}_{N_p}^T]^T \in \mathbb{R}^{9N_p+6}$ is a vector containing $5N_p+6$ decision variables.Because $\boldsymbol{\tau}_k$ does not have an accociated reference in this thesis; it is seperated out as it's own part of the function. $\mathbf{Q}$ 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:

$$\min_{\boldsymbol{\omega}} \quad \boldsymbol{\Phi}(\boldsymbol{\omega}, \boldsymbol{\omega}_{ref_{1:N_p}}) \tag{2.23a}$$

$$\text{Subject to:} \quad \boldsymbol{\omega}_{lb} \leq \boldsymbol{\omega} \leq \boldsymbol{\omega}_{ub} \tag{2.23b}$$

$$\mathbf{g}(\boldsymbol{\omega})_{lb} \leq \mathbf{g}(\boldsymbol{\omega}) \leq \mathbf{g}(\boldsymbol{\omega})_{ub} \tag{2.23c}$$

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

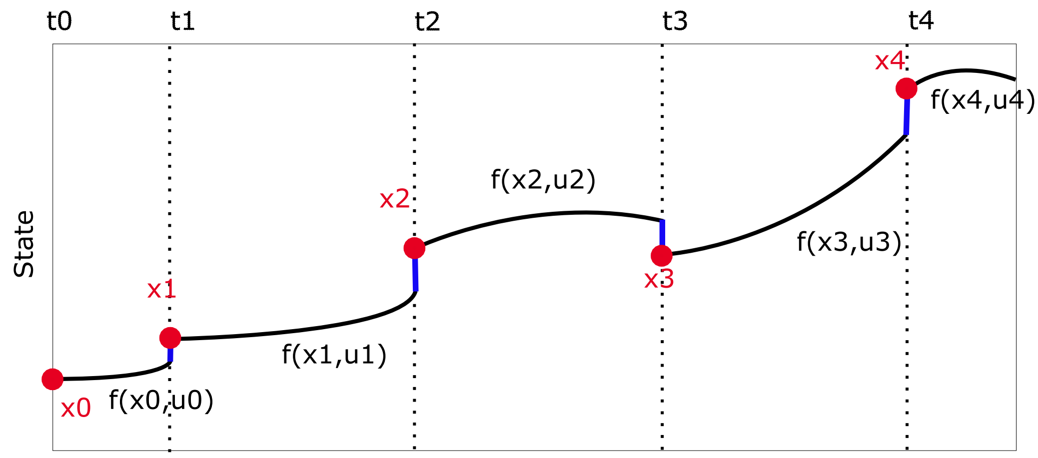


Figure 2: A physically feasible trajectory is formed by "pinching" the shooting gaps close. Reproduction from Gros 2017

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 2. 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:

$$\begin{aligned} k_1 &= \mathbf{f}(\boldsymbol{\theta}_k \ , \ \boldsymbol{\tau}_k) \\ k_2 &= \mathbf{f}(\boldsymbol{\theta}_k + \frac{h}{2}k_1 \ , \ \boldsymbol{\tau}_k) \\ k_3 &= \mathbf{f}(\boldsymbol{\theta}_k + \frac{h}{2}k_2 \ , \ \boldsymbol{\tau}_k) \\ k_4 &= \mathbf{f}(\boldsymbol{\theta}_k + hk_3 \ , \ \boldsymbol{\tau}_k) \\ \mathbf{F}(\boldsymbol{\theta}_k, \boldsymbol{\tau}_k) &= \boldsymbol{\theta}_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{2.24}$$

Where h is the 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:

$$\mathbf{g}(\boldsymbol{\omega}) = \begin{bmatrix} \bar{\boldsymbol{\theta}}_0 - \boldsymbol{\theta}_0 \\ \mathbf{F}(\boldsymbol{\theta}_0, \boldsymbol{\tau}_0) - \boldsymbol{\theta}_1 \\ \mathbf{F}(\boldsymbol{\theta}_1, \boldsymbol{\tau}_1) - \boldsymbol{\theta}_2 \\ \vdots \\ \mathbf{F}(\boldsymbol{\theta}_{N_{p-1}}, \boldsymbol{\tau}_{N_{p-1}}) - \boldsymbol{\theta}_{N_p} \end{bmatrix} \tag{2.25}$$

Setting the lower and upper bounds for **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 **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 grants the control system an opportunity to react and adjust to unexpected disturbances, which greatly increases robustness of the automated system. (TODO: citation needed?)

2.3 Collision Avoidance

- Convention on the International Regulations for Preventing Collisions at Sea (COLREGs), cite IMO 1972 I guess.
- Classification Tam and Bucknall 2010.
- Modification & Figur er simpelt: Courtesy of Emil Thyri.
- Alternative classification algorithm Woerner 2016.
- dCPA og tCPA, Kufoalor et al. 2018.
- My modifications to dCPA and tCPA method to get 'full coverage' of path.
- Static Obstacles "detection" and constraint placements
- Dynamic obstacles constraints.
- Computer vision, radar/lidar/sonar/etc. Is not a part of this thesis, see for example Ruud et al. 2018 for that topic.
- Maneuvers for avoiding collision Cockcroft and Lameijer 2012
- No right of way, only Give Way or Stand On.
- cite Cho et al. 2018 as a method for identifying intent, and a lead in to target ship prediction.

2.4 Target Ship prediction

——— Genereal thoughts —————————

- Gjenfortelling fra fordypningsprosjekt, da kalt traffic pattern
- Fant en annen artikkel fra Kina som skrev om nogenlunde det samme, AIS data - prediksjon
- Skiller seg fra fordypningsprosjekt fordi det er egentlig ikke traffic pattern som er den viktige antagelsen, Det er heller viktig at vi antar det finnes en måte å gjette/vite hvor andre båter vil være fremover i tid.
- Andre metoder for target ship prediction kan være f.eks utvidelse av AIS som inkluderer autonav data for de neste 5 minuttene eller noe lignende.
- Disambiguate Simple and Full prediction.

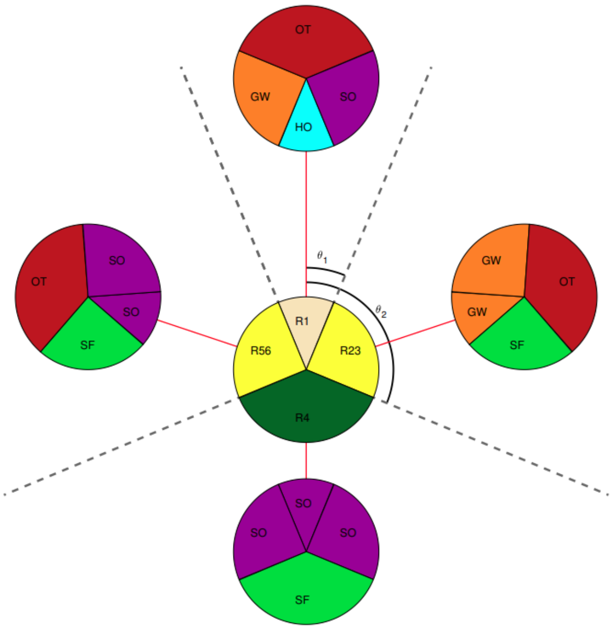


Figure 3: (TODO:ENDRE)Assigning COLREGS flag: with OS in the center we can place the TS in one of four regions. Similarly the relative bearing from TS to OS can be assigned regions with region 1 pointed directly at the OS and the rest following in a clockwise rotation. Courtesy of Emil Thyri.

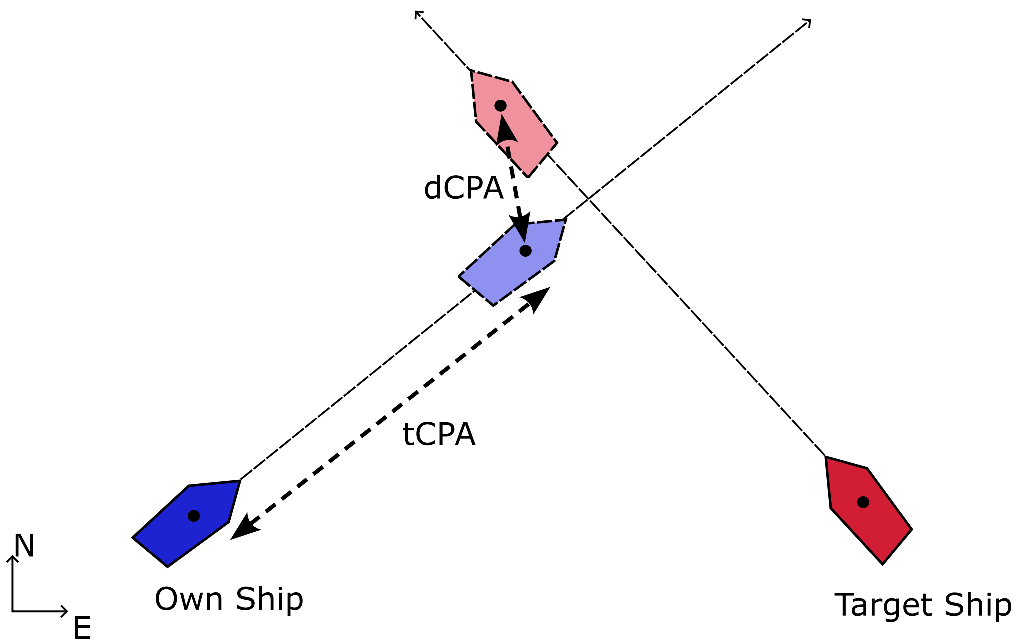


Figure 4: Visualizing dCPA and tCPA.

——— /General Thoughts———————

- Naval navigation is an 'active' task, always looking out for obstacles and making sure the way forward is clear.
- There are no lanes, instead proper conduct is dictated by COLREGs, the rules are laid out so that different situations have different rules
- Knowing which situation you are in is half the battle, therefor the ability to predict and estiamte how encounters will happen is a powerful tool. Human navigators do this by experience.
- Autonomous vessels could also predict by experience, that would be the machine learning approach. AIS can already provide training data.
- But it could be easier than that, what if AIS data packets were expanded to send out autonav data for where a vessel intends to traverse. The assumption here is that vessels using autonav will correct their course when spotting a conflict. Vessels not using autonav would still be able to observe the indended path of other vessels and adjust their plans 'manually'.
- Relying on fully predicting the transit of a target vessel is fragile, relying on a few known waypoints would be much more robust.
- In the end the best solution would be if all vessels could fully communicate their intended path either peer to peer or through a centralized system. Solving conflicting paths would be done on a higher level than individual vessel.
- cite Schöller et al. 2021 for AIS prediction.
- cite Zhang et al. 2021 For mention of massive AIS data.
- cite ais?? har sett noen gjøre det.
- cite Huang et al. 2020 to refer to current methods for target ship prediction.

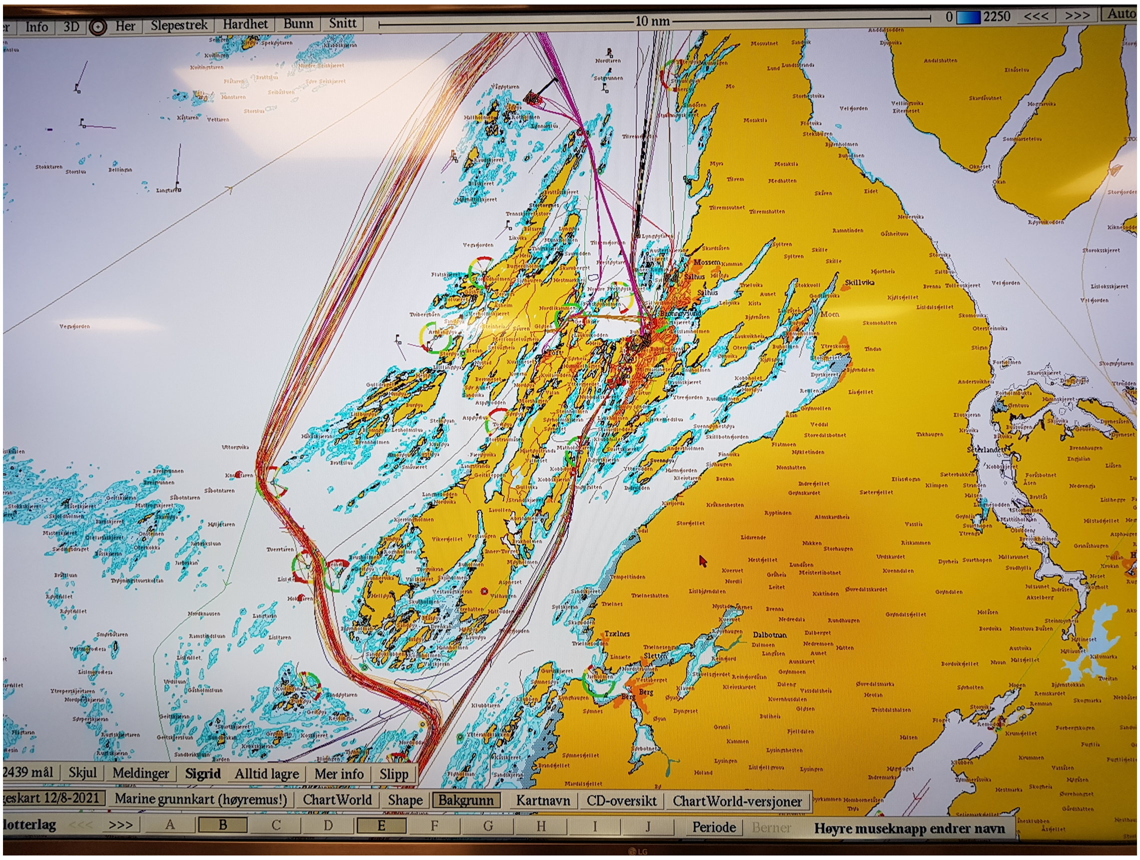


Figure 5: TODO: Skriv. AIS data can show common transit routes. Image courtesy of Olex AS

3 Trajectory planner

- Tidligere kjent som 'Method'.
- Har lyst å skrive litt om tankegangen bak utviklingen, ikke bare om hvordan ting endte opp med å bli.
- Ingen 'Preliminaries', alt av forkunnskaper og antagelser burde vært gjort rede for i 'Background'.
- Spesifikt mitt arbeid.
- Tar det fra start til slutt.

  Persistent variables & settings.

  COLREGs assessment.

  Dynamic Horizon.

  Casadi setup (generer F)

  Feasibility check.

Initial conditions and Reference LOS guidance.

NLP init.

Main loop, med alt som skjer der.

Solve NLP, give output.

- Bit for bit, forklar hva, hvorfor, hvordan, eventuellt andre versjoner eller ideer som ble prøvd.
- forklar informasjonsflyt, kanskje som eget delkapittel.
- My method is NOT a dock-to-dock system, the developed algorithm is NOT suitable for stationkeeping or docking maneuvers. Dock-To-Dock does exist see WÄRTSILÄ 2018, otherwise there are OTHER algorithms more suitable for docking. (TODO: FIND SOMETHING TO CITE)

(TODO: skriv bedre og mer korrekt.) This chapter presents the development of the trajectory planning and collision avoidance algorithm, explaining the design decisions made and analyzing some of the problems that arose during development. First the general dataflow of the algorithm is explained so that an intuition is formed as for how the individual parts are connected. Second a piece by piece construction of the algorithm up until the construction of the NLP. Lastly the NLP is constructed and solved using framework provided by CasADi. The core design of the algorithm is that path following is done through numerical optimization of a cost function, whilst collision avoidance and safety is implemented as hard constraints in the NLP, together an optimal trajectory is formed.

3.1 Dataflow

The dataflow is depicted in Figure 6, 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, as well as information about other ships (called Tracks in the diagram), static obstacles and miscellaneous other settings. These structs need to contain data in some specific fashions which will be discussed more deeply later in this chapter.(TODO: This part). The Tracks structure as it arrives from the higher lever system is assumed to have full prediction, the algorithm can simplify the prediction for testing purposes but there is no internal logic to switch between prediction detail level. Regardless of if the tracks were simplified or not they are parsed through a COLREGs assessment algorithm

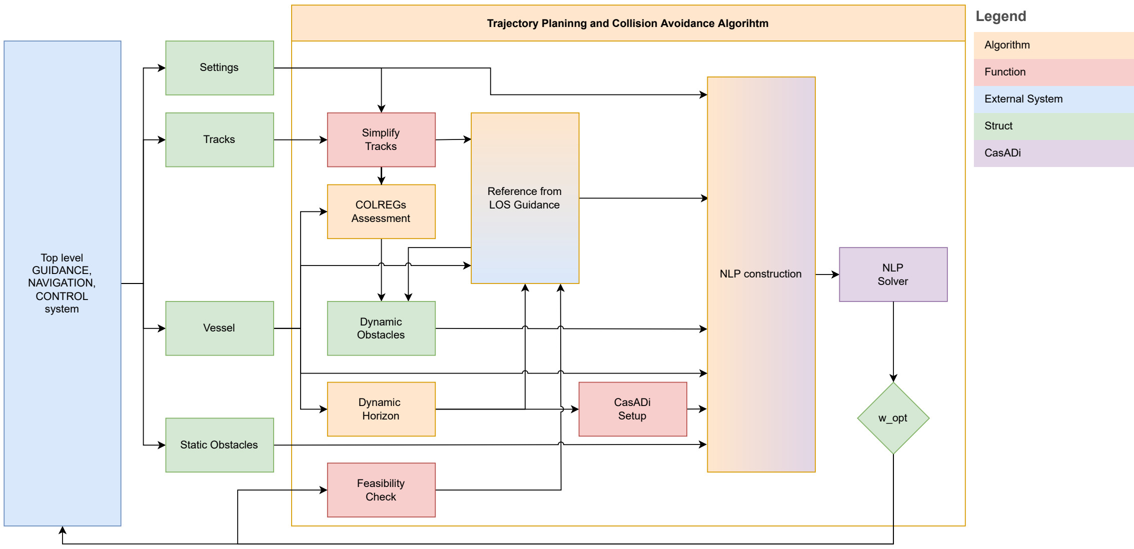


Figure 6: The simple version of the dataflow...

to determine which TSs need to be considered active obstacles, as well as finding how long until the time to Closest Point of Approach (tCPA), and which COLREGs situation the encounters would be. Independently of the creation of dynamic obstacles we use the information about our own vessel's position and end goal to determine how long the time horizon for this instance of the algorithm should be. When time horizon and discretization step size is known the CasADi setup can be ran to generate the function 'F' which will be used during the construction of the NLP later. With a dynamic horizon in place a LOS algorithm can be used to generate the final piece of the dynamic obstacles struct; the trajectory of the TSs themselves, which are used to place constraints later in the NLP construction. One crucial detail to take note of is that COLREGs assessment of a TS is conducted using only the waypoints provided by the Tracks struct, while the placement of dynamic constraints are based on the trajectory generated by a LOS guidance law. The feasibility check is not ran in the first instance of the algorithm after starting up the system. The result of the feasibility check is used when generating our OS's reference path to determine if the desired velocity should be reduced. the LOS guidance returns as mentioned the positions for the dynamic obstacles constraints as well as the reference positions and velocities in NED.

The NLP can now be initialized. Initial position and velocity forms the first constraint of the NLP (TODO: SKRIV MER)

3.2 Setup

- All the stuff before main loop.
- subsubsection for each 'block' as outlined by the dataflow.

- when the trajectory planner is called we need to run through some calculations before constructing the NLP problem
- These calcualtions are a mix of situation analysis, simulation settings, and CasADi initialization.
- Some of these calculations could be redundant in a complete control and navigation system, where other modules of the system would calculate the same thing.
- It's also important to remember that the value of many parameters are just guesswork, many of the subfunctions would benefit from a more sophesticated design that are tuned based on the situation the vessel finds itself in.

3.2.1 Simplify Prediction

This part of the setup is only required in simulations, the aim is to emulate the 'standard' way target ship (TS) prediction is conducted, which is to say constant course and velocity [TODO: Citation needed]. The TS trajectory is changed so that the first waypoint is the current position of the ship, and the next waypoint is one nautical mile in the direction of the ships heading. Ideally course over ground would be used instead of heading, however in the simulator crab angle and sideslip are not accounted for, therefor heading and course are the same angle. Excess waypoints stored in the TS struct are also truncated and the current waypoint index is forcefully set to 1 to prevent index out of range type errors.

3.2.2 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 [TODO: cite paper], COLREGs assessment is also explained in [TODO: fordypningsoppgaven].

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.

$$t_{AB}^{CPA} = \begin{cases} \frac{\mathbf{P}_{BA} \cdot \mathbf{V}_{A|B}}{||\mathbf{V}_{A|B}||^2} & \text{if } ||\mathbf{V}_{A|B}|| > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.1a}$$

$$d_{AB}^{CPA} = ||(\mathbf{P}_A + t_{AB}^{CPA}\mathbf{V}_A) - (\mathbf{P}_B + t_{AB}^{CPA}\mathbf{V}_B)|| \tag{3.1b}$$

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 vessel A and vessel B parameterized in NED. However if we are to utilize the advanced prediction we have on other agents a bit more logic must be applied to achieve full coverage of our intended path. Presume that our path contains a set of waypoints we intend to pass by, and similarly we know of a set of waypoints another agent intends to pass by. To find the true dCPA and tCPA between our own ship and the target vessel we use equations 3.1 as the situation is when each agent passes by one of their waypoints(TODO: TRENGER FIGUR FOR Å VISE BEDRE HVA JEG MENER). Speed is tougher to account for, unless we know better speed should be assumed to remain constant (TODO: dette hører hjemme i background).

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 3.1 are used. getCPAlist

3.2.3 Dynamic Horizon

- Dynamic horizon is a balancing act between distance to goal, encompassing all active dynamic obstacles, and not looking to far ahead into the future.
- changing the dynamic horizon is really just changing how many control intervals we want the NLP to have.
- As the distance to goal approaches zero we want the number of control intervals to shrink accordingly, otherwise we end up with too many control intervals stationary at the goal, which can cause problems like the cost function becoming unbalanced.
- If there are active dynamic obstacles we need the dynamic horizon to encompass them.

Algorithm 1 getCPAlist. Denne ble jævelig stygg, beholder den for synlighet

Input: $Agent1.Agent2$ ▷ Agent is a struct that includes path waypoints

$dCPAlist \leftarrow []$
$tCPAlist \leftarrow []$
$pos\_OS\_list \leftarrow []$
$pos\_TS\_list \leftarrow []$
$timer \leftarrow 0$ ▷ Initialize timer used to calculate position of Agent2
for $i \leftarrow Agent1.current\_wp : agent\_wplist\_length - 1$ **do**
    $[pos_{OS}, vel_{OS}] \leftarrow VesselReadout(Agent1, i)$ ▷ VesselReadout explained in algorithm...
    $DisttonextWP \leftarrow$Distance to Agent1's next waypoint
    $TimetonextWP \leftarrow DisttonextWP \div$ Agent1's speed over ground
    $[pos_{TS}, vel_{TS}] \leftarrow whereisTS(Agent2, Timer)$ ▷ whereisTS explained in algorithm...
    $[dCPA, tCPA] \leftarrow$ Equation for dCPA & tCPA as shown by...
    $tCPA \leftarrow tCPA + timer$ ▷ Add travel time to reach current wp
    $timer \leftarrow timer + TimetonextWP$
    $pos\_OS\_list \leftarrow [pos\_OS\_list, pos_{OS}]$ ▷ Append all values to respective list.
    $pos\_TS\_list \leftarrow [pos\_TS\_list, pos_{TS}]$
    $dCPAlist \leftarrow [dCPAlist, dCPA]$
    $tCPAlist \leftarrow [tCPAlist, tCPA]$
    $i \leftarrow i + 1$
end for
return $pos\_OS\_list, pos\_TS\_list, dCPAlist, tCPAlist$

- we don't want the dynamic horizon to be too short during transit (why not?).

3.2.4 CasADi setup

- sym x = [N, E, psi, u, v, r]
- sym tau as a free variable
- sym xref as reference
- model parameters
- M, C, D matrix
- xdot ̃[nu_dot, eta_dot]
- Error in the correct reference frame.
- Why is the cost function the way it is.
- runge-kutta method.
- the final function F that CasADi needs.
- Noe av disse greiene blir dekt av Background, forhåpentligvis.

3.2.5 Feasibility check

- The feasibility check came from the wish to read out the status report CasADi prints to the command window.
- It's very important to know if the previous iteration of the trajectory planner function yielded a feasible result or not.
- if the result is not feasible the path forward might be completely blocked, in which case reducing our vessel speed is the best option.
- Very simple check, just checks if every point in the previously calculated optimal trajectory is within 5 meters of each other. This is very lenient and should of course change depending on vessel speed.

3.2.6 Reference from LOS

- I didn't write this)
- The important part is that the time discretization is consistent with the trajectory planner's time step
- You don't need to use LOS for reference.
- Position reference and speed reference need to be consitent with each other.

3.3 NLP construction and solver

- inputs vessel, ref_trajectory, static_obs, dynamic_obs, F, settings, h, N, previous_w_opt.
- sub funksjoner

  Dynamic Obs.

  Static Obs.

  integration step.
- output w_opt

3.3.1 NLP initialization

- Initial conditions and end of interval coditions, we need the end of one control interval and the beginning of the next to match.

- Tror dette delkapittelet er litt unødvendig

3.3.2 Integration step

- Getting the correct reference
- make sure all the indexes are correct!
- put the references in w0, speeds up runtime significantly.

3.3.3 Dynamic Obstacles constraints

- When to place constraints
- Where to place them
- How to place them

3.3.4 Static Obstacles constraints

- Explain static_obstacles_check and the theory for convex-free set
- explain why circles, such as the ones used for dynamic obstacles are insufficient.
- this is sort of similar to finding a cross track error, if that helps to explain what is going on.

3.3.5 Solver

- Options, there are many options.
- things to try / were tried for optimizing runtime.
- CasADi really does all the hard work.

3.4 Alternative ideas and lessons

Burde kanskje heller gå under discussion, og igjen i future work.

- Change w0 based on previous solution runtime.
- Gamle versioner av Static_obs.
- eksperimenter med feasibility check.

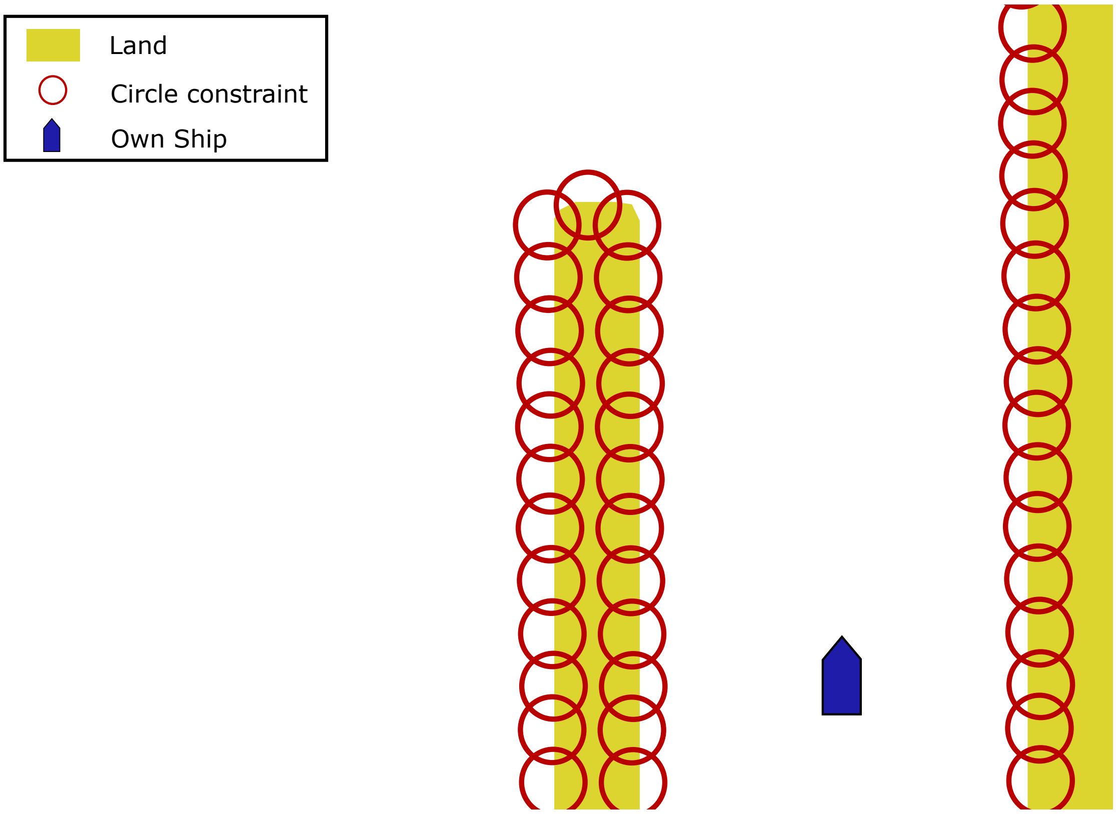


Figure 7: TODO: SKRIV OG REFERER. Naiv approach 1

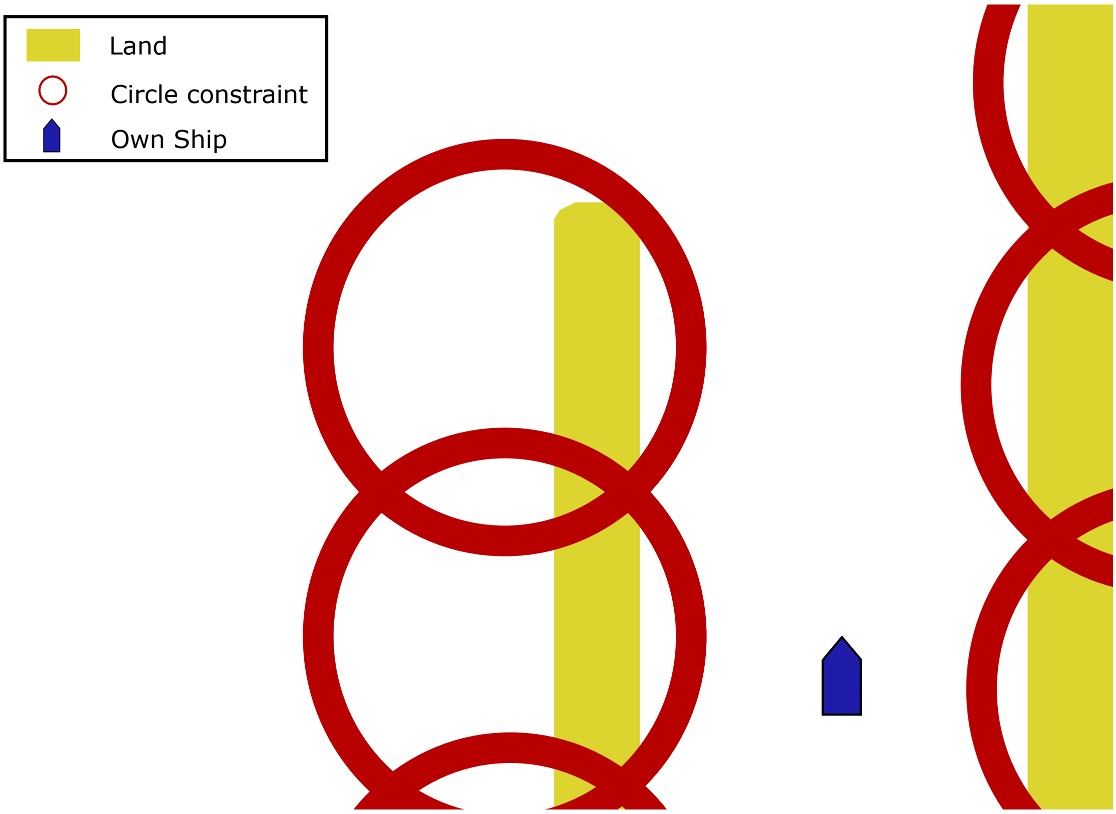


Figure 8: TODO: SKRIV OG REFERER. Naiv approach 2

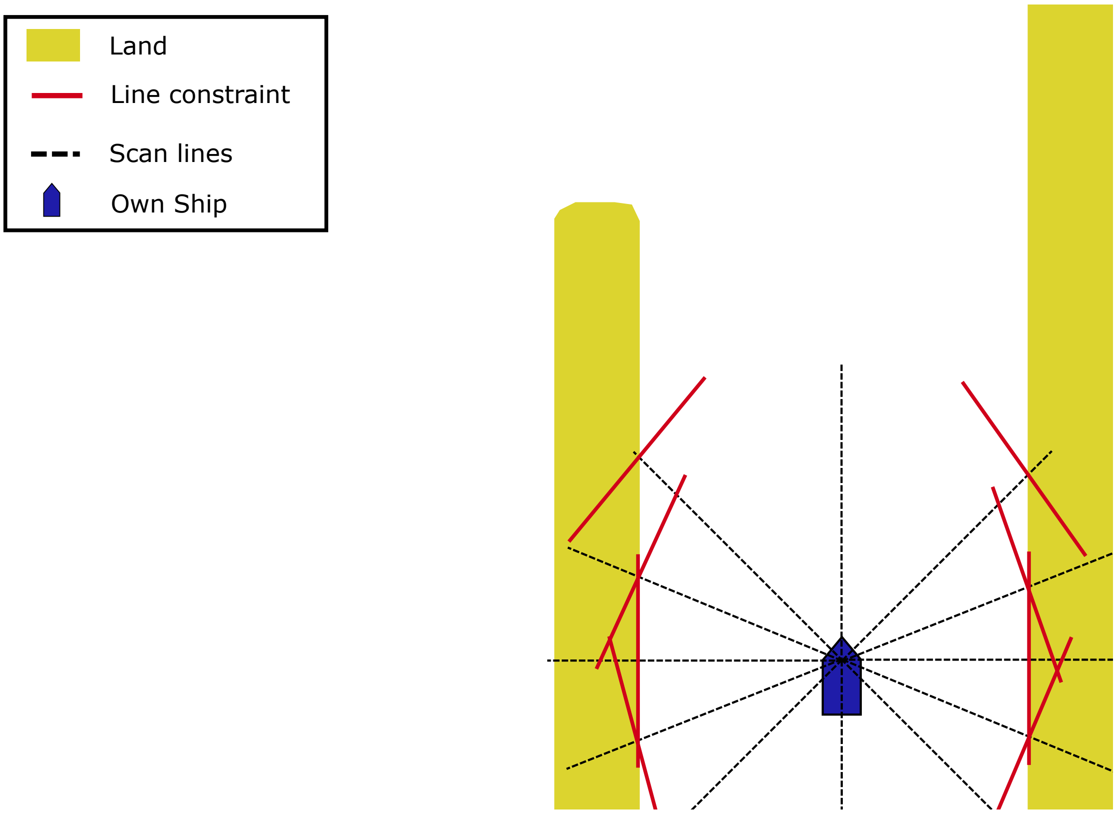


Figure 9: TODO: SKRIV OG REFERER. Convex free set

- Masse styr med COLREGs assessment, tcpa og dcpa.
- ipopt innstillinger.

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 algorithm. 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.

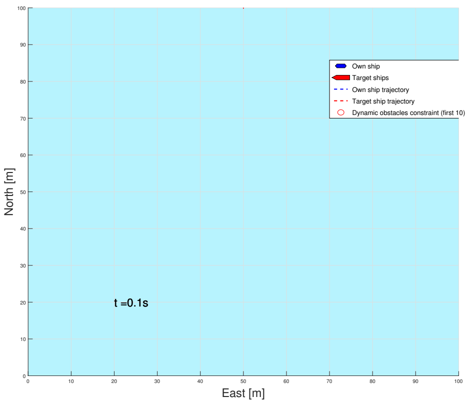
- The scale is not uniform across all simulations, sometimes the boats are scaled up to make the figures easier to read.
- The results are accompanied by MALTAB figures, as well as a youtube video that compiles all the results into a video which sampled the simulations every second.

4.2.1 Simple Head On

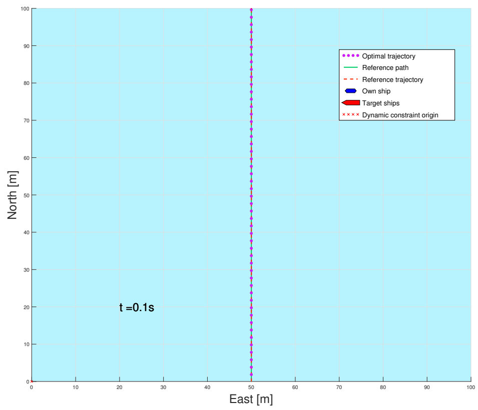
- Very straight forward result, behaviour is exactly as one would expect given the placement of the constraint.
- No difference between full and simple prediction because target ship holds steady course and velocity.
- This is the behaviour we can expect every time a target ship is met directly head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

4.2.2 Simple Give Way

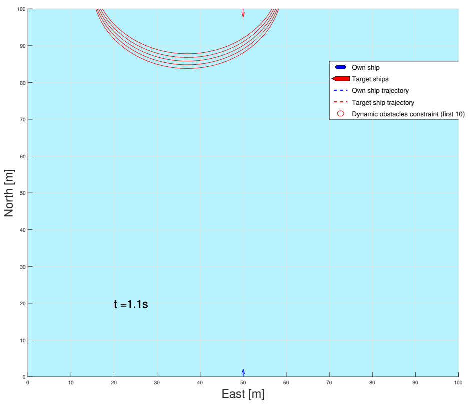
- It would be highly unusual to start using the trajectory planner when alredy this close to a situation
- This is reflected in the strange behaviour where our path is completely blocked when dynamic obstacle constrains are enabled.



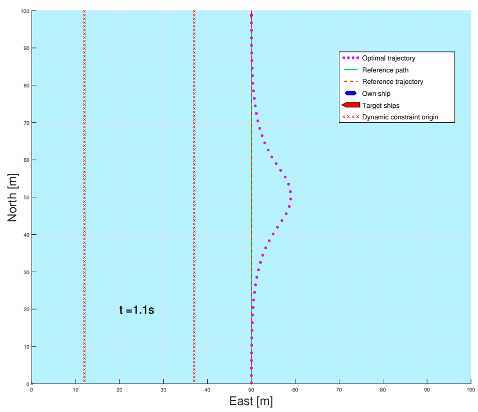
(a) caption



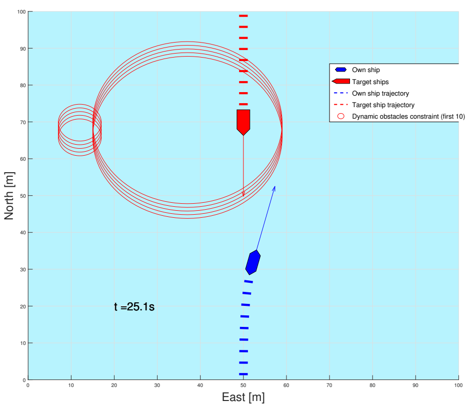
(b) mhm



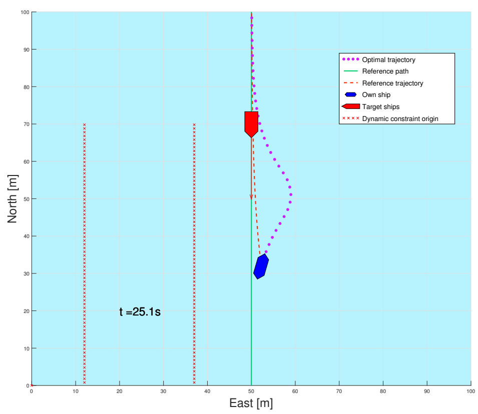
(c) caption



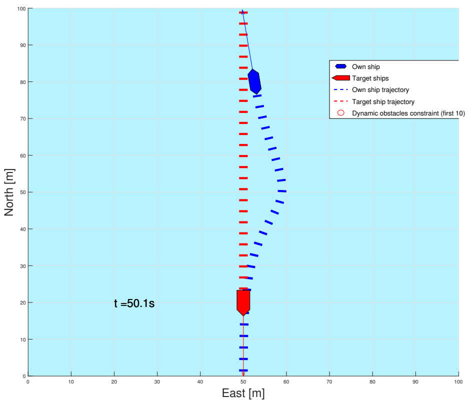
(d) mhm



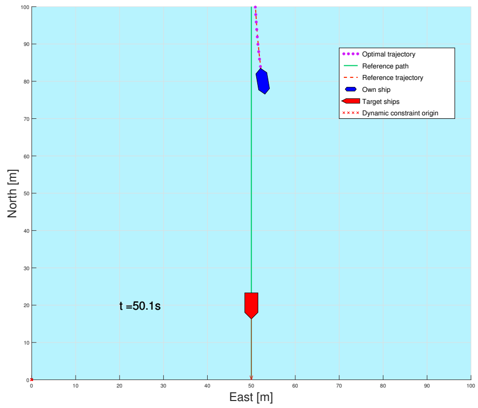
(e) caption



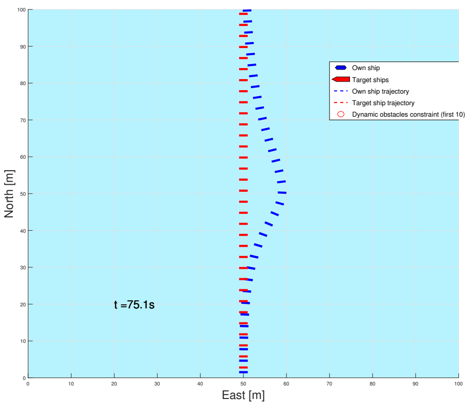
(f) mhm



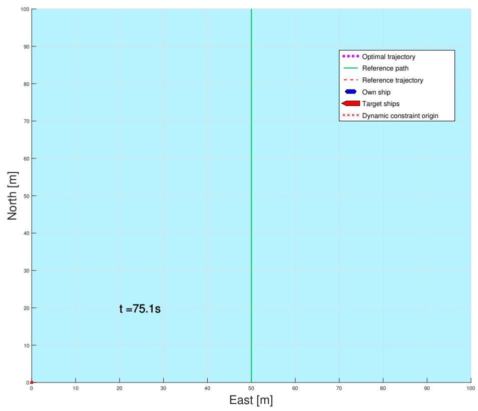
(g) caption



(h) mhm



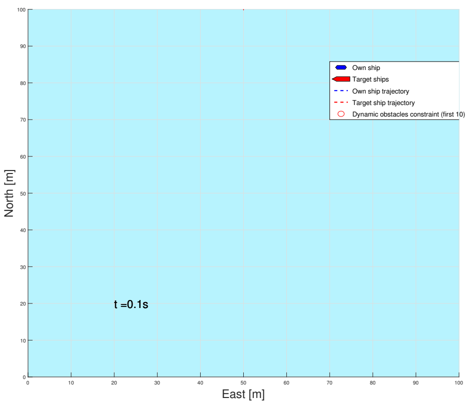
(i) caption



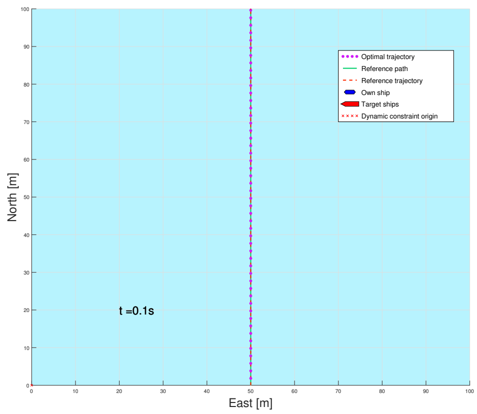
(j) mhm

Figure 10: Simple Head on With Prediction

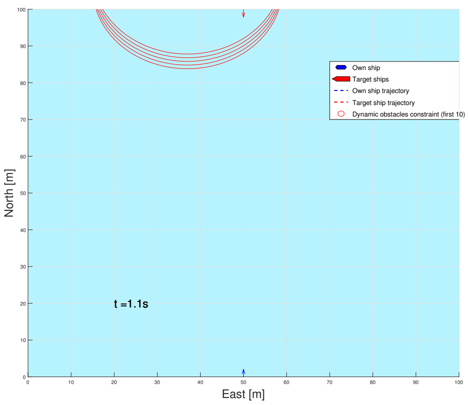
- otherwise the behaviour ends up being exactly as expected considering the placements of the constraints.



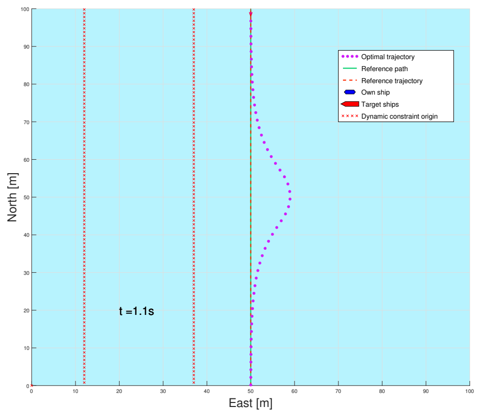
(a) caption



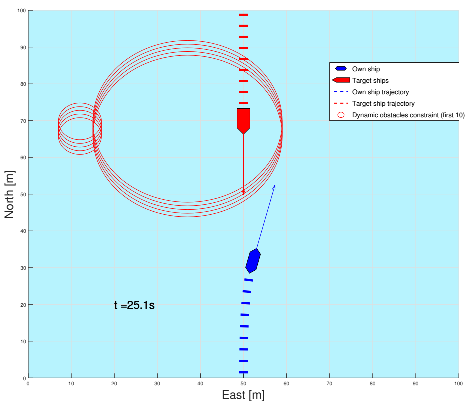
(b) mhm



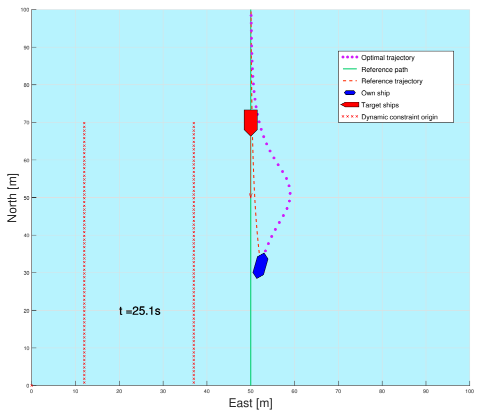
(c) caption



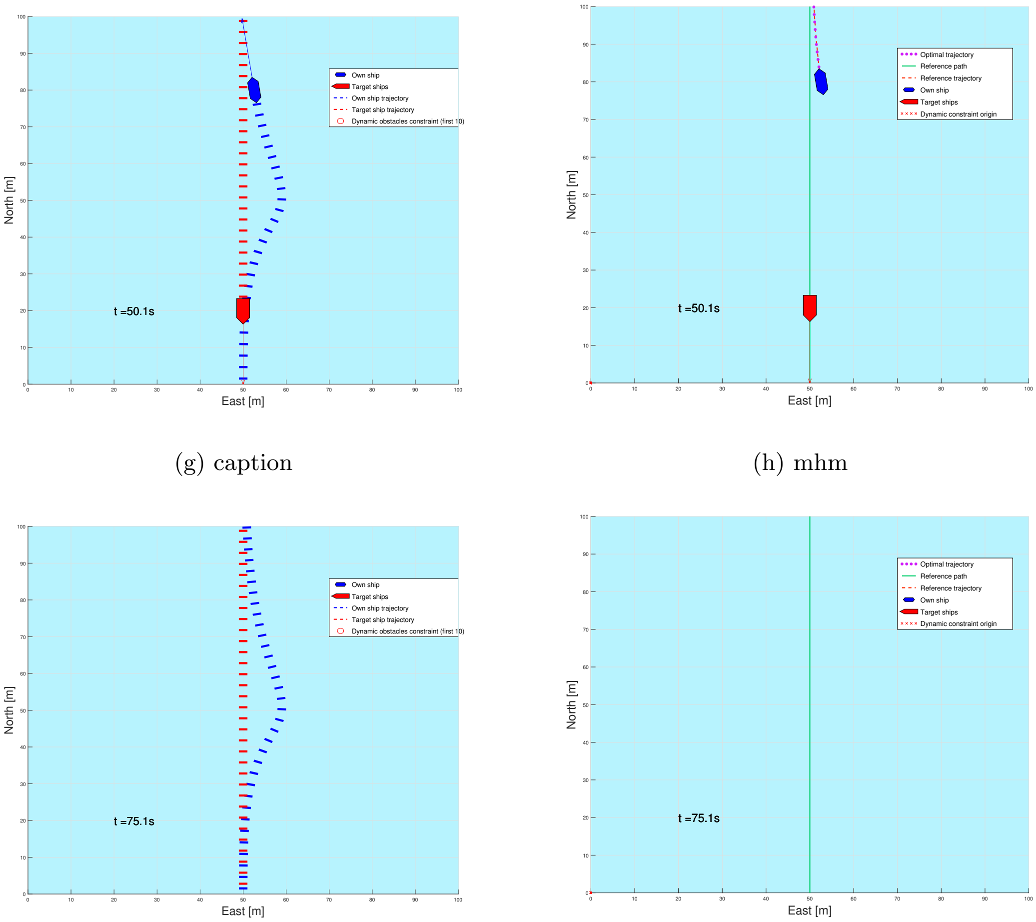
(d) mhm



(e) caption



(f) mhm



(g) caption

(h) mhm

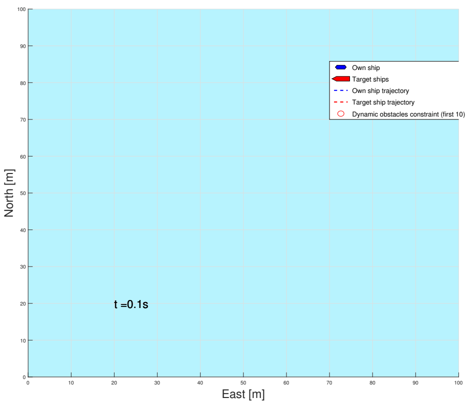
(i) caption

(j) mhm

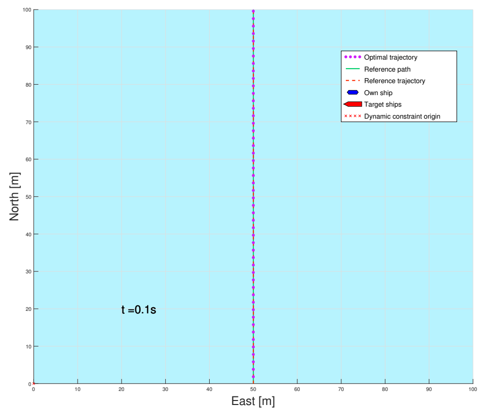
Figure 11: Simple Head on Without Prediction

4.2.3 Simple Stand On

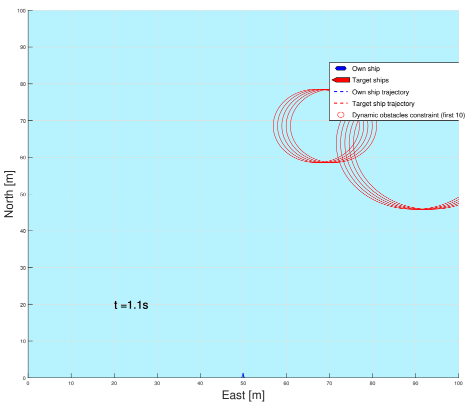
- This scenario assumes that the TS plays nice and attempts to follow the COLREGs rules.
- When using simple prediction the optimal path is pushed towads port side behind the TS, and as the TS begins to turn the OS is dragged along by the movement of the constraints.



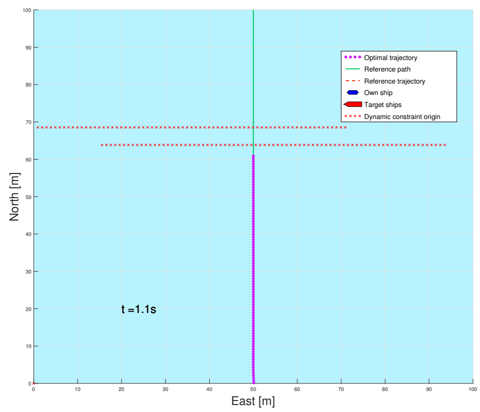
(a) caption



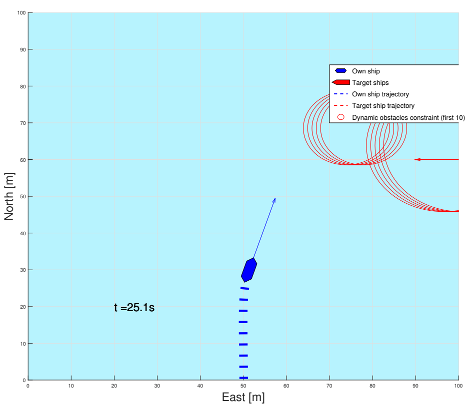
(b) mhm



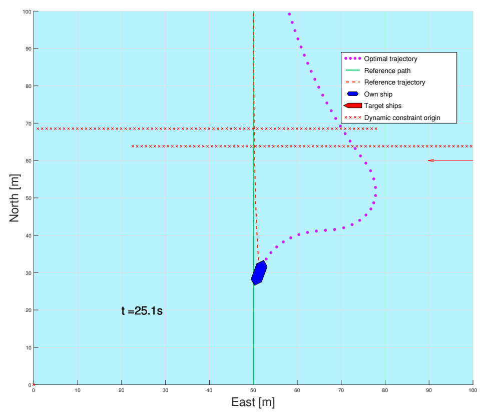
(c) caption



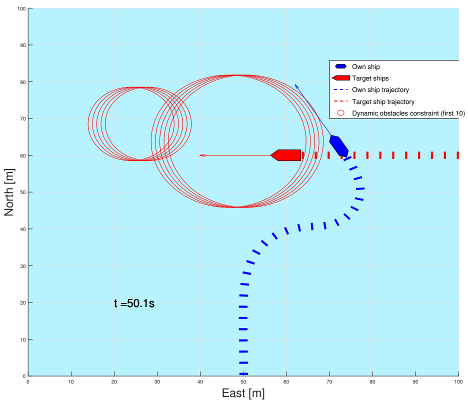
(d) mhm



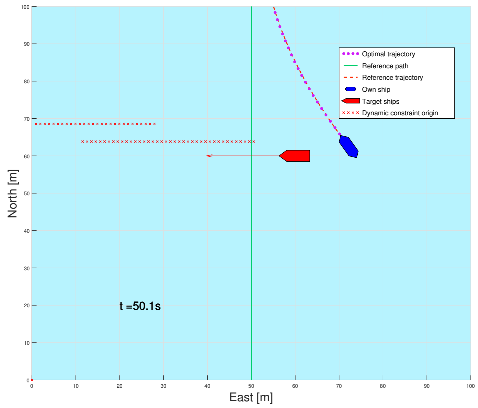
(e) caption



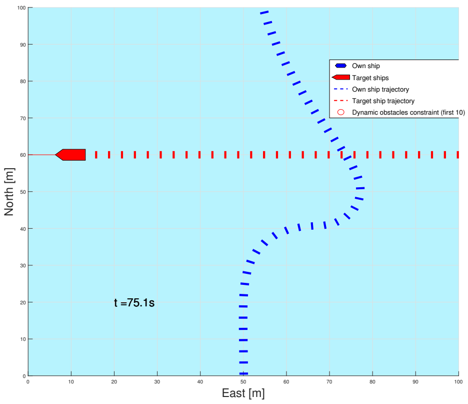
(f) mhm



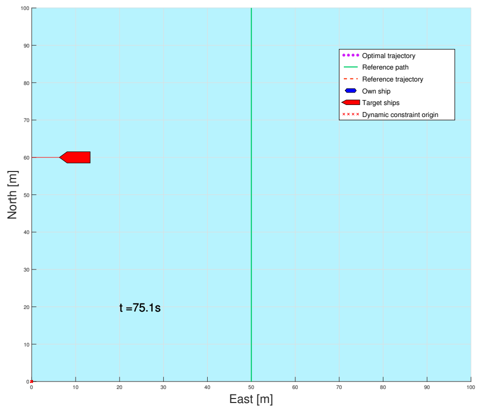
(g) caption



(h) mhm



(i) caption



(j) mhm

Figure 12: Simple Give Way With Prediction

- 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

  for other navigator.

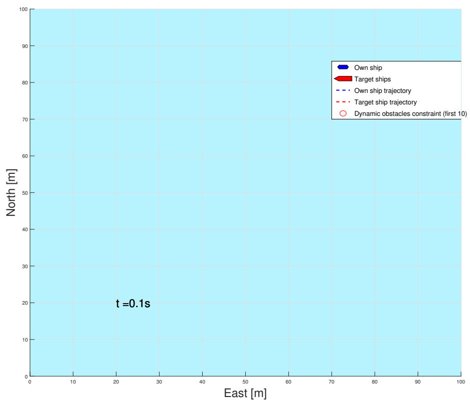
- The author realized late that the way prediction is done of the TS is not entirely consistent with the way prediction is handled internally in the algorithm. In the algorithm waypoints are used to predict TSs trajectories, which is not neccessary

  the exact same trajectory as the TS ends up following.

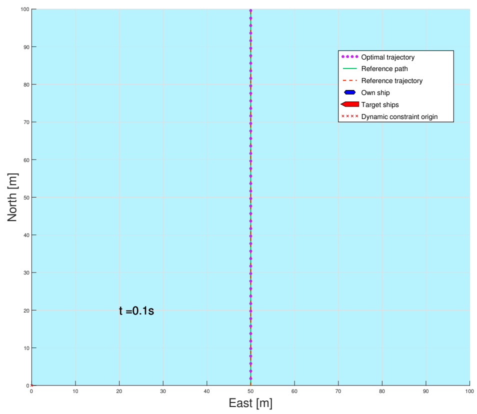
- In order to get the TS to comply with expected COLREGs behaviour a waypoint was placed some meters south of the OS position at the would be tCPA. This waypoint would obviously not exist in a normal transit situation and so

  this scenario is actually a bit of a cheat in regards to how prediction is argued for in prior chapters.

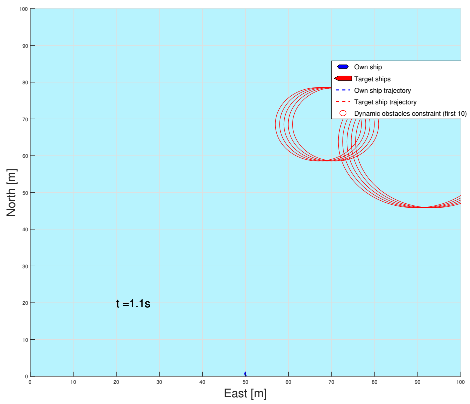
- The result is still interesting, for this scenario only we can exchange the advanced prediction result with a fully accurate prediction, and the result for simple prediction would hold for the current algorithm in all cases.



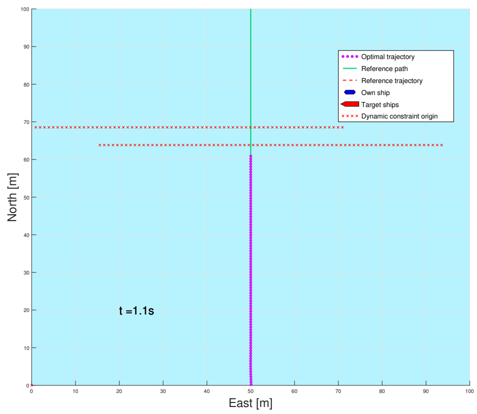
(a) caption



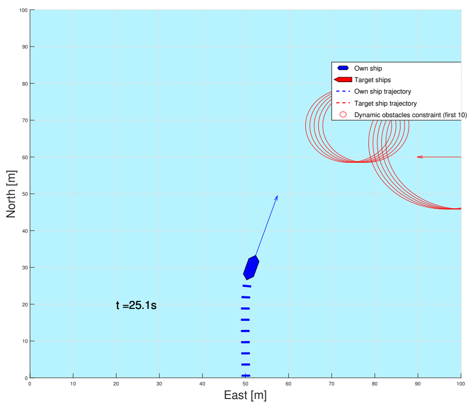
(b) mhm



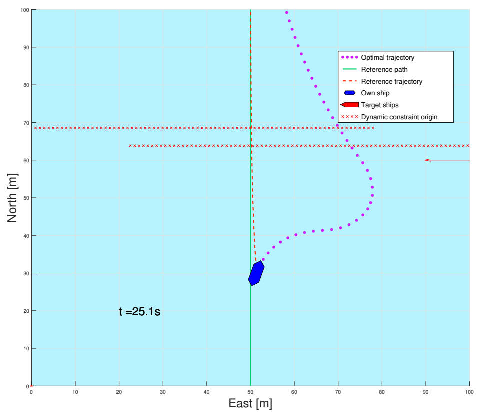
(c) caption



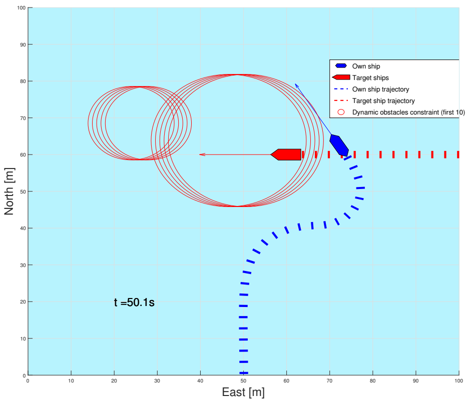
(d) mhm



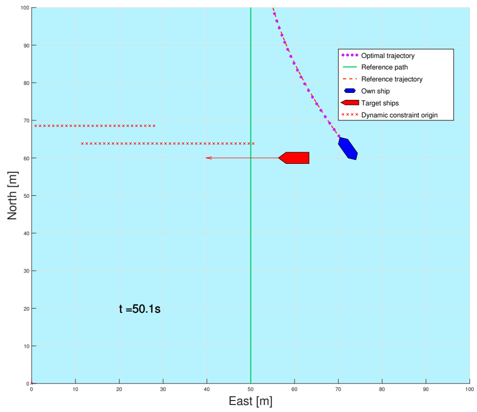
(e) caption



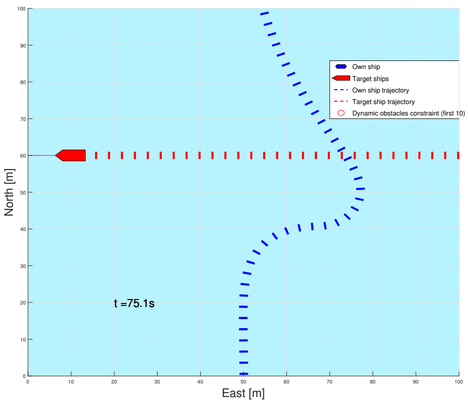
(f) mhm



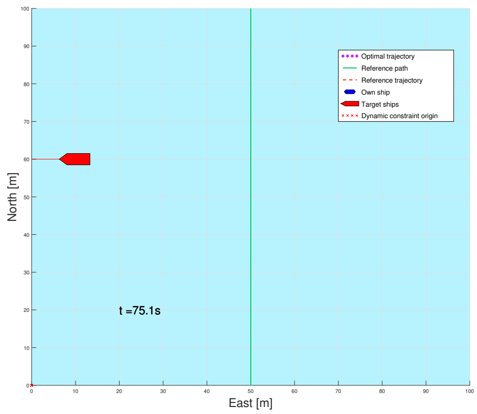
(g) caption



(h) mhm



(i) caption

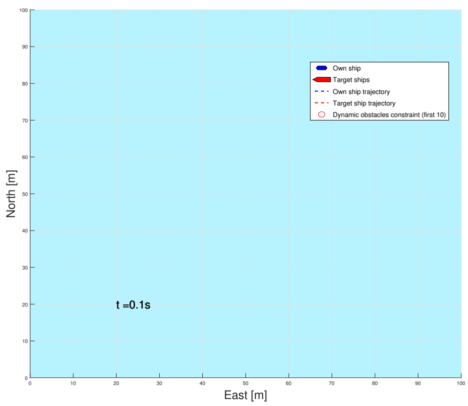


(j) mhm

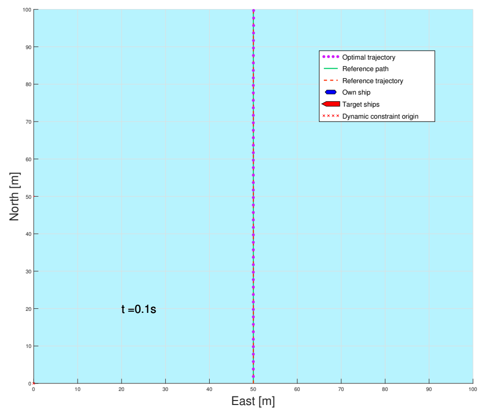
Figure 13: Simple Give Way Without Prediction

4.2.4 Turn Head On

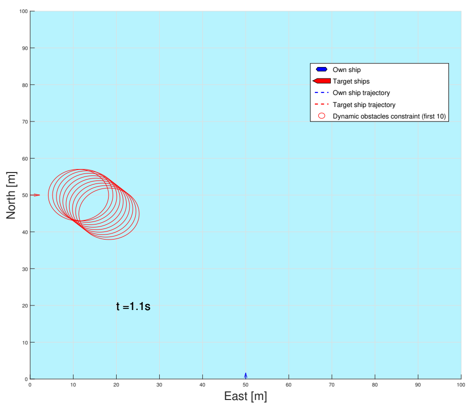
- THIS SCENARIO SHOULD BE MIRRORED HORIZONTALLY TO INCREASE THE CHANCES OF AN OBSERVABLE DIFFERENCE BETWEEN PREDICTIONS.
- otherwise business as usual, turns aren't predicted quite right.



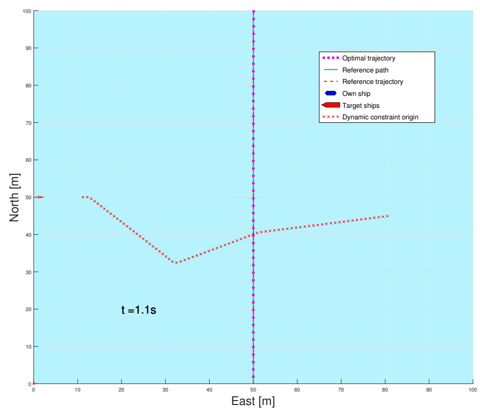
(a) caption



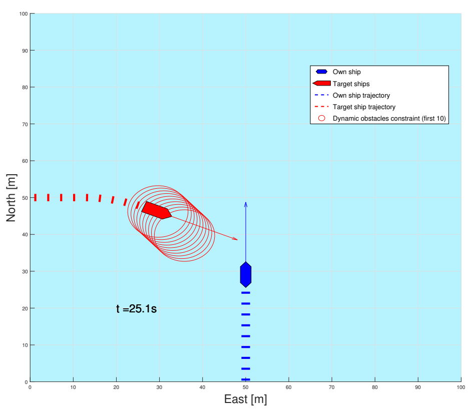
(b) mhm



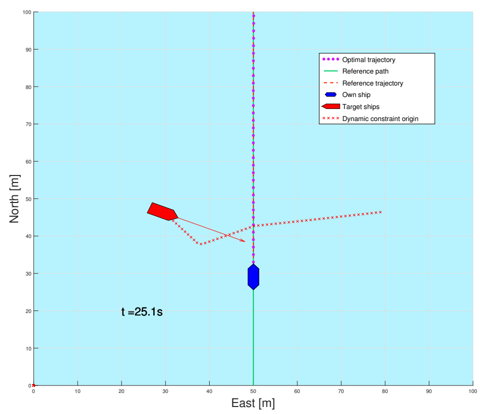
(c) caption



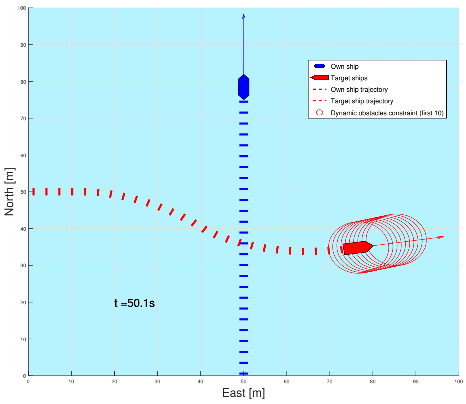
(d) mhm



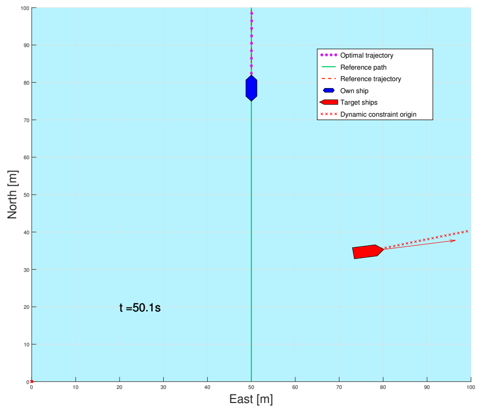
(e) caption



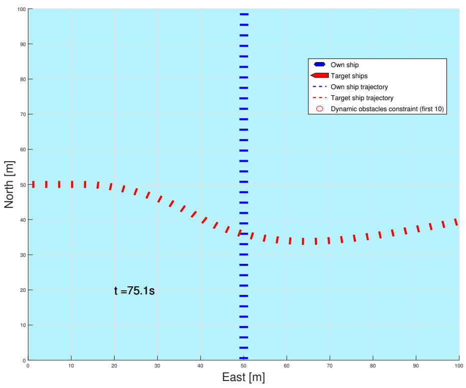
(f) mhm



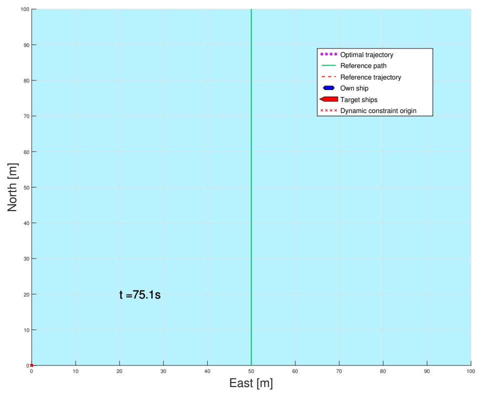
(g) caption



(h) mhm

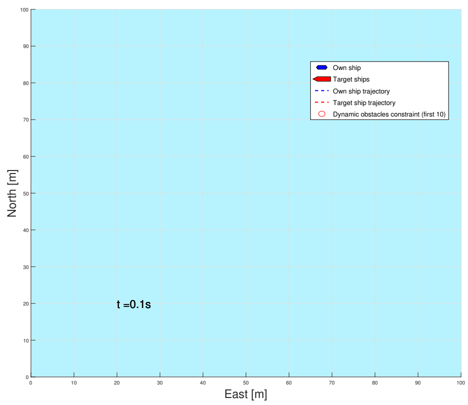


(i) caption

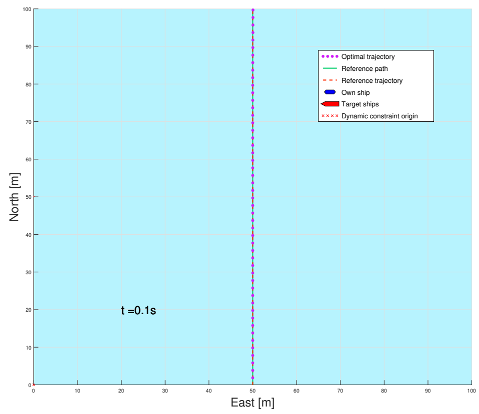


(j) mhm

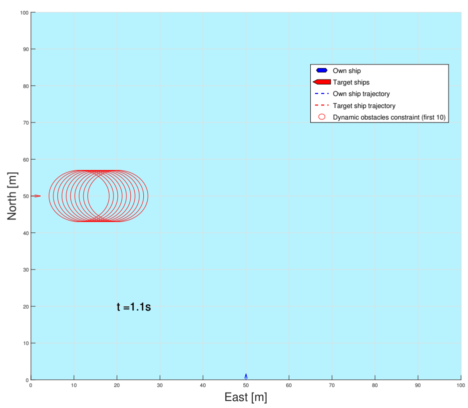
Figure 14: Simple Stand On With Prediction



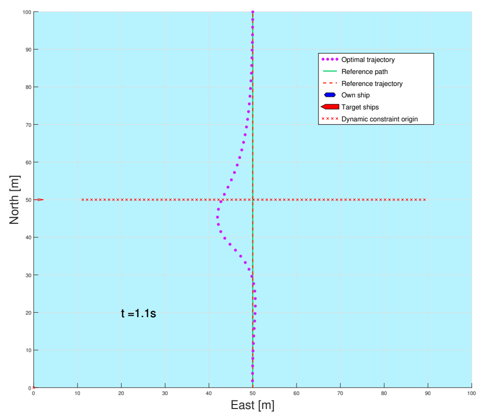
(a) caption



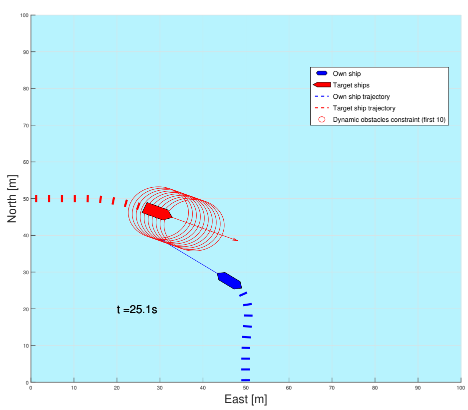
(b) mhm



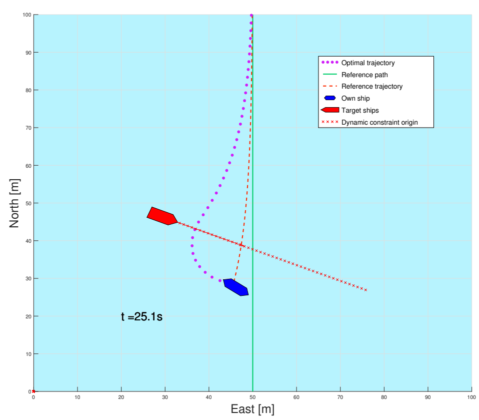
(c) caption



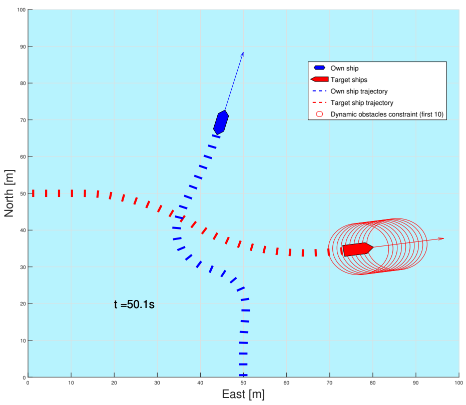
(d) mhm



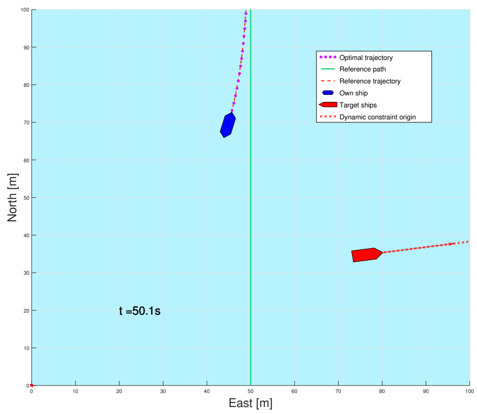
(e) caption



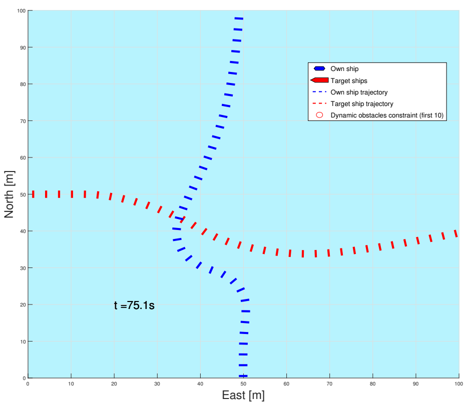
(f) mhm



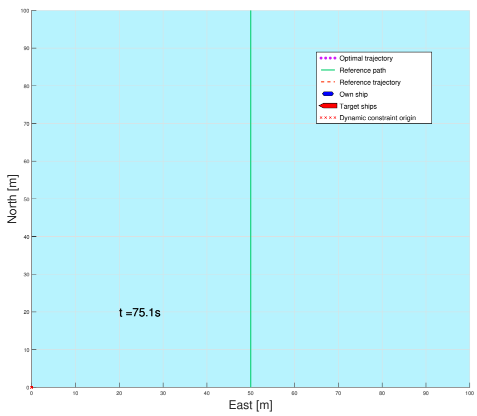
(g) caption



(h) mhm



(i) caption

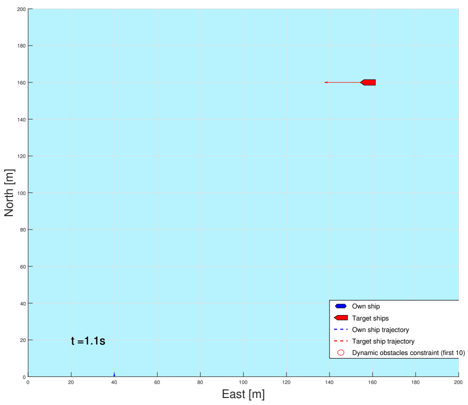


(j) mhm

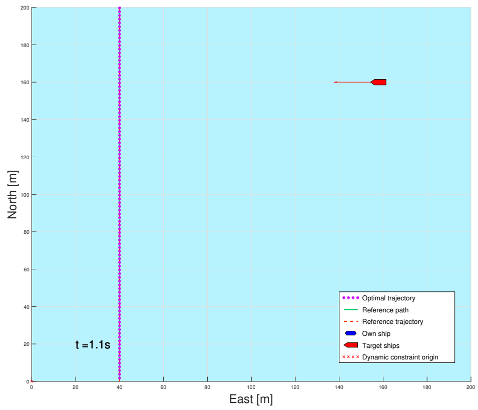
Figure 15: Simple Stand On Without Prediction

4.2.5 Turn Give Way

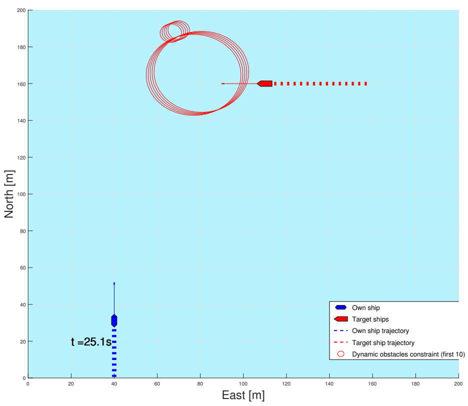
- Here we finally observe a big differene between full and simple prediction.
- due to how the situation plays out the OS is dragged along by the constraints of the TS.
- being dragged by constraints is not unique to this situation, and is one of the problems that



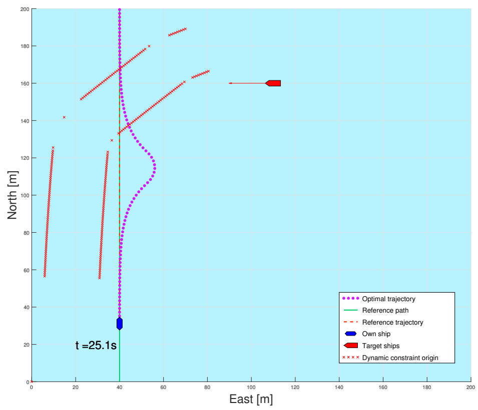
(a) caption



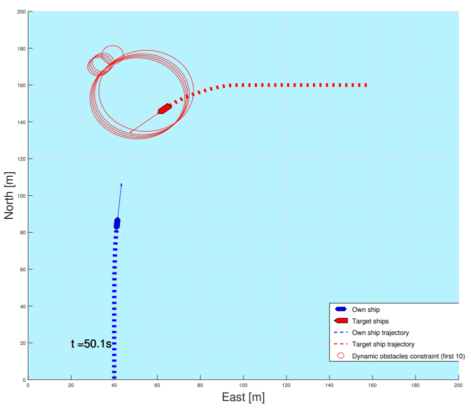
(b) mhm



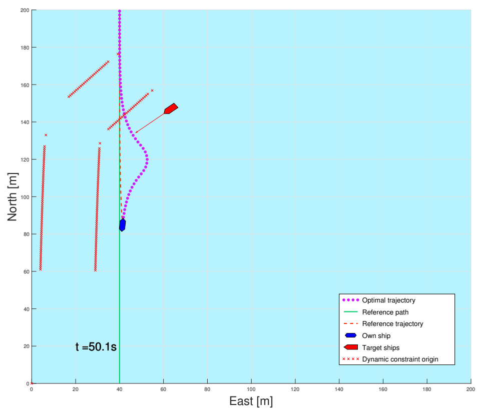
(c) caption



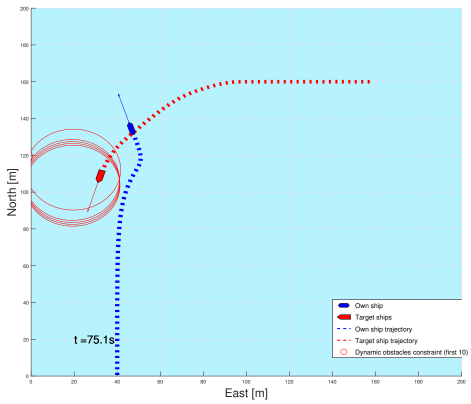
(d) mhm



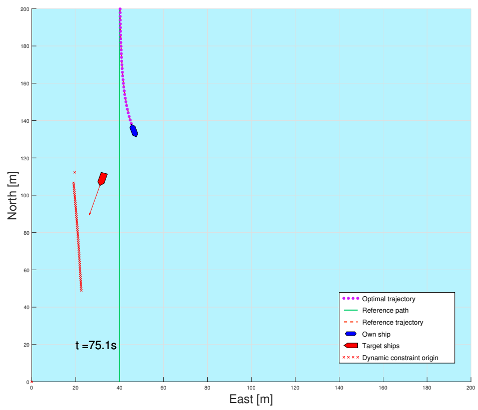
(e) caption



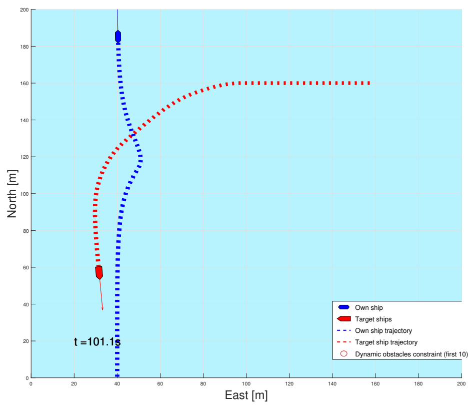
(f) mhm



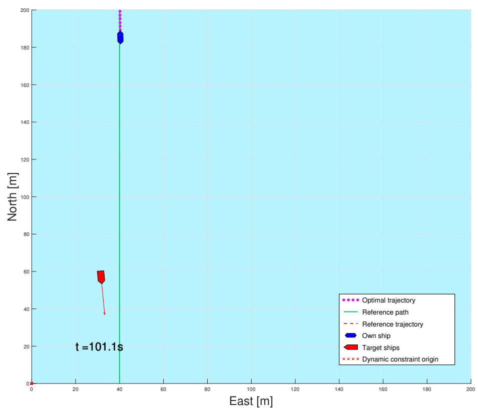
(g) caption



(h) mhm



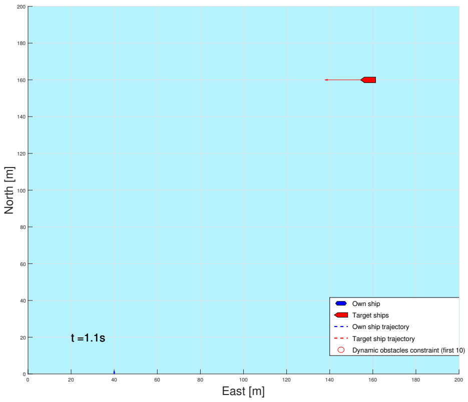
(i) caption



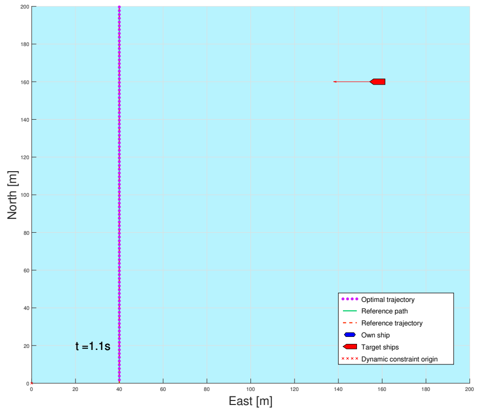
(j) mhm

Figure 16: Turn Head On With Prediction

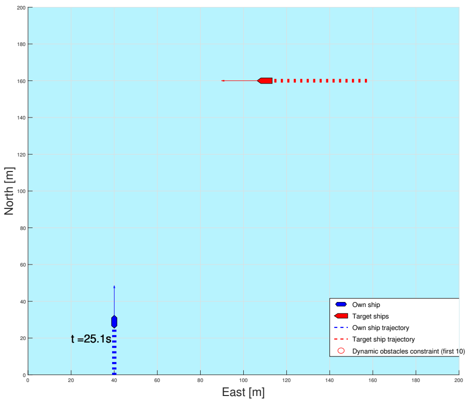
can occur with this method of collision avoidance.



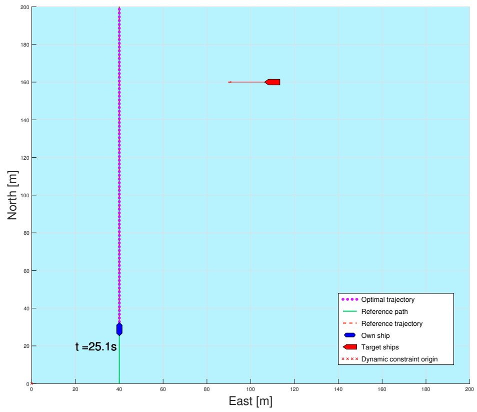
(a) caption



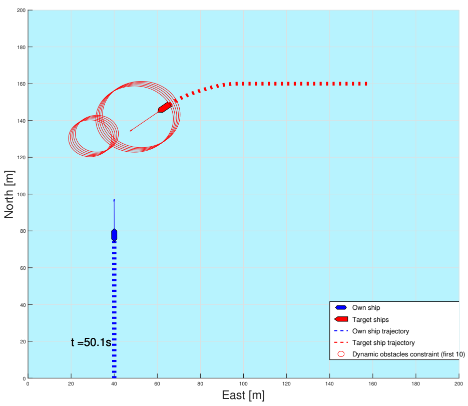
(b) mhm



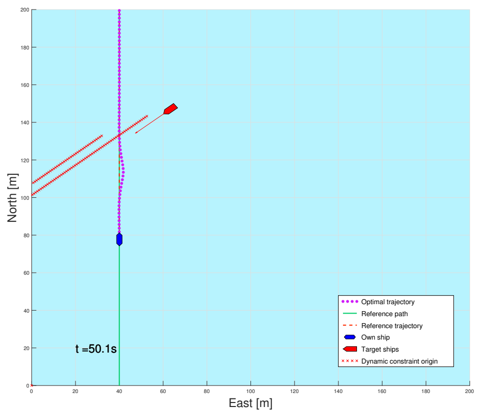
(c) caption



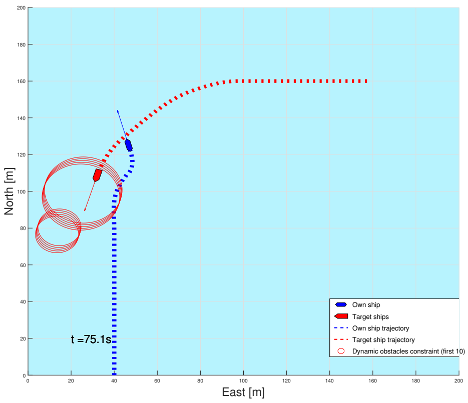
(d) mhm



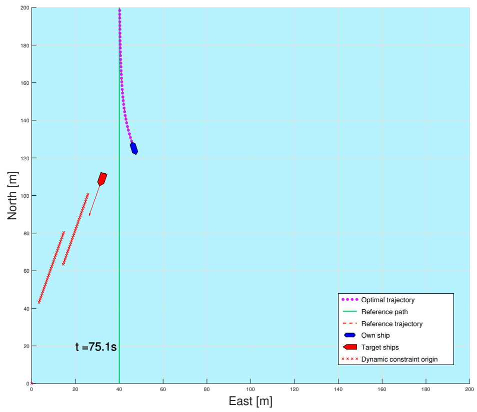
(e) caption



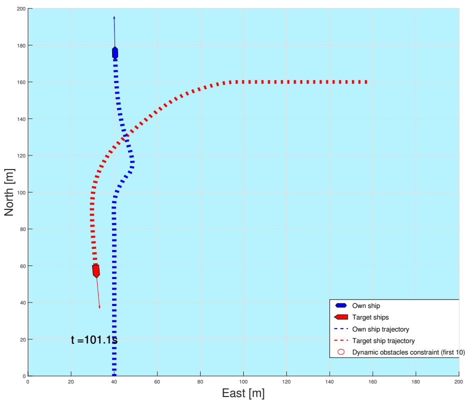
(f) mhm



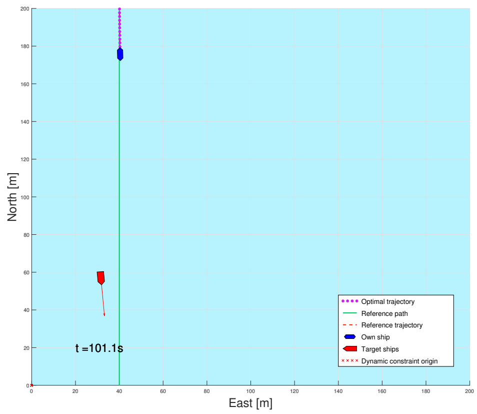
(g) caption



(h) mhm



(i) caption

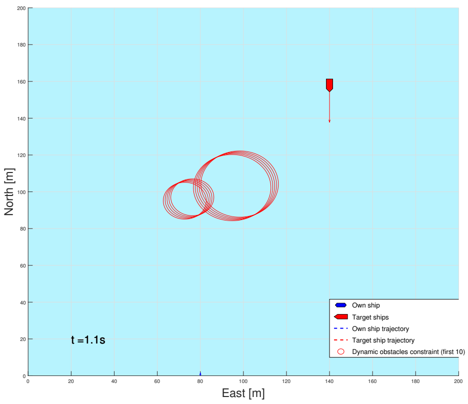


(j) mhm

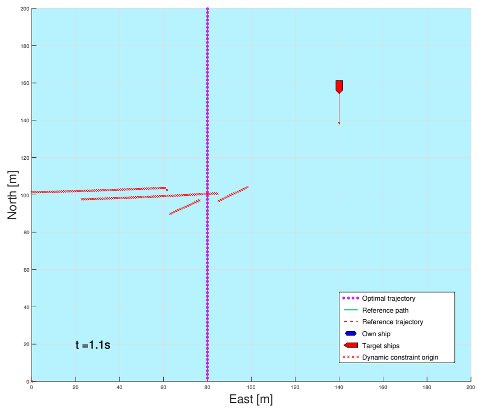
Figure 17: Turn Head On WithOUT Prediction

4.2.6 Turn Stand On

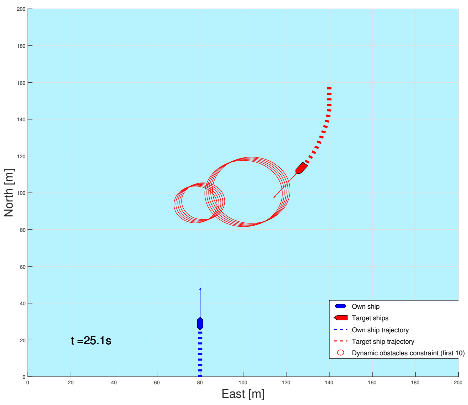
- Absolutely nothing happens here. move on.



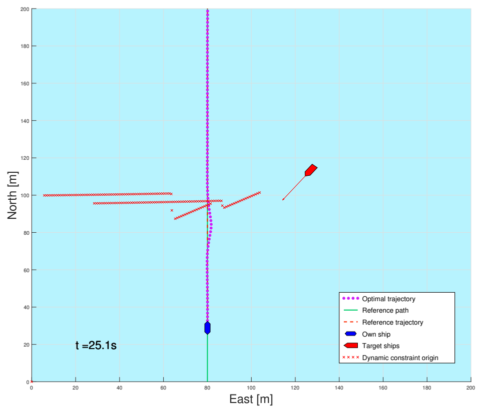
(a) caption



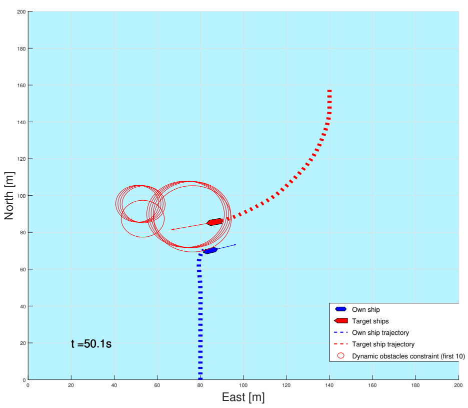
(b) mhm



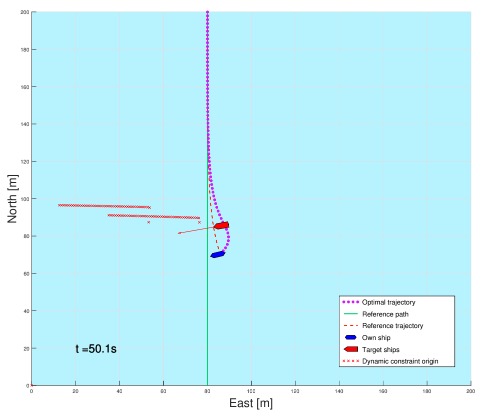
(c) caption



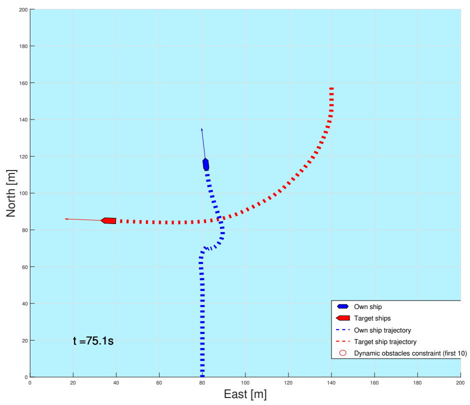
(d) mhm



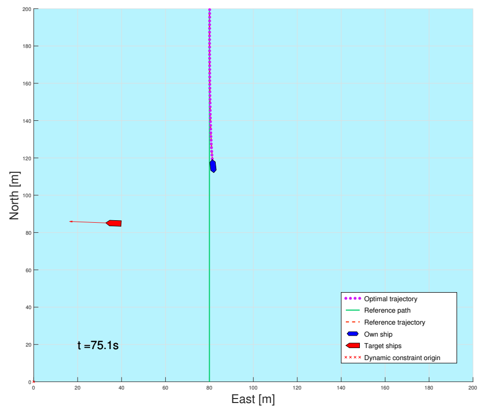
(e) caption



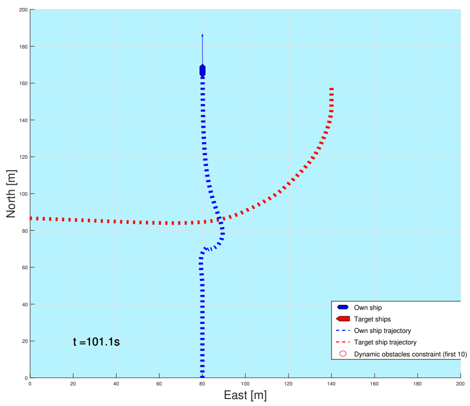
(f) mhm



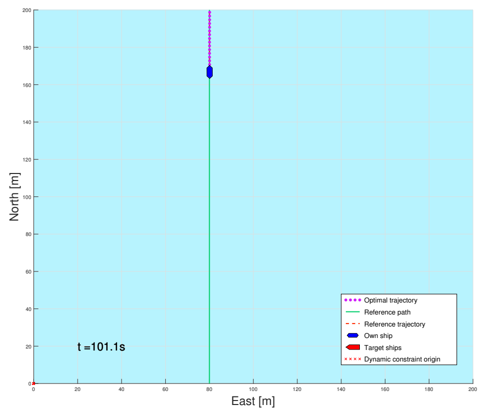
(g) caption



(h) mhm

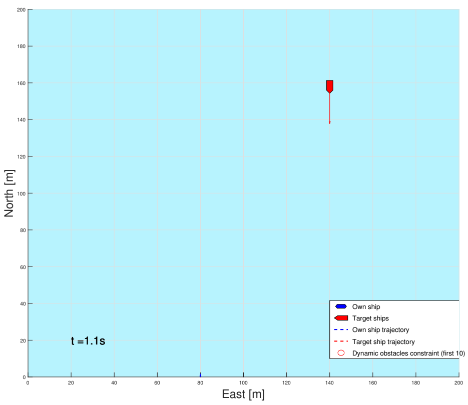


(i) caption

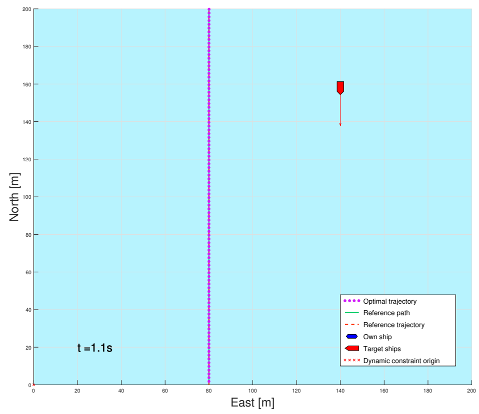


(j) mhm

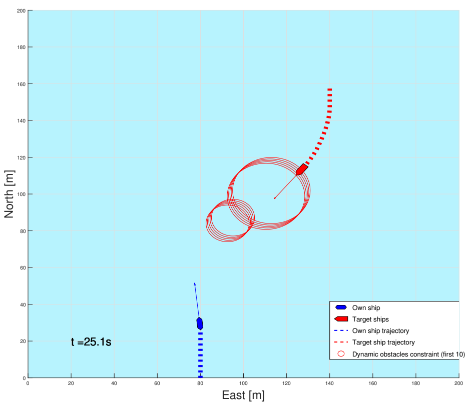
Figure 18: Turn GIVE WAY With Prediction



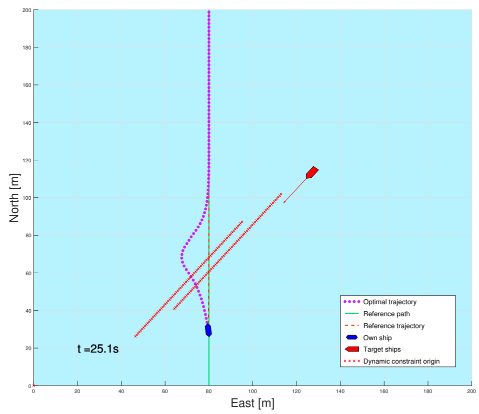
(a) caption



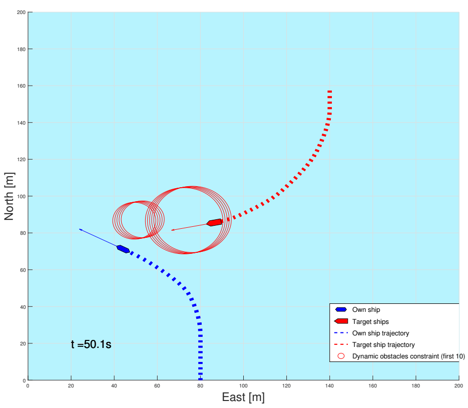
(b) mhm



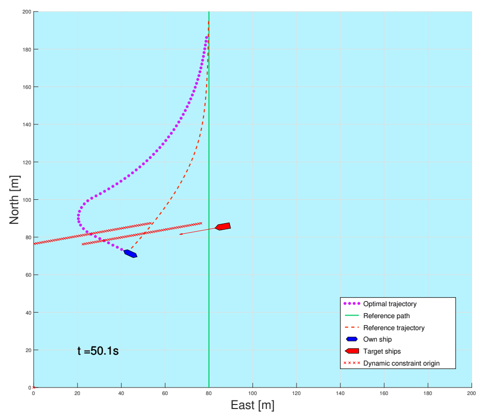
(c) caption



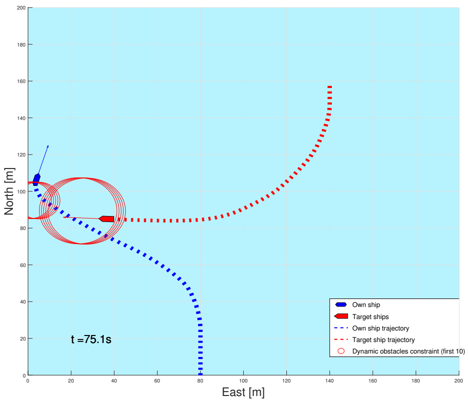
(d) mhm



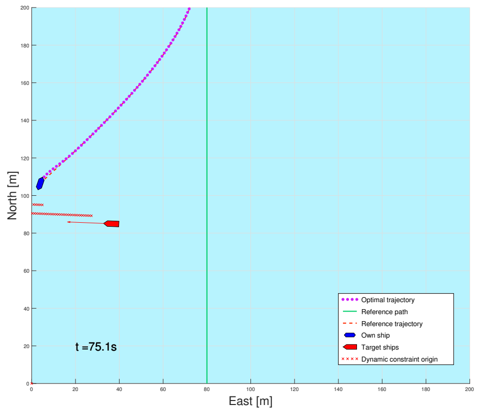
(e) caption



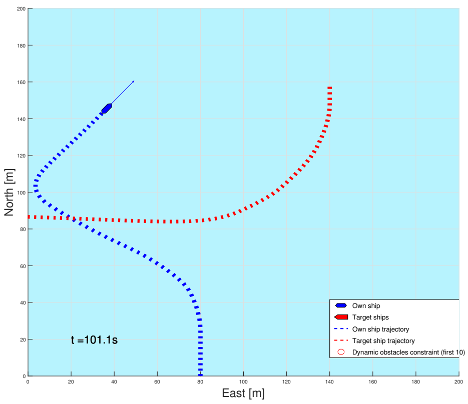
(f) mhm



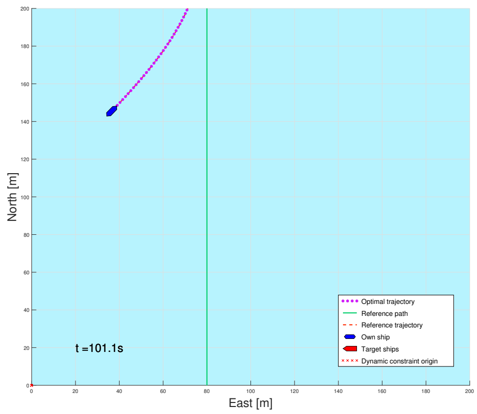
(g) caption



(h) mhm



(i) caption

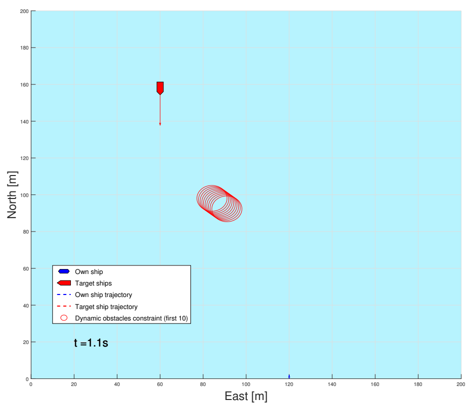


(j) mhm

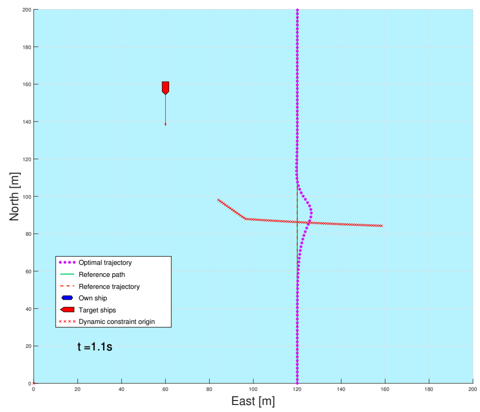
Figure 19: Turn GIVE WAY WithOUT Prediction

4.2.7 Canals

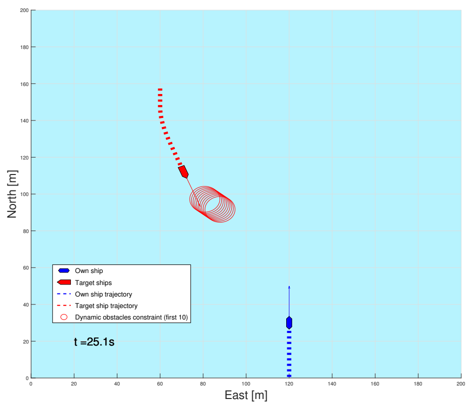
- blocked path
- feasibility check
- forskjell mellom prediction metoder



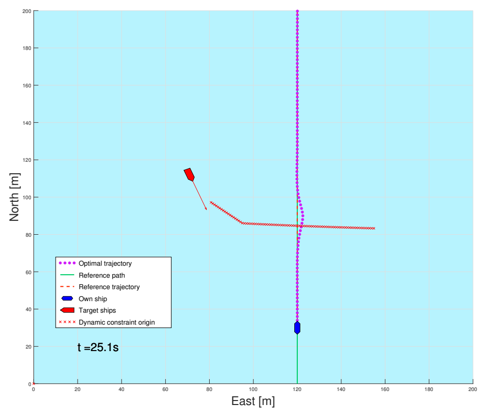
(a) caption



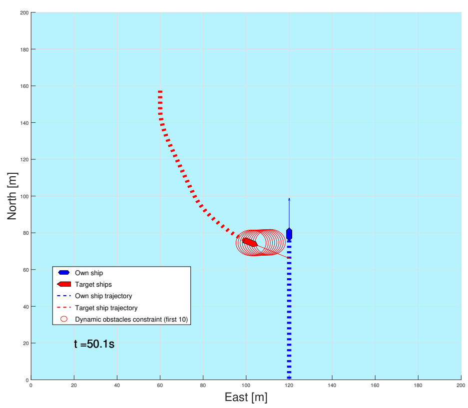
(b) mhm



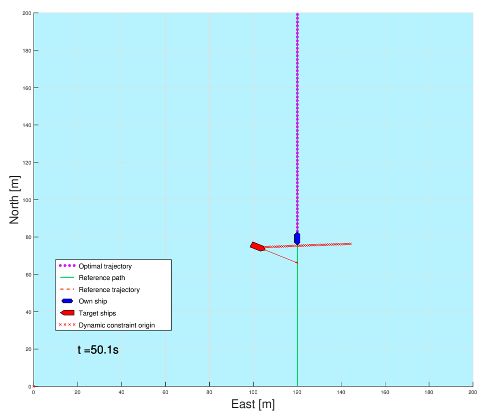
(c) caption



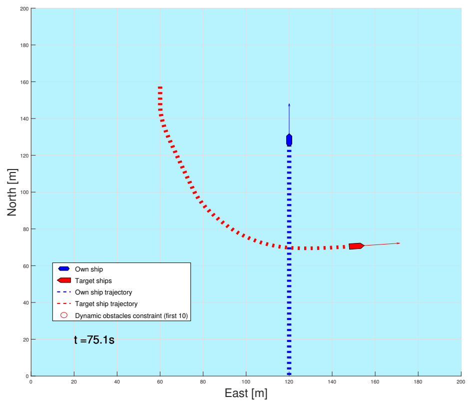
(d) mhm



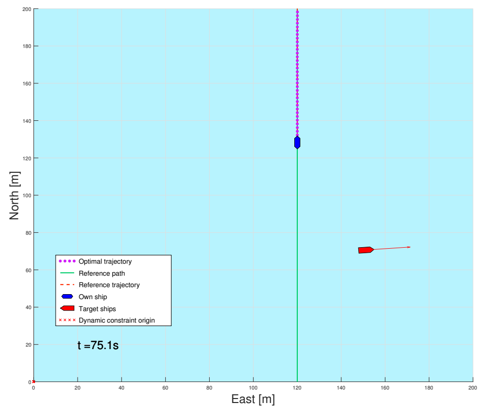
(e) caption



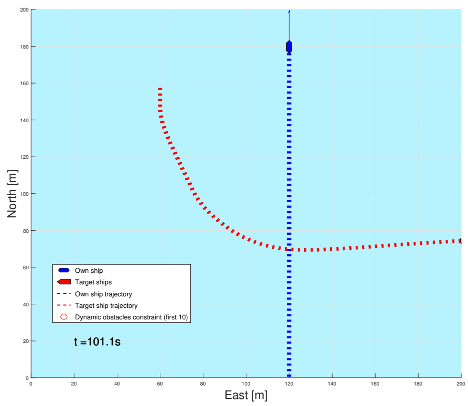
(f) mhm



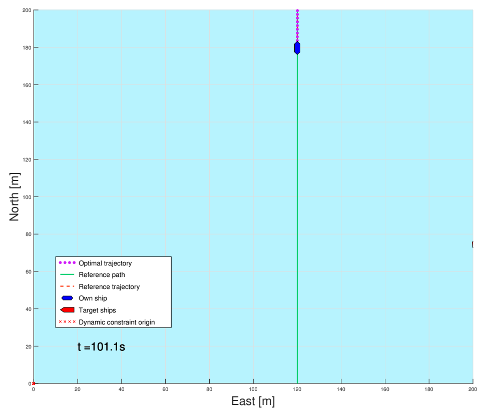
(g) caption



(h) mhm



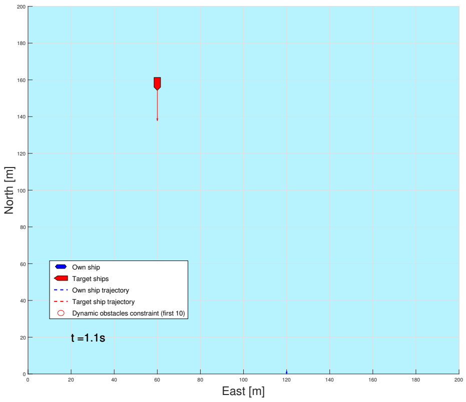
(i) caption



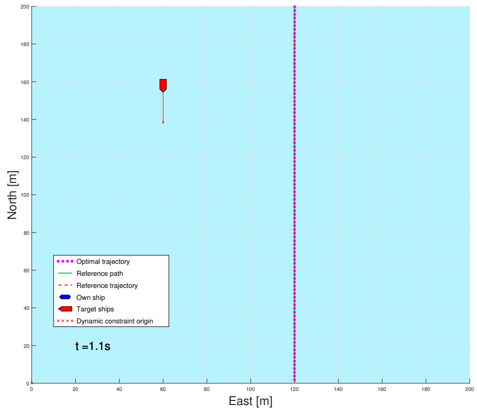
(j) mhm

Figure 20: Turn Stand On With Prediction

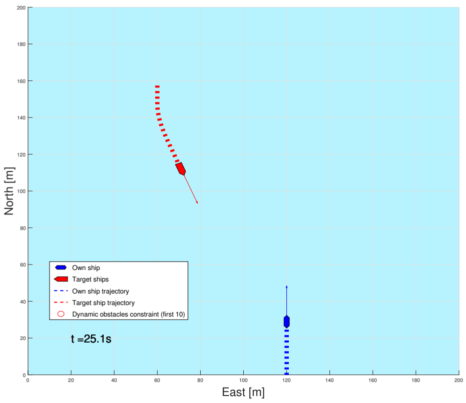
- When the path is blocked it takes a very very very long time to solve the NLP, users beware.
- Get to see static obstacles in effect.



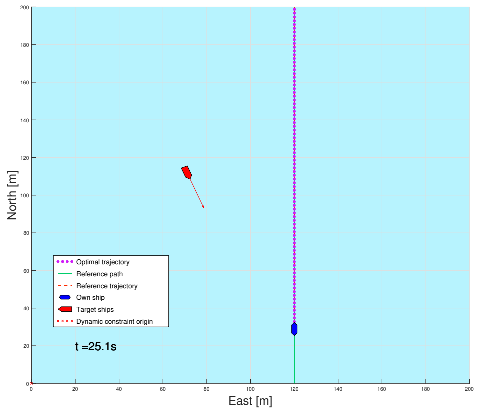
(a) caption



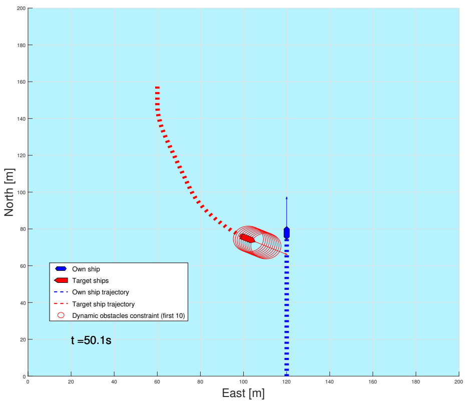
(b) mhm



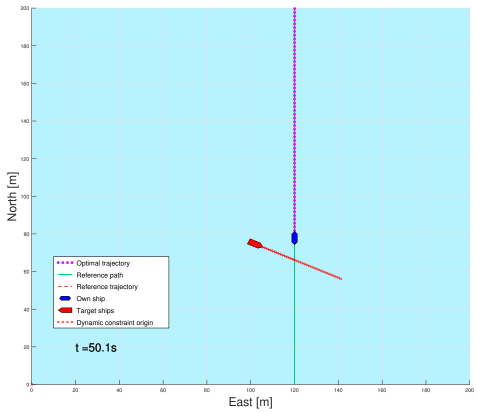
(c) caption



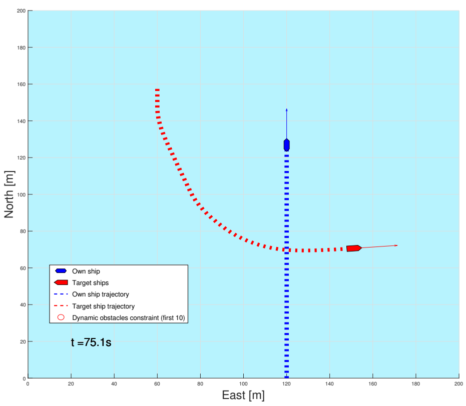
(d) mhm



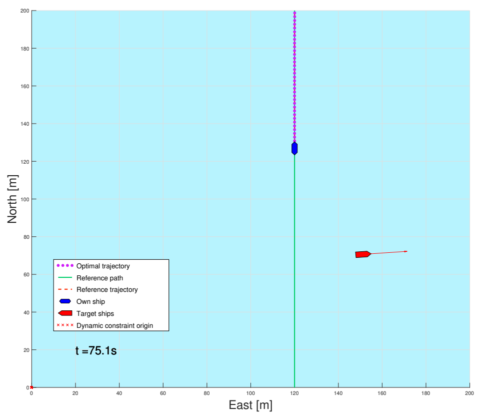
(e) caption



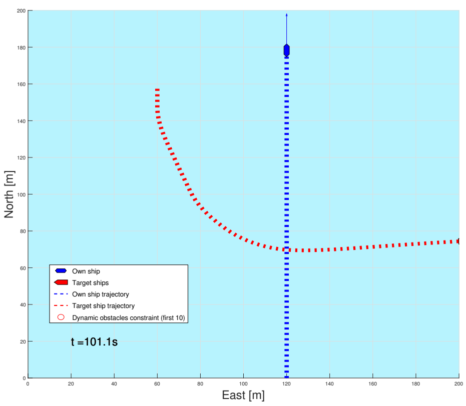
(f) mhm



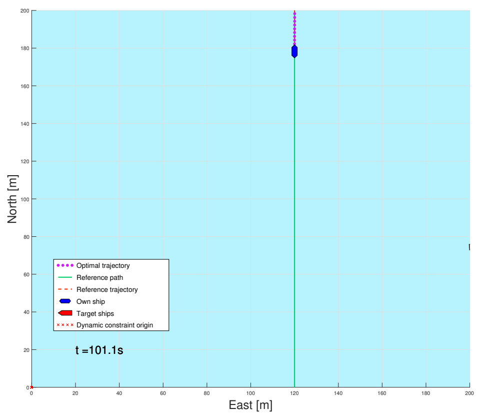
(g) caption



(h) mhm

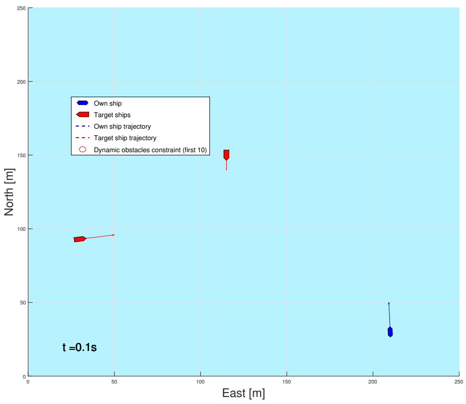


(i) caption

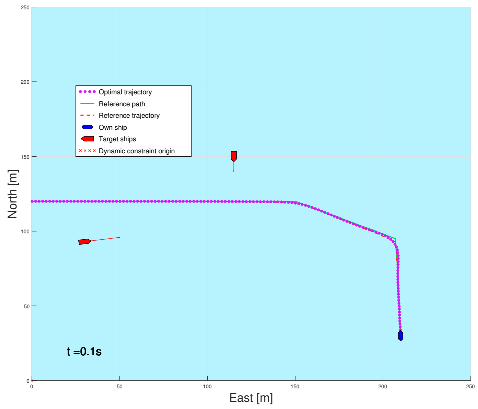


(j) mhm

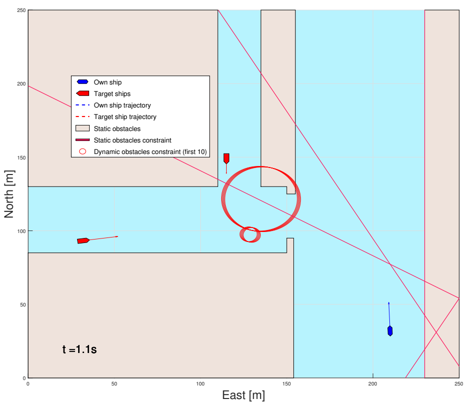
Figure 21: Turn Stand On WithOUT Prediction



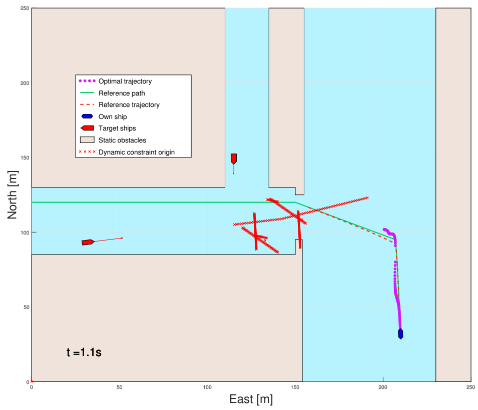
(a) caption



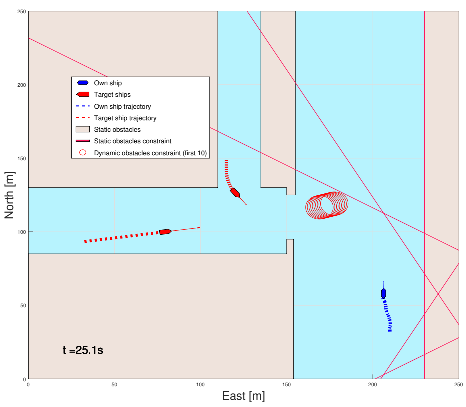
(b) mhm



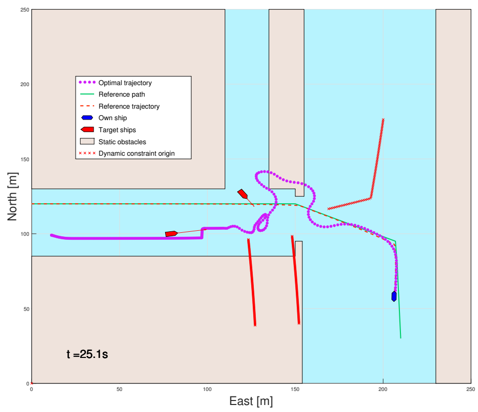
(c) caption



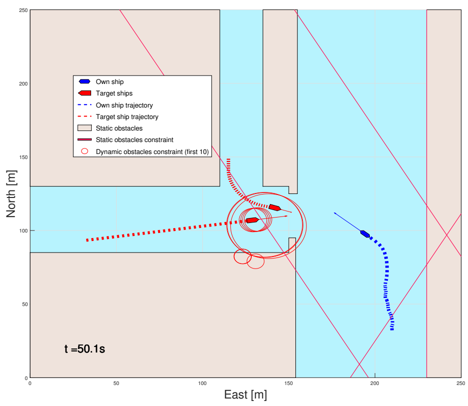
(d) mhm



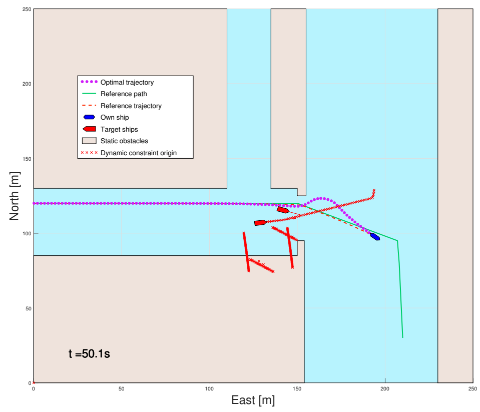
(e) caption



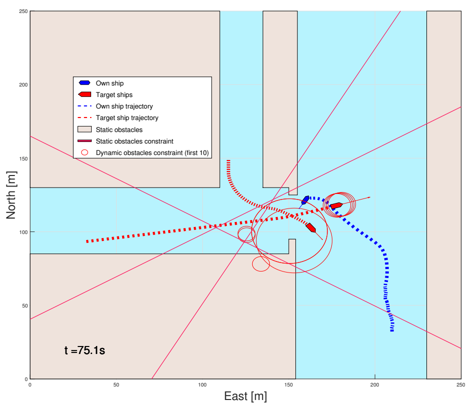
(f) mhm



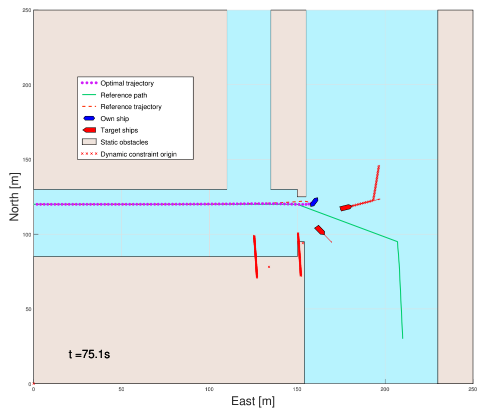
(g) caption



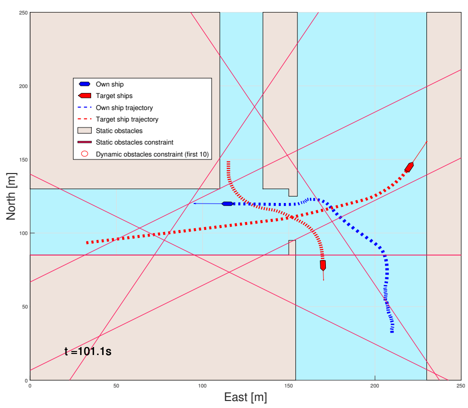
(h) mhm



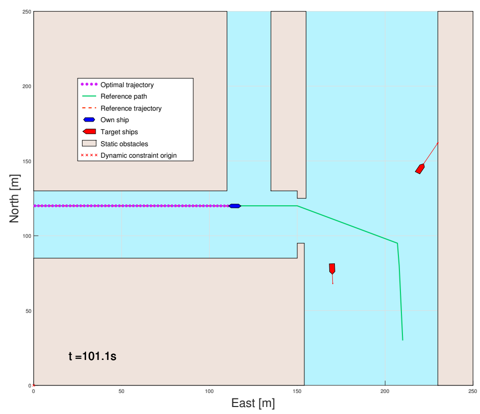
(i) caption



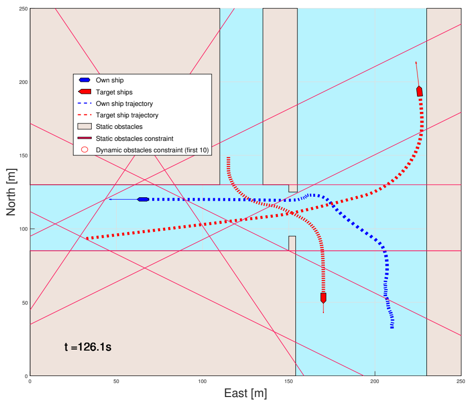
(j) mhm



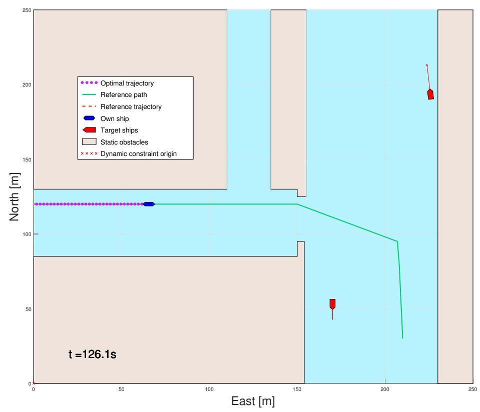
(k) caption



(l) mhm



(m) caption

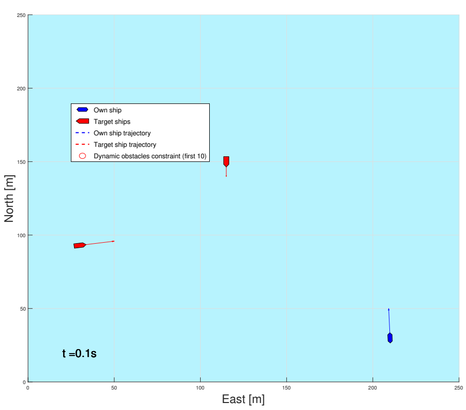


(n) mhm

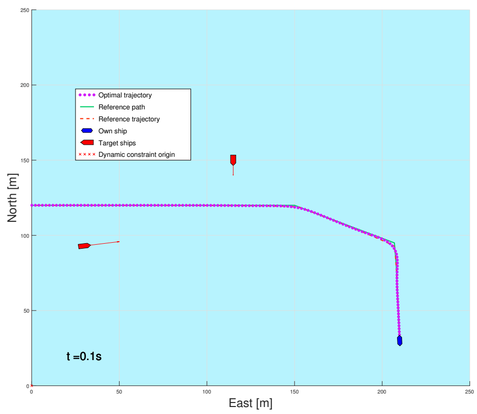
Figure 22: Canals simulation. On the left shown with current active dynamic and static constraints. On the right seen with projected future trajectory

4.2.8 Trondheimsfjord

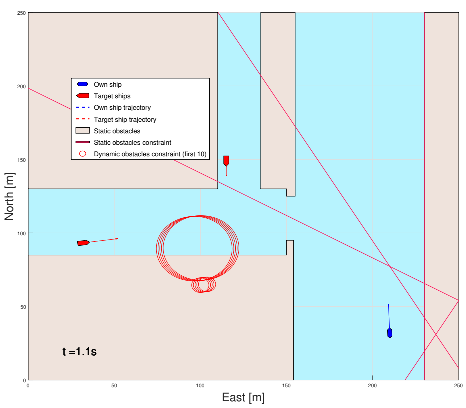
- COLREGs stress test
- Full prediction behaves much 'calmer', which is better in the author's opinion.
- Full prediction is also much more computationally efficient, for reasons that will be discussed



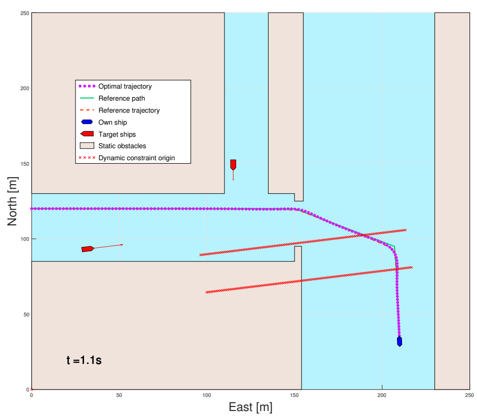
(a) caption



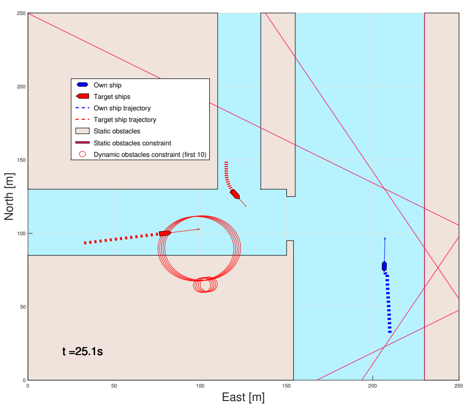
(b) mhm



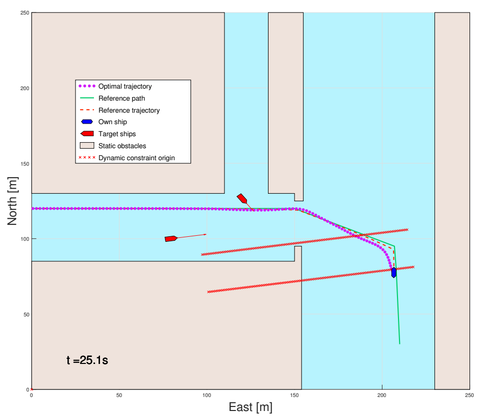
(c) caption



(d) mhm

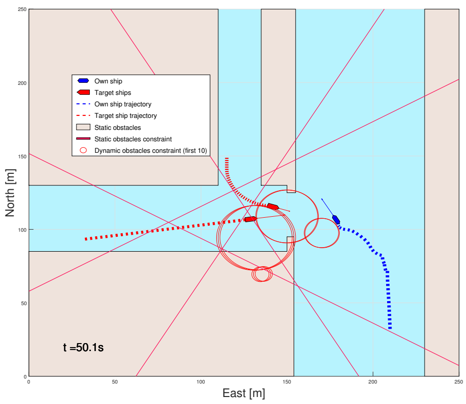


(e) caption

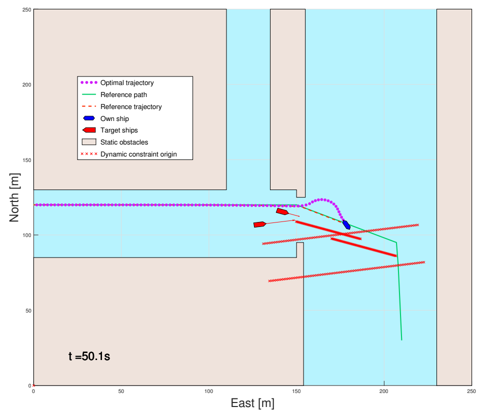


(f) mhm

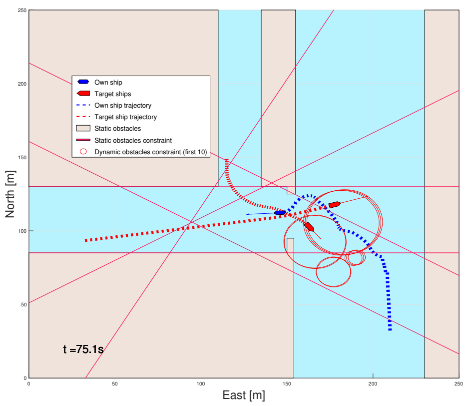
later.



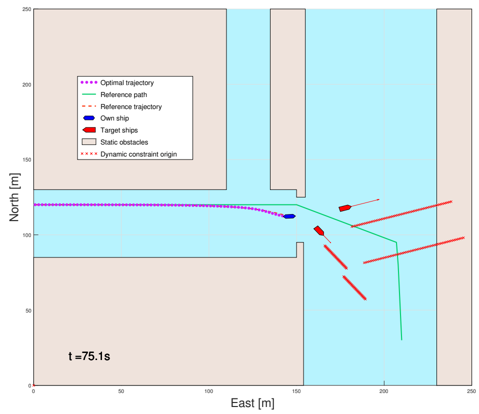
(g) dsadada



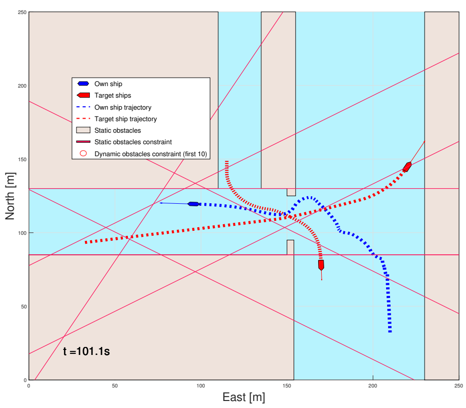
(h) mhm



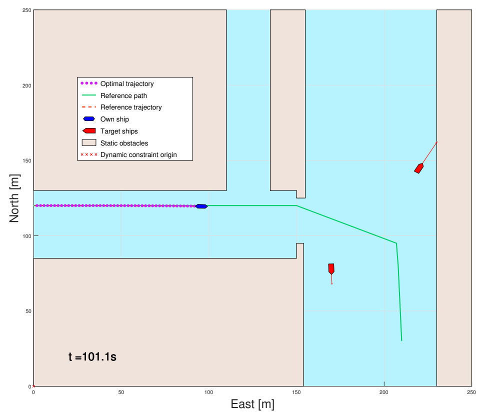
(i) caption



(j) mhm

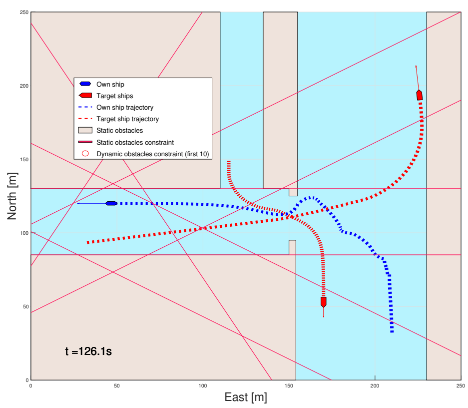


(k) caption

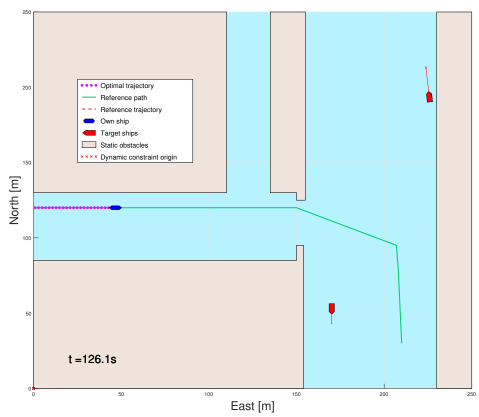


(l) mhm

Figure 23: Canals simulation without prediction, shown with and without constraints

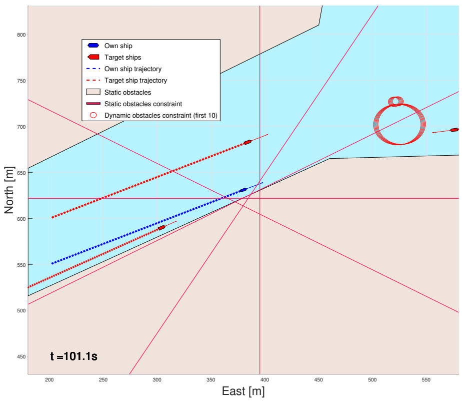


(m) caption

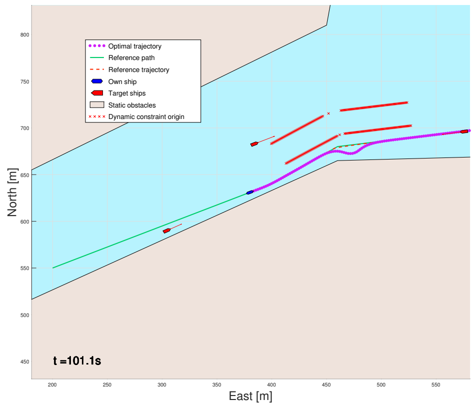


(n) mhm

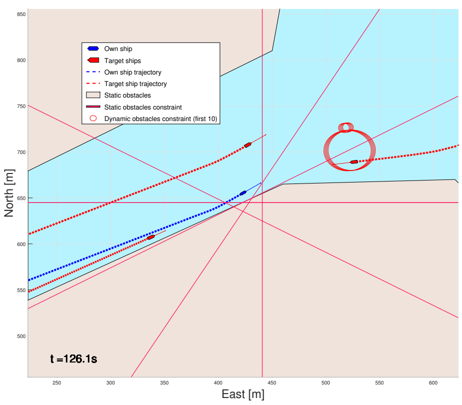
Figure 23: TODO: skriv. Havn1 w_opt without prediction



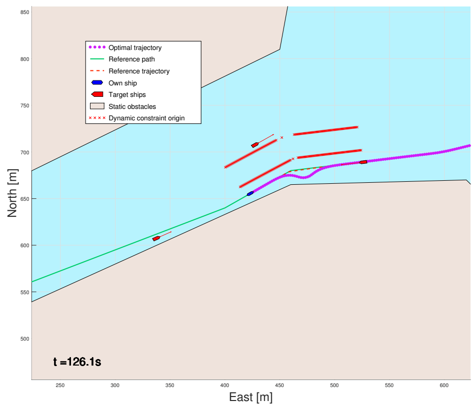
(a) caption



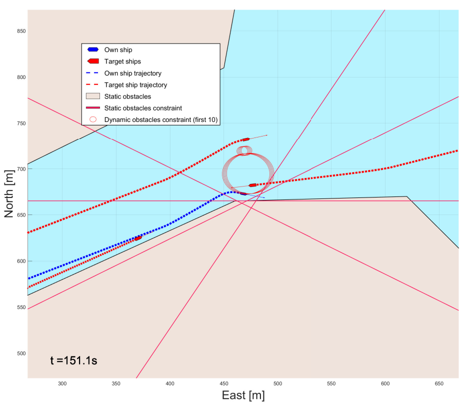
(b) mhm



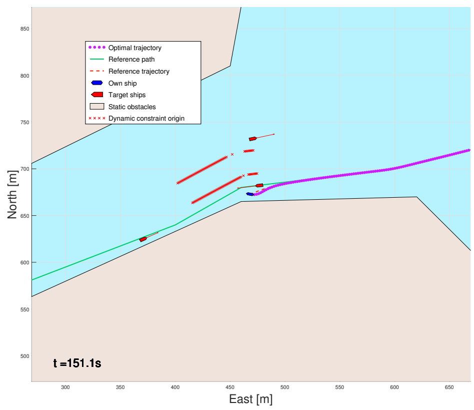
(c) caption



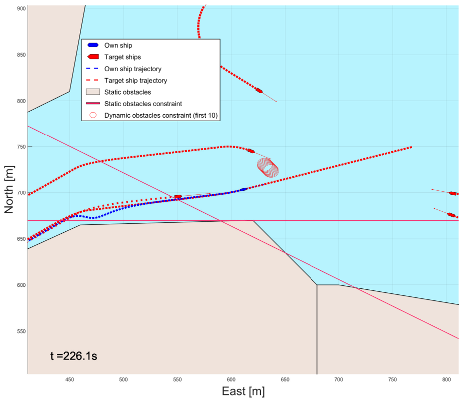
(d) mhm



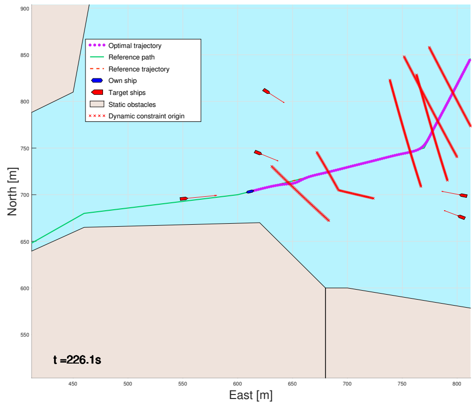
(e) caption



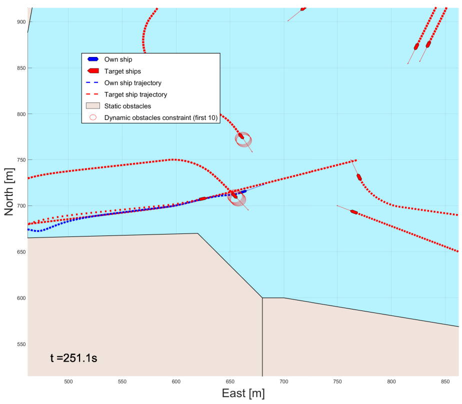
(f) mhm



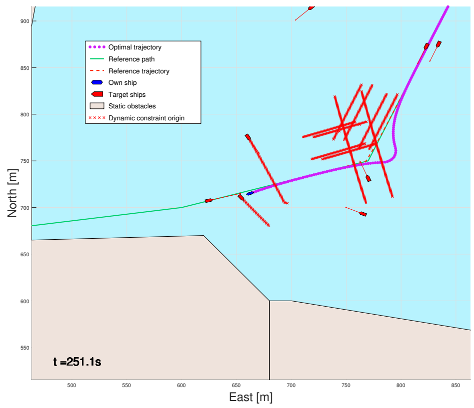
(g) caption



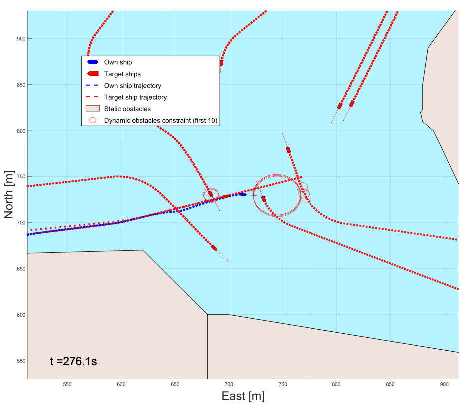
(h) mhm



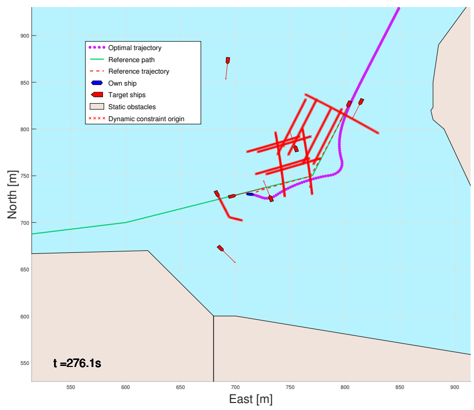
(i) caption



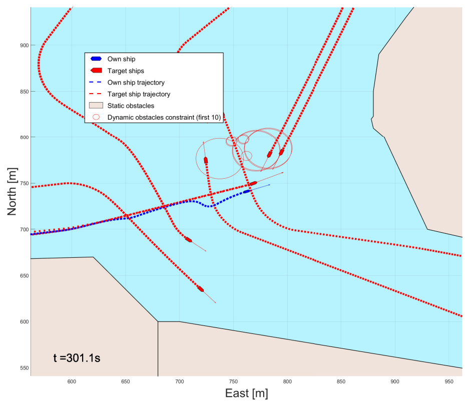
(j) mhm



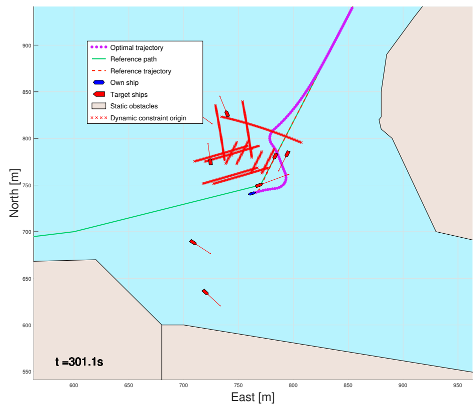
(k) caption



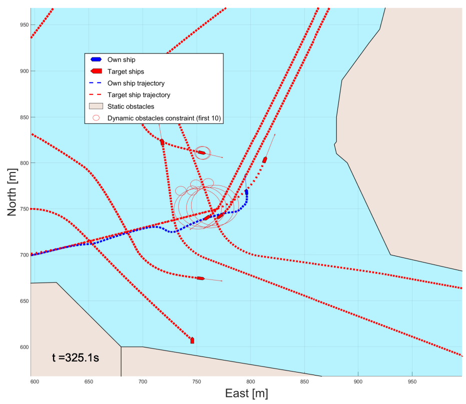
(l) mhm



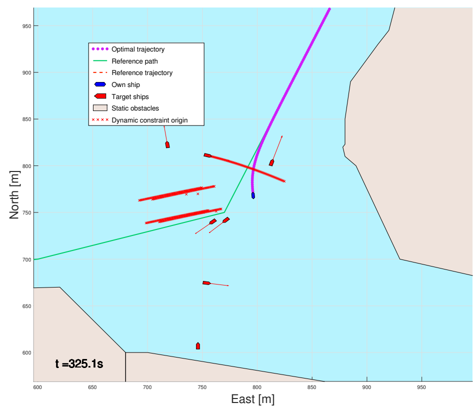
(m) caption



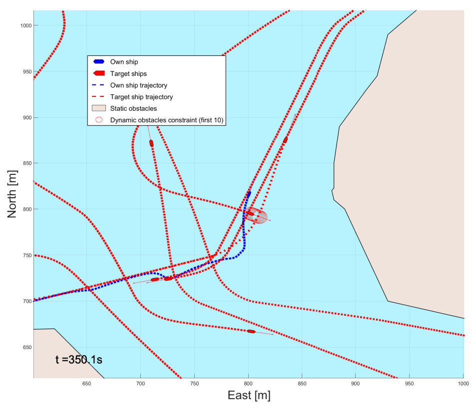
(n) mhm



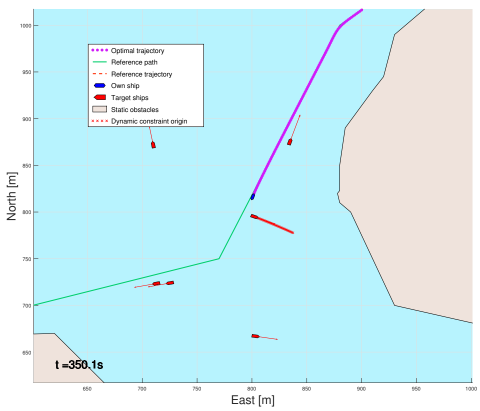
(o) caption



(p) mhm



(q) caption

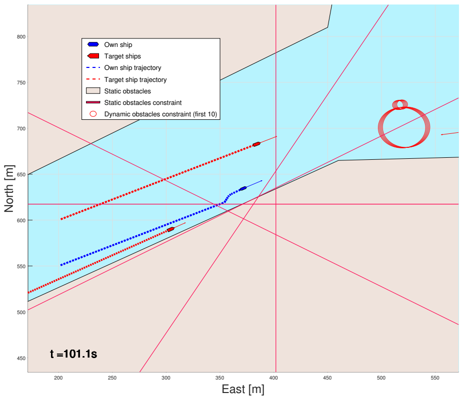


(r) mhm

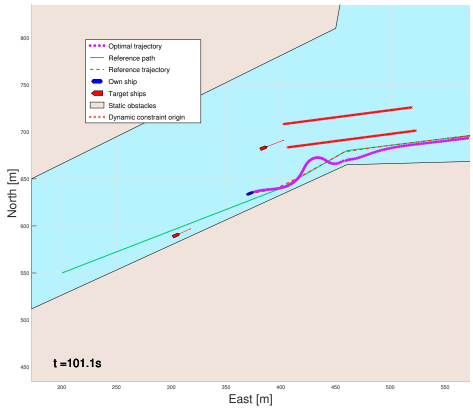
Figure 24: Trondheimfjord simulation. On the left shown with current active dynamic and static constraints. On the right seen with projected future trajectory

4.2.9 Helløya

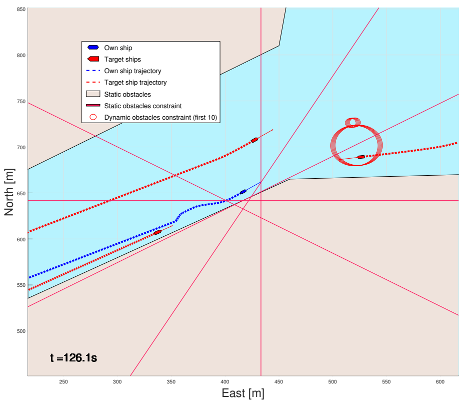
- Both simple and full prediction actually navigate the 'invisible' turn quite well
- even in the reverse direction it's not really a problem, though simple prediction cuts the turn, which could be considered bad.



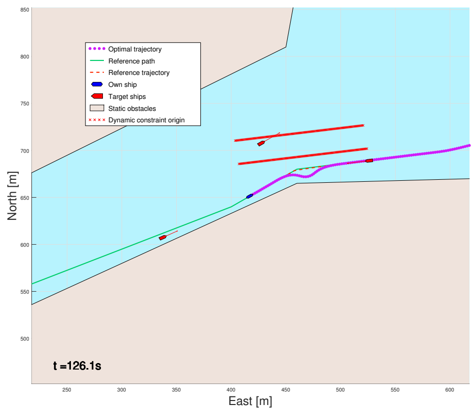
(a) caption



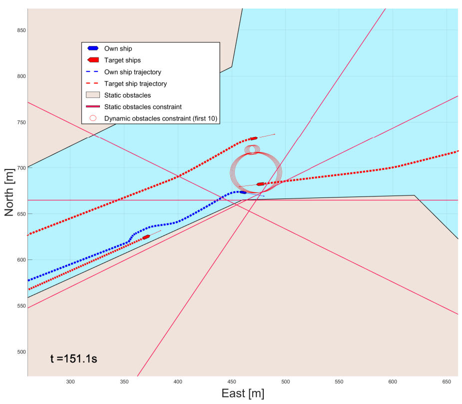
(b) mhm



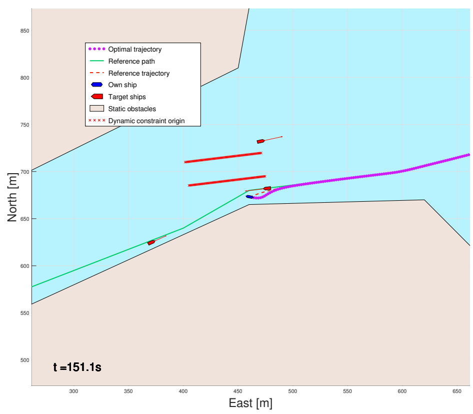
(c) caption



(d) mhm



(e) caption

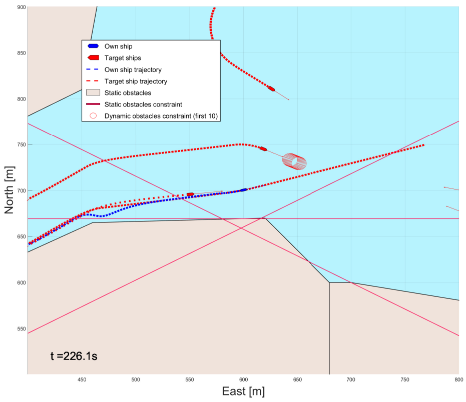


(f) mhm

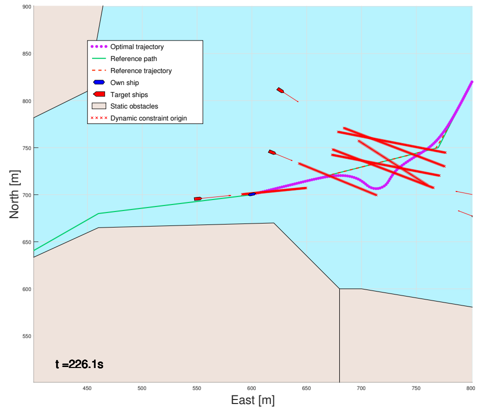
- when being overtaken the full prediction exhibits much better behaviour.

- This is one of the scenarios where the scale is very exaggerated.

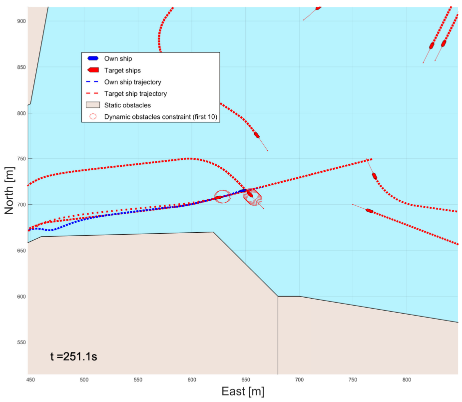
(TODO: Skriv og figurer)



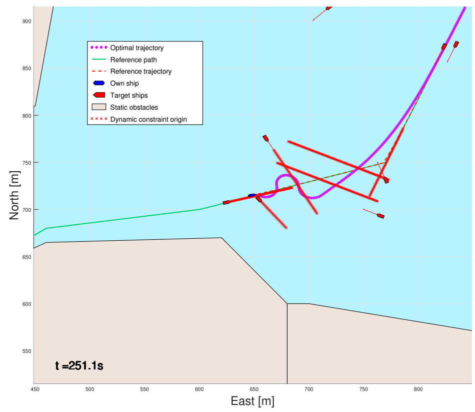
(g) caption



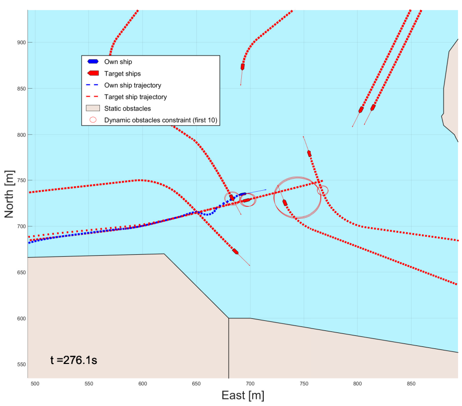
(h) mhm



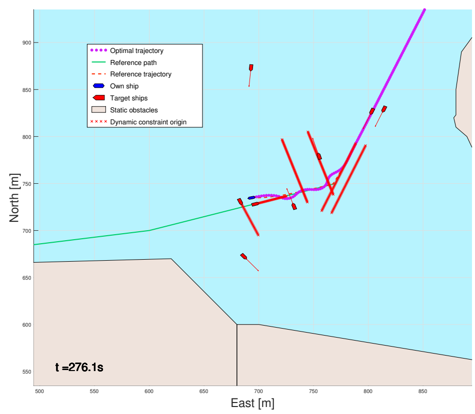
(i) caption



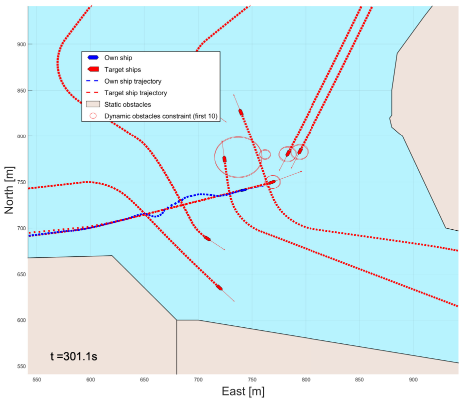
(j) mhm



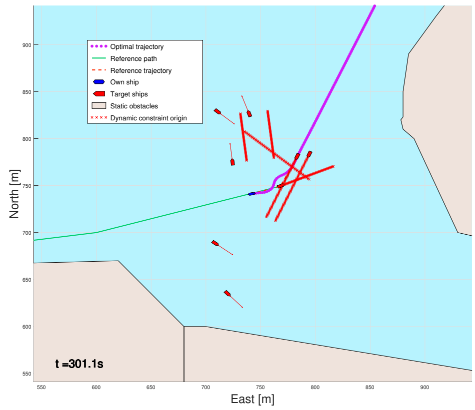
(k) caption



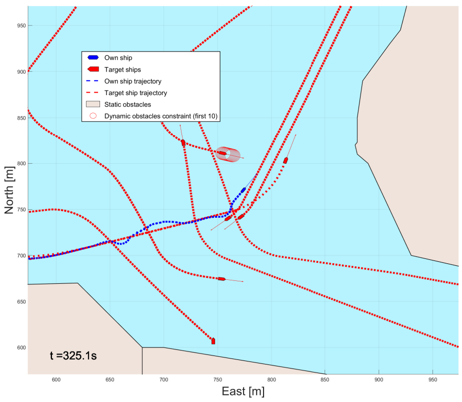
(l) mhm



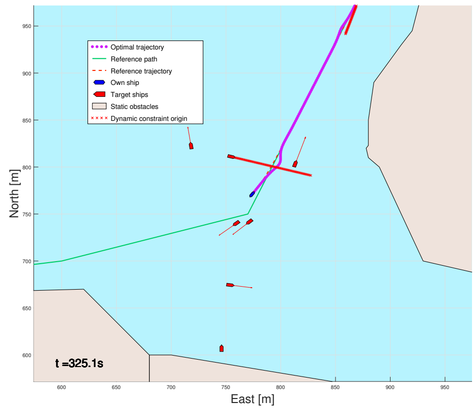
(m) caption



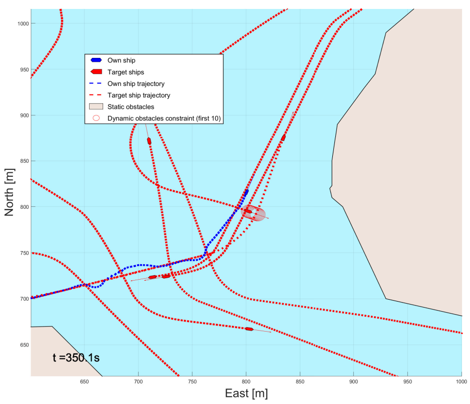
(n) mhm



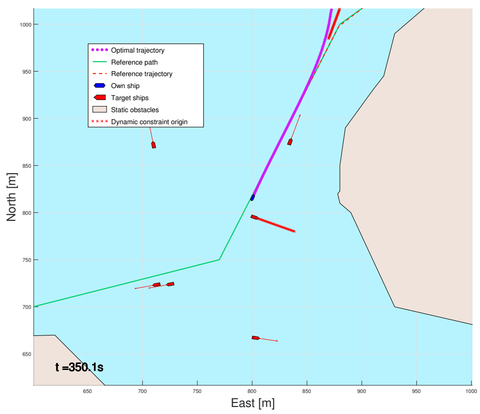
(o) caption



(p) mhm

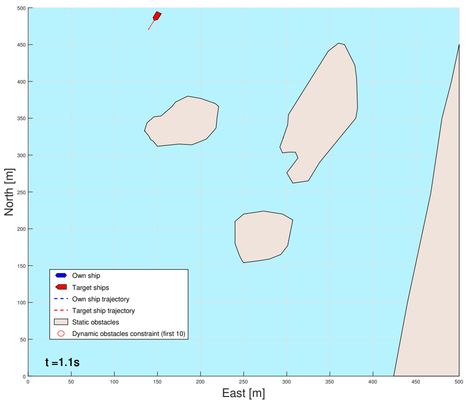


(q) caption

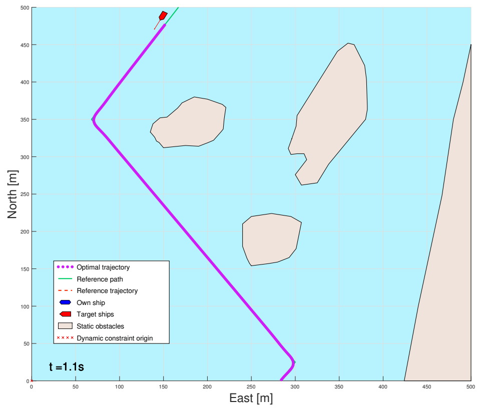


(r) mhm

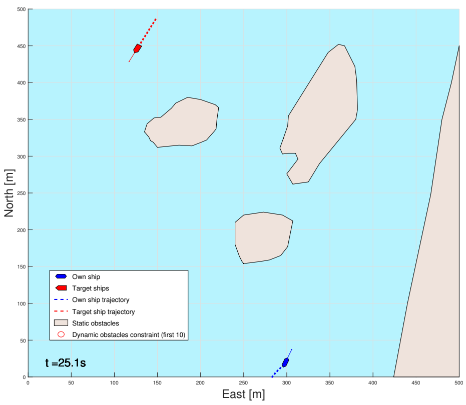
Figure 25: Trondheimfjord simulation. On the left shown with current active dynamic and static constraints. On the right seen with projected future trajectory



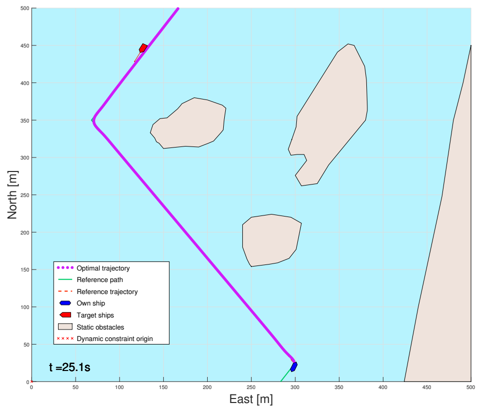
(a) caption



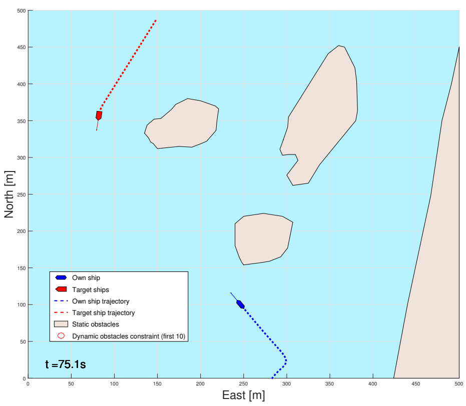
(b) mhm



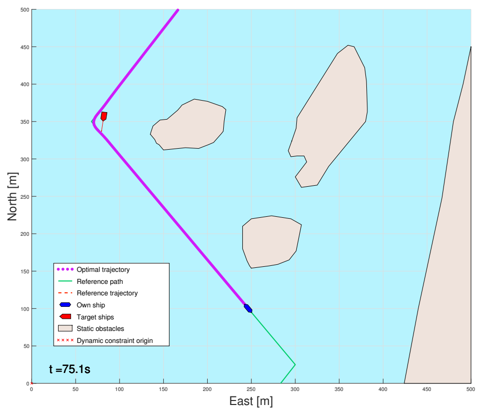
(c) caption



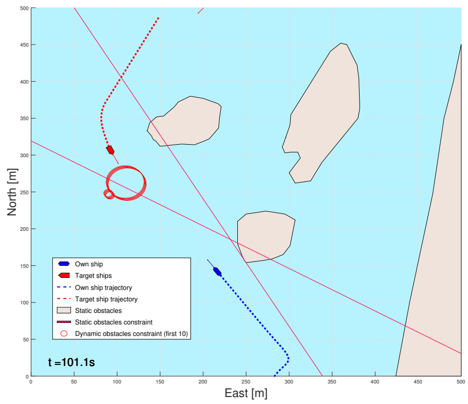
(d) mhm



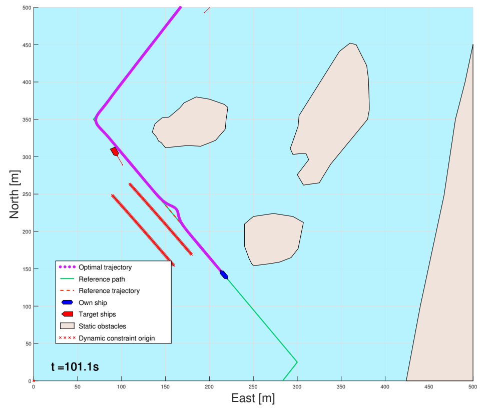
(e) caption



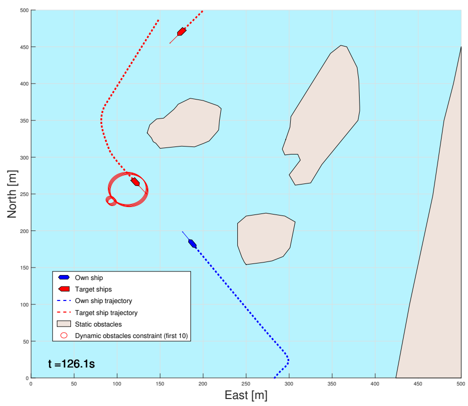
(f) mhm



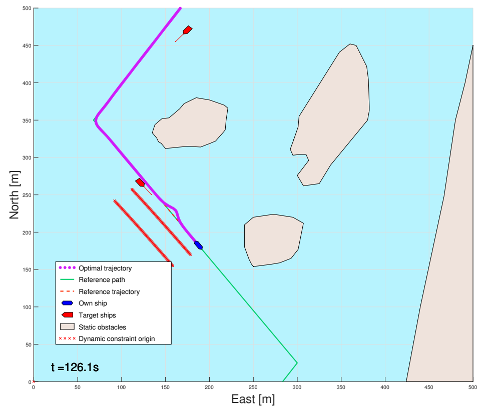
(g) caption



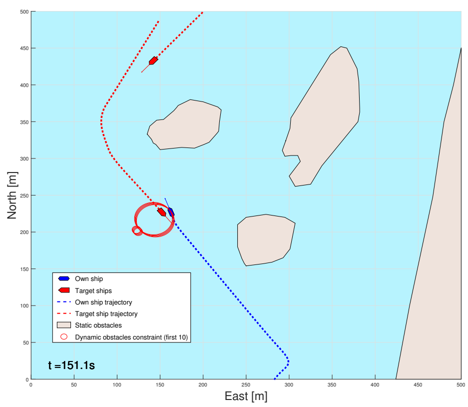
(h) mhm



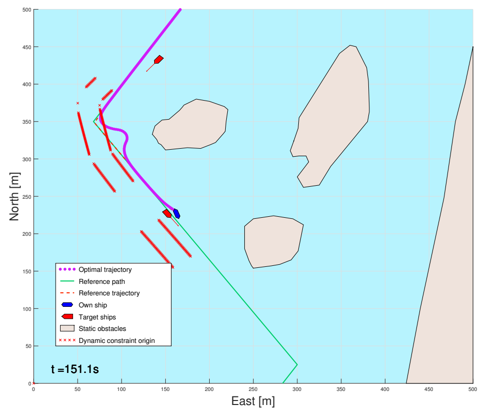
(i) caption



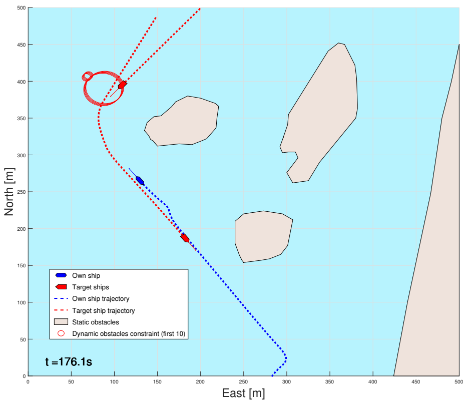
(j) mhm



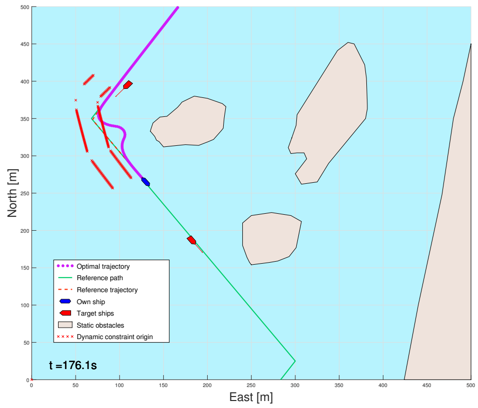
(k) caption



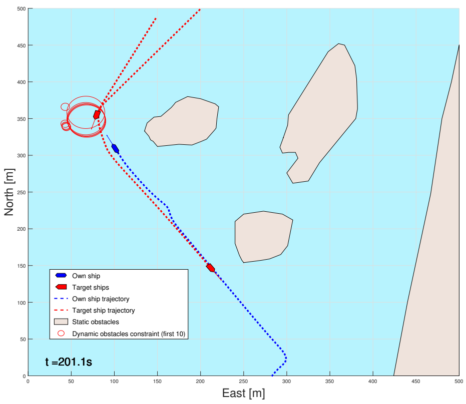
(l) mhm



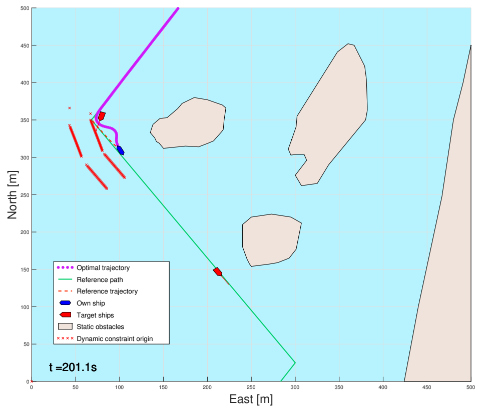
(m) caption



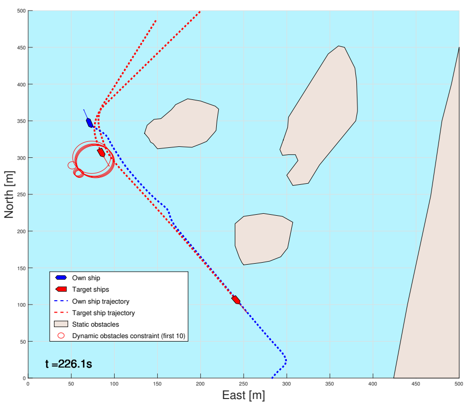
(n) mhm



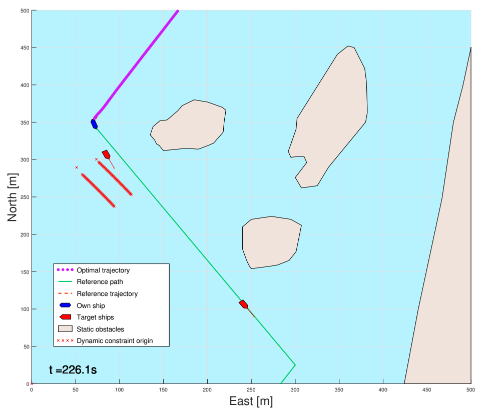
(o) caption



(p) mhm



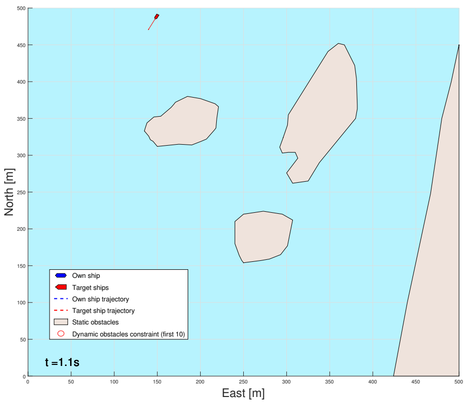
(q) caption



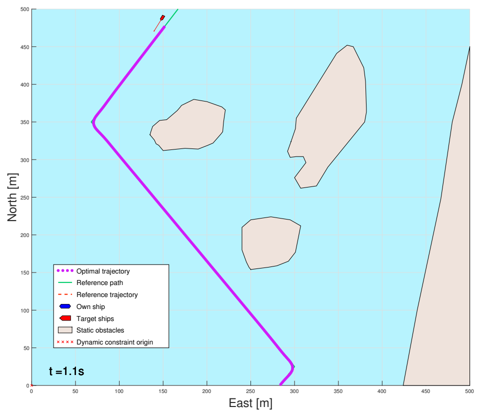
(r) mhm

Figure 26: Helloya simulation. On the left shown with current active dynamic and static constraints. On the right seen with projected future trajectory

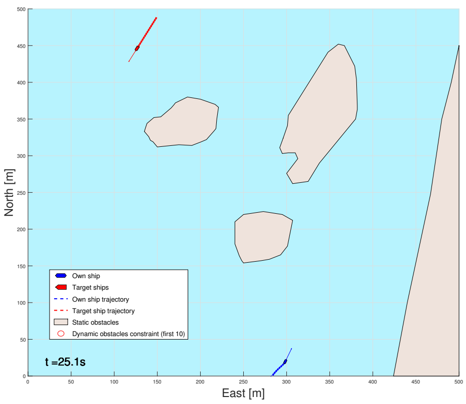
4.2.10 Helløya Reversed



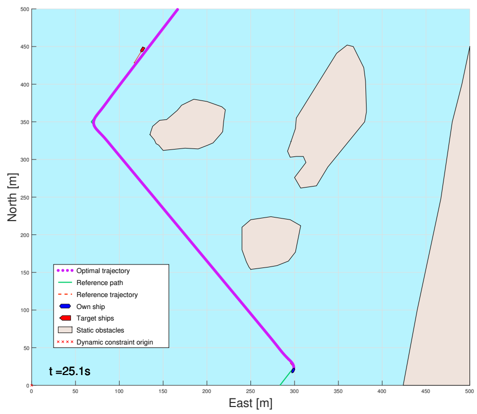
(a) caption



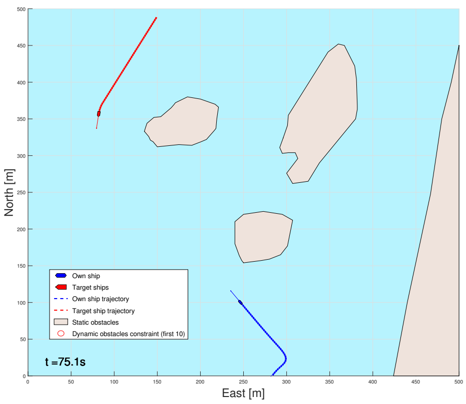
(b) mhm



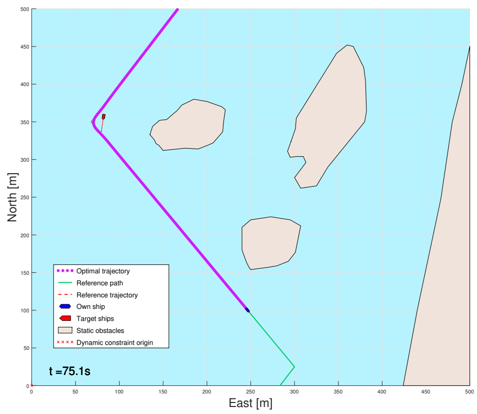
(c) caption



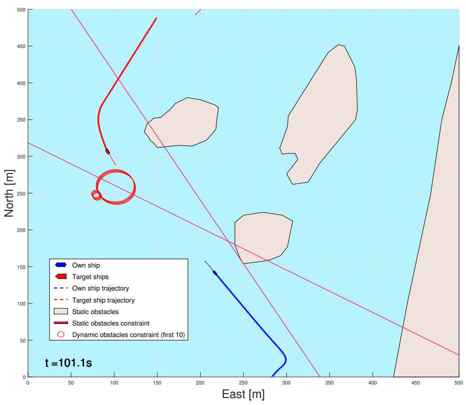
(d) mhm



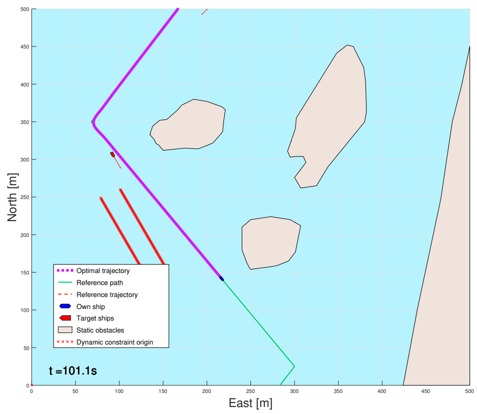
(e) caption



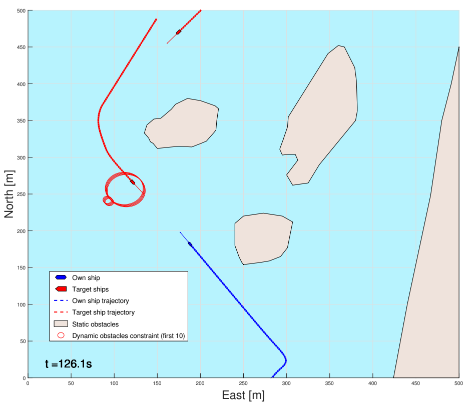
(f) mhm



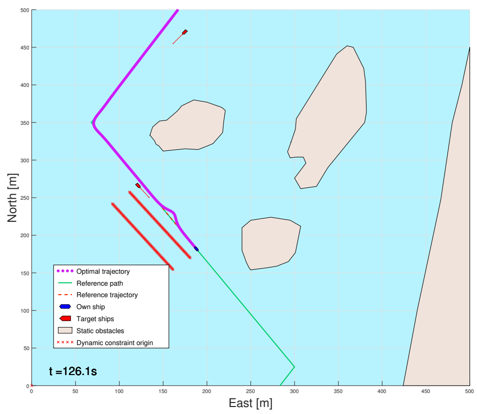
(g) caption



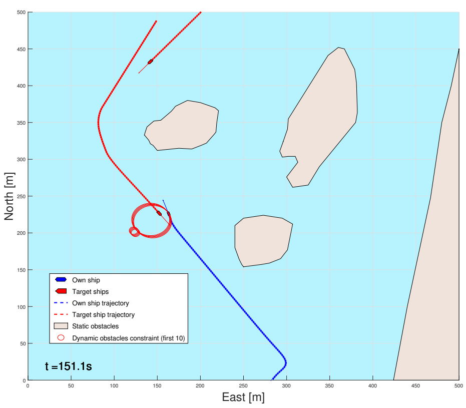
(h) mhm



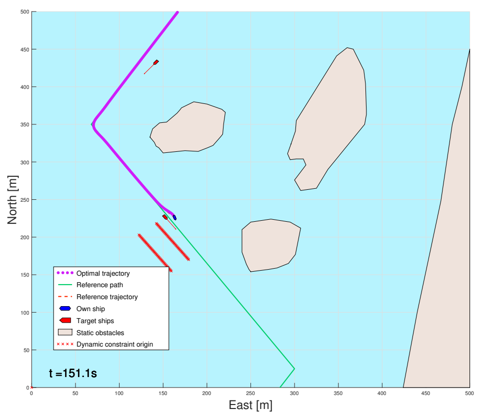
(i) caption



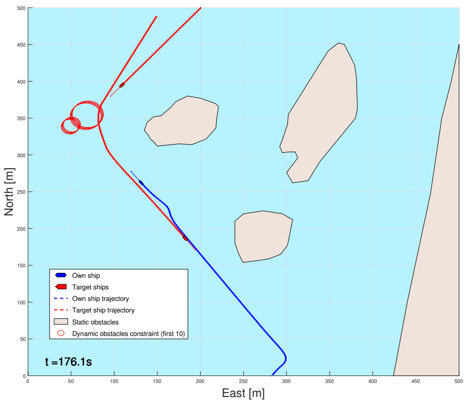
(j) mhm



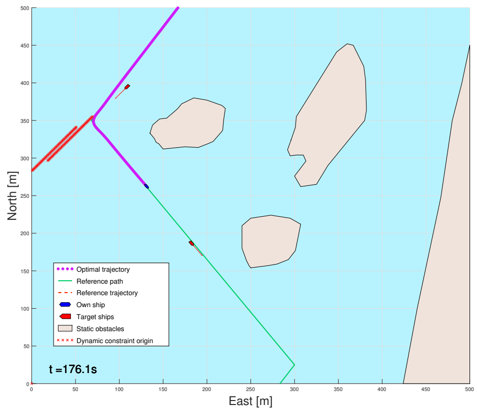
(k) caption



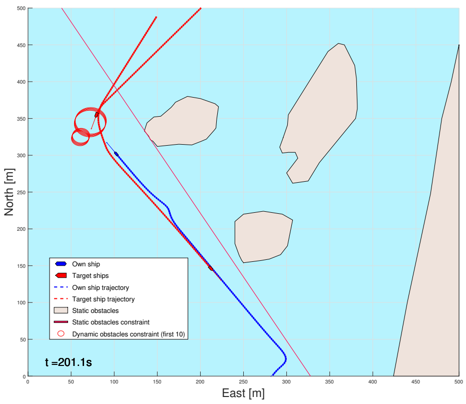
(l) mhm



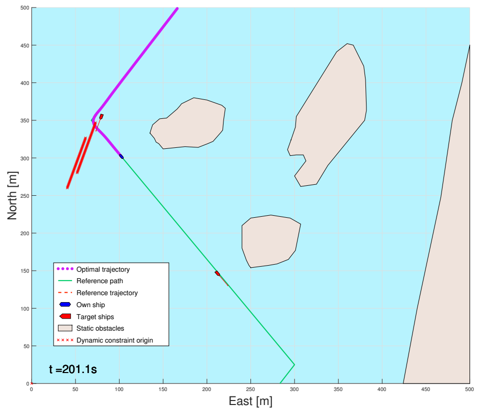
(m) caption



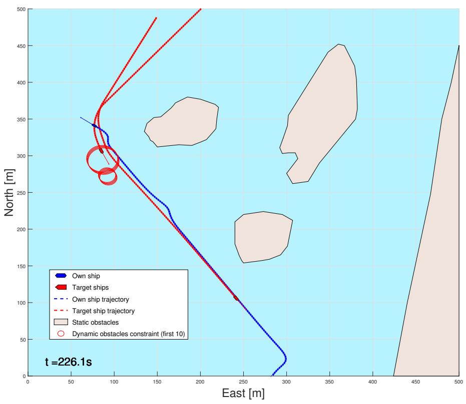
(n) mhm



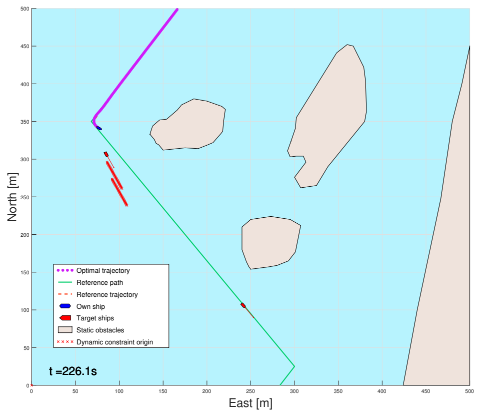
(o) caption



(p) mhm

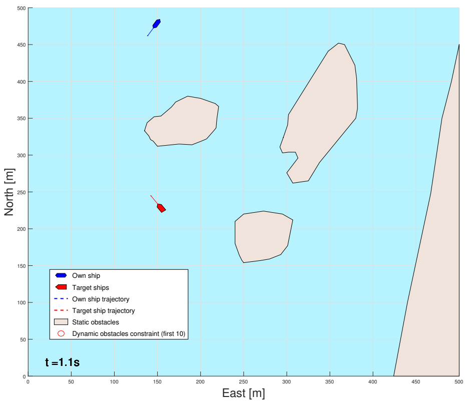


(q) caption

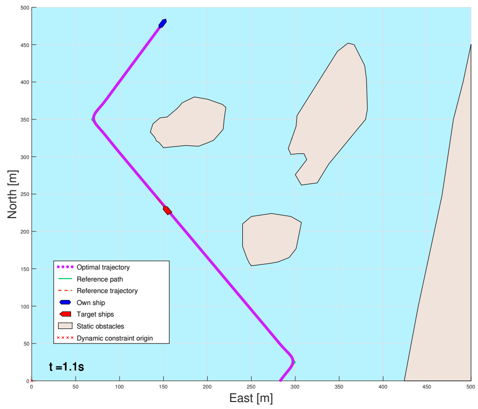


(r) mhm

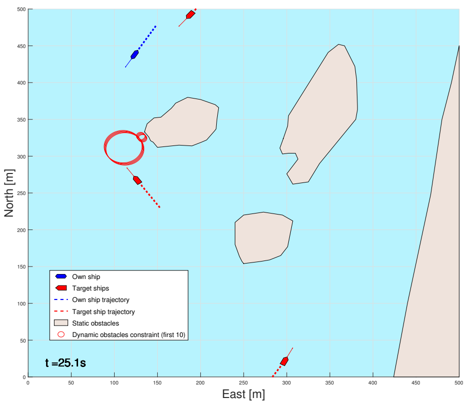
Figure 27: Helloya simulation. On the left shown with current active dynamic and static constraints. On the right seen with projected future trajectory



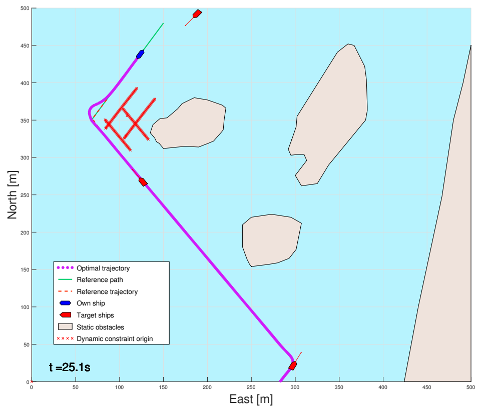
(a) caption



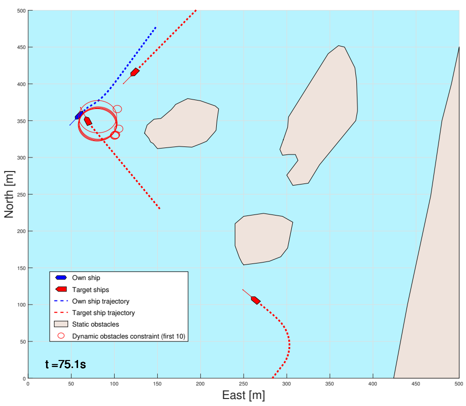
(b) mhm



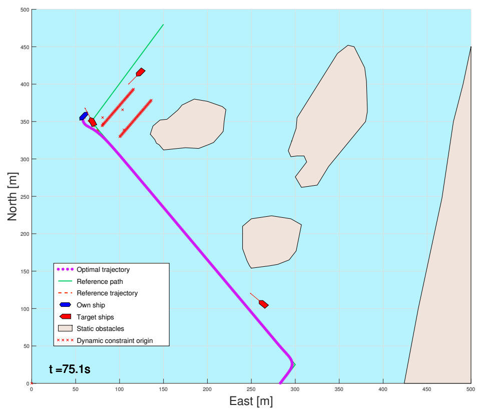
(c) caption



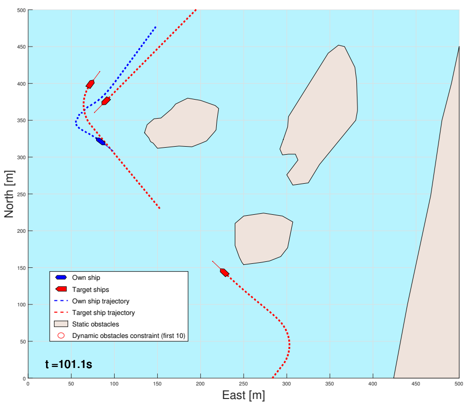
(d) mhm



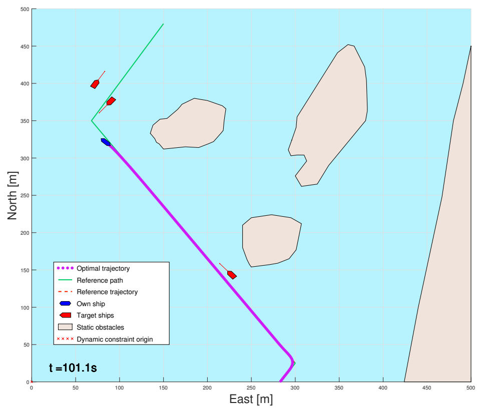
(e) caption



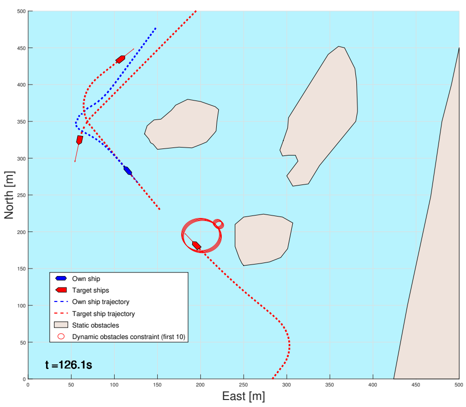
(f) mhm



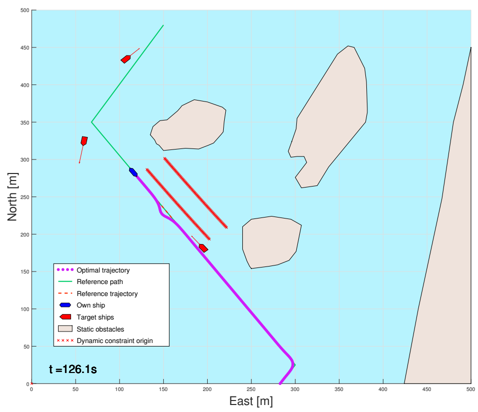
(g) caption



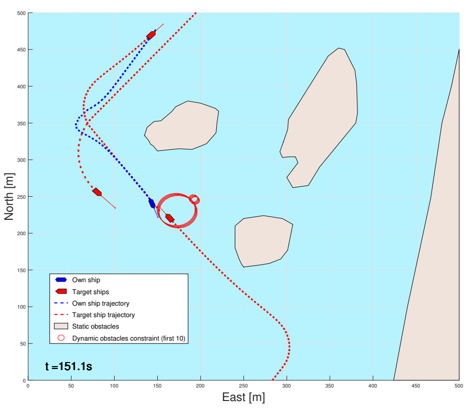
(h) mhm



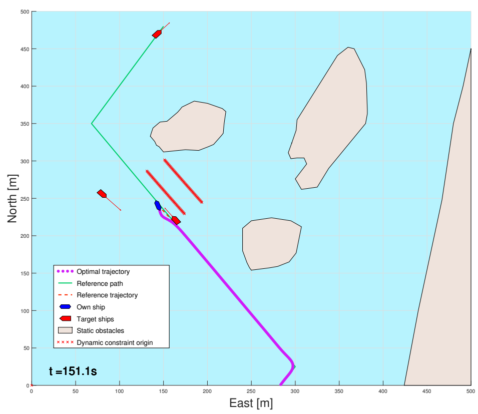
(i) caption



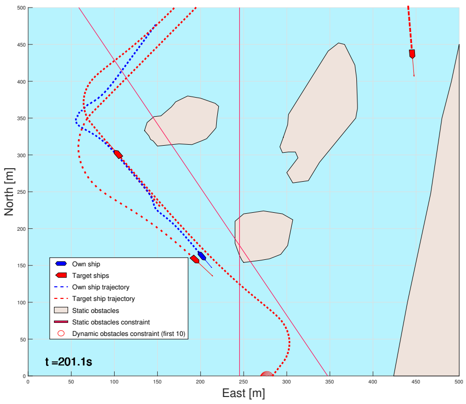
(j) mhm



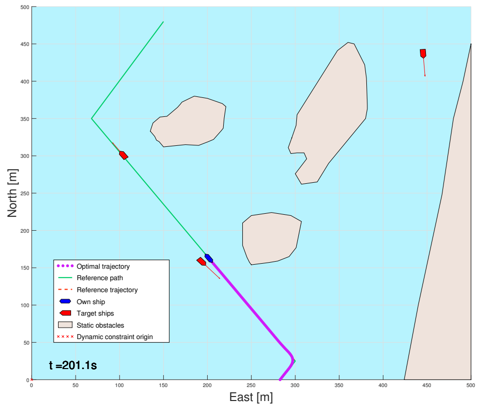
(k) caption



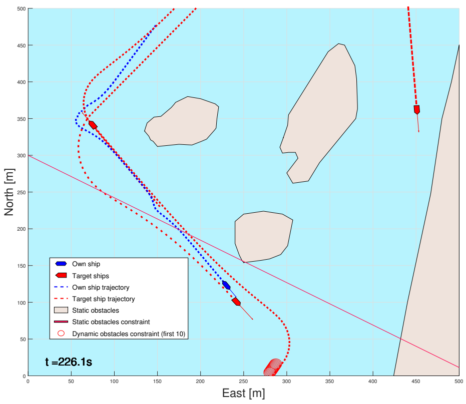
(l) mhm



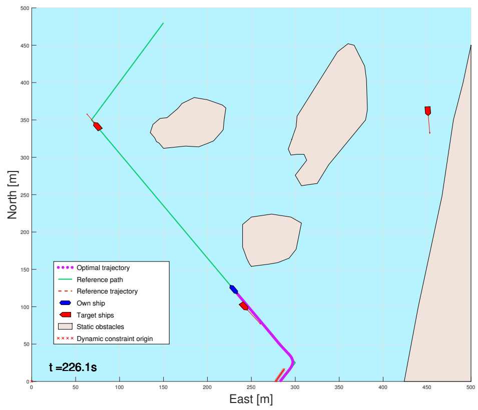
(m) caption



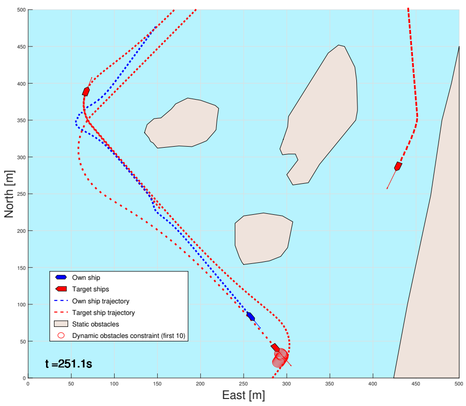
(n) mhm



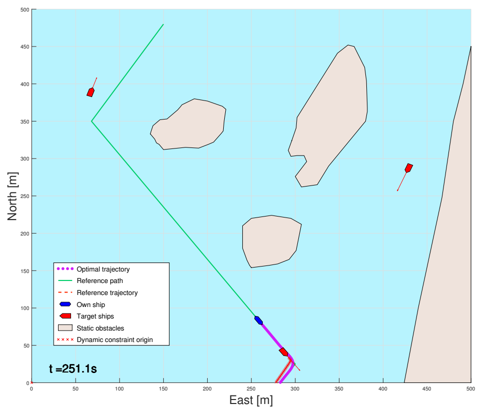
(o) caption



(p) mhm



(q) caption

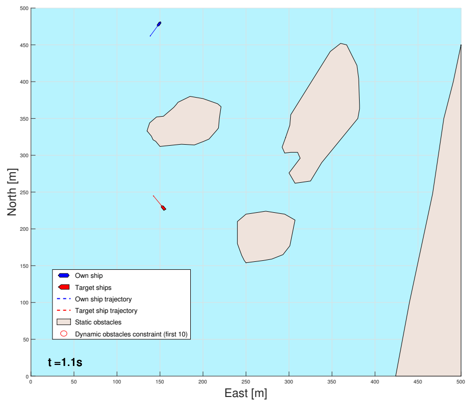


(r) mhm

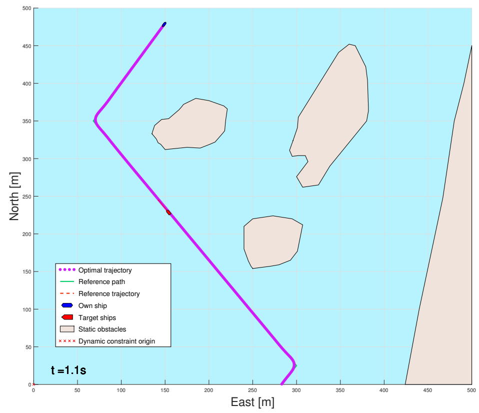
Figure 28: Helloya Reverse simulation. On the left shown with current active dynamic and static constraints. On the right seen with projected future trajectory

4.2.11 Skjærgård with traffic

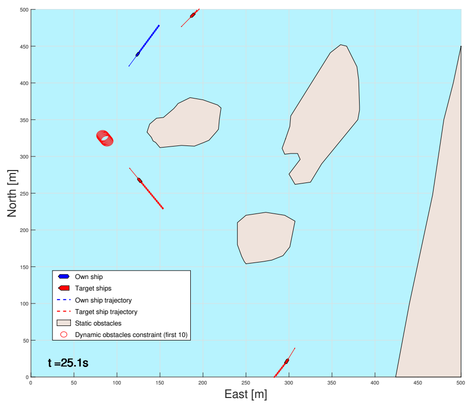
- Both prediction levels successfully navigate this scenario
- with simple prediction the algorithm has an absolute nightmare trying to get past the third TS, this sim took hours to run.



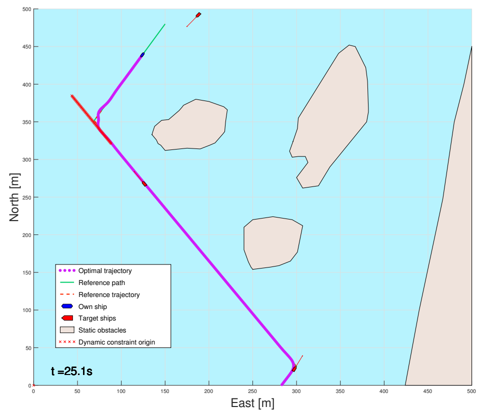
(a) caption



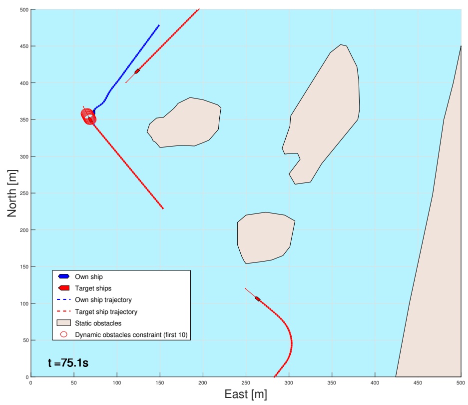
(b) mhm



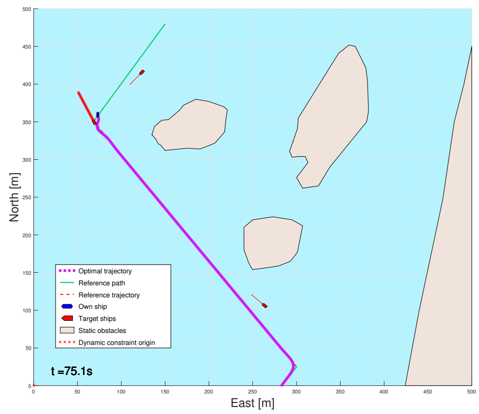
(c) caption



(d) mhm

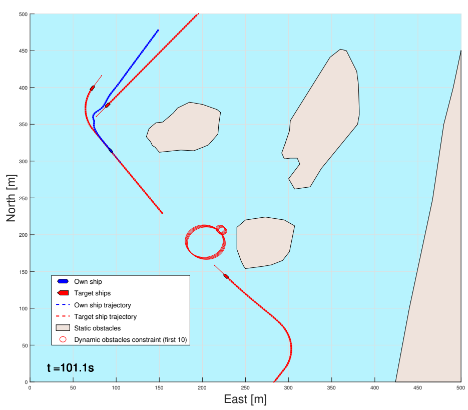


(e) caption

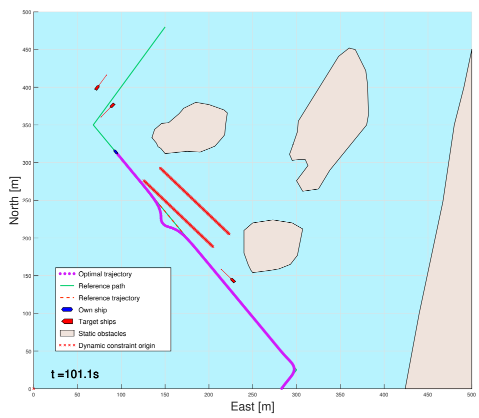


(f) mhm

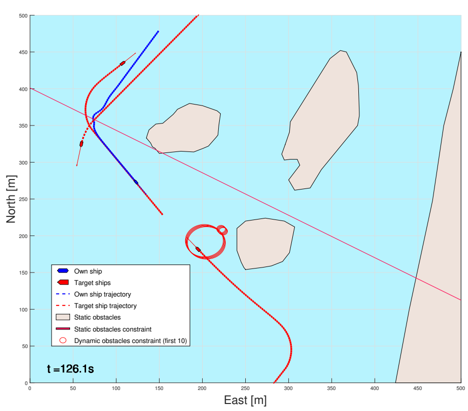
- an interesting problem with 'islands' is that the optimal trajectory could get stuck inside one.
- Tried experimenting with placing bigger and bigger islands on top of the reference path, did not go too well.



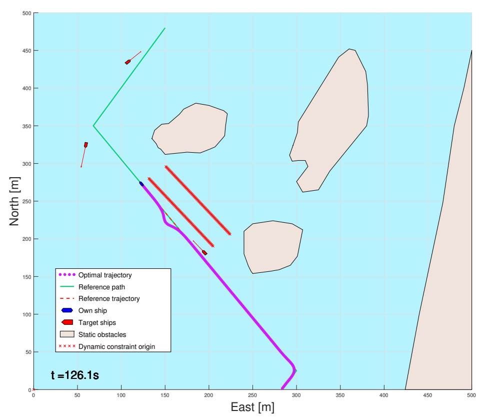
(g) caption



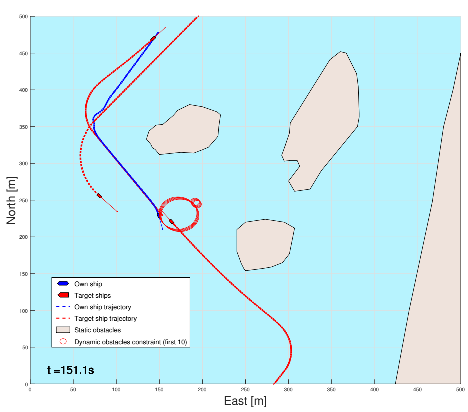
(h) mhm



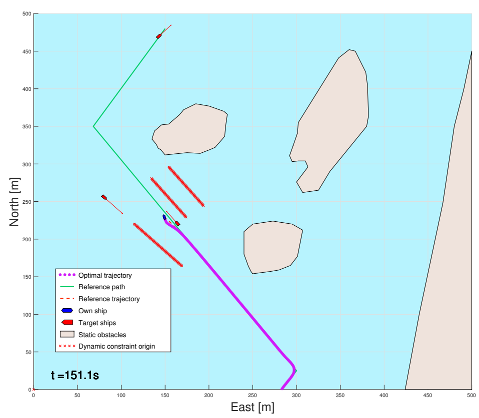
(i) caption



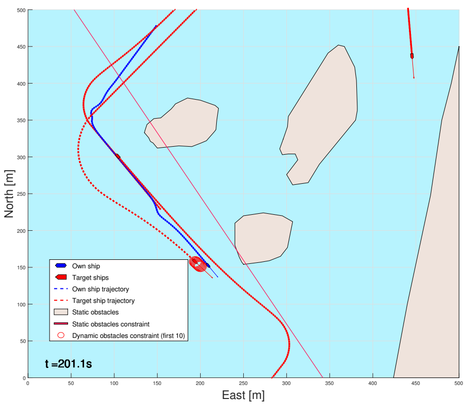
(j) mhm



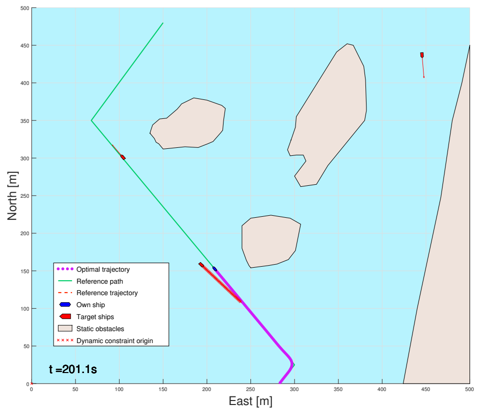
(k) caption



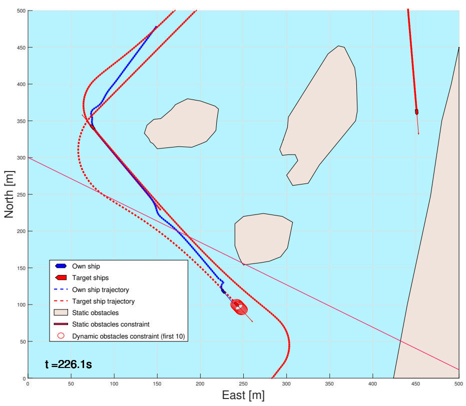
(l) mhm



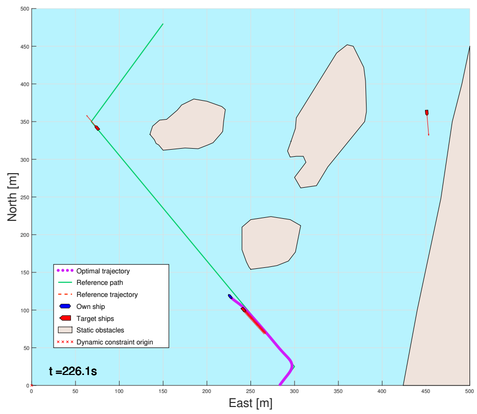
(m) caption



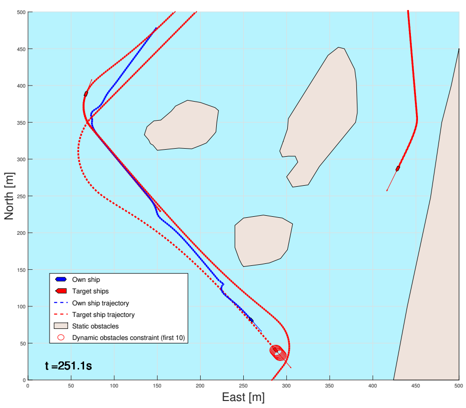
(n) mhm



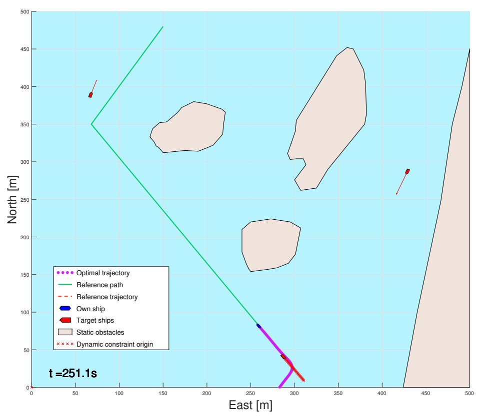
(o) caption



(p) mhm

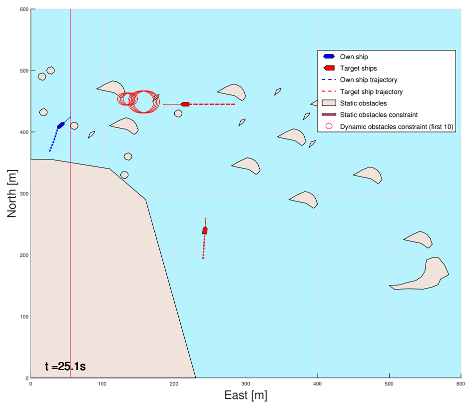


(q) caption

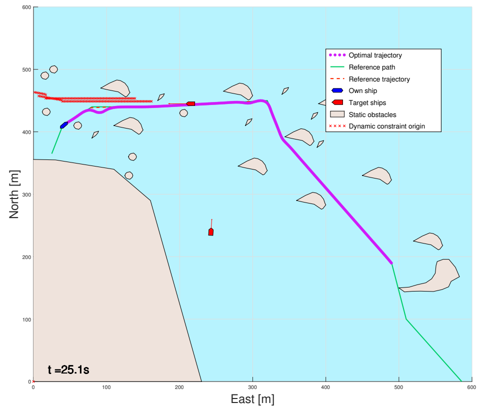


(r) mhm

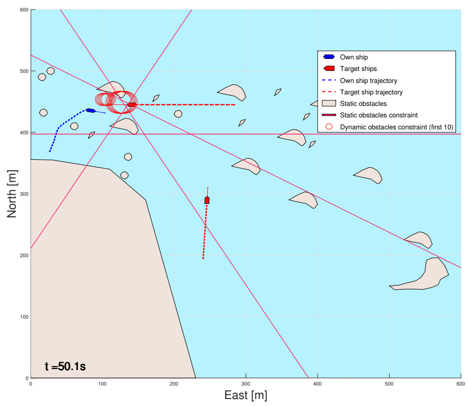
Figure 29: Helloya Reverse simulation. On the left shown with current active dynamic and static constraints. On the right seen with projected future trajectory



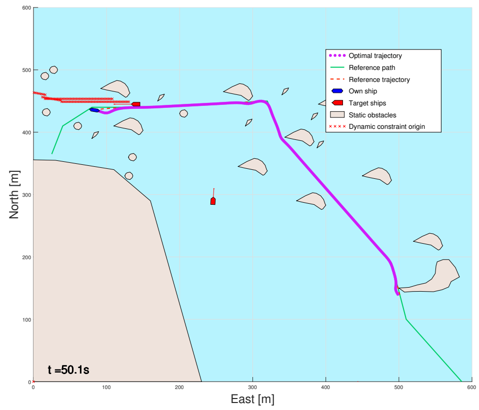
(a) caption



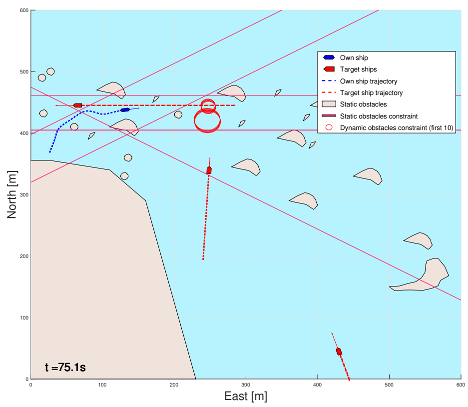
(b) mhm



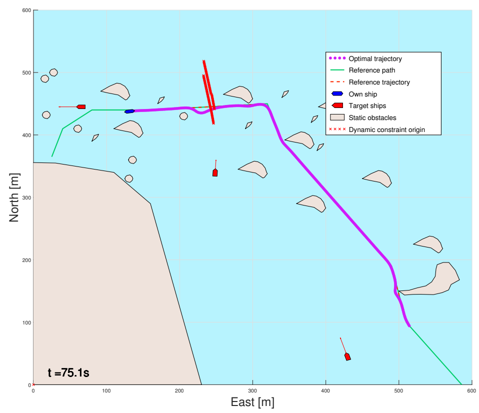
(c) caption



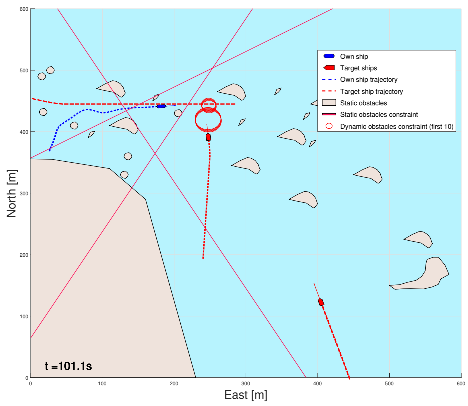
(d) mhm



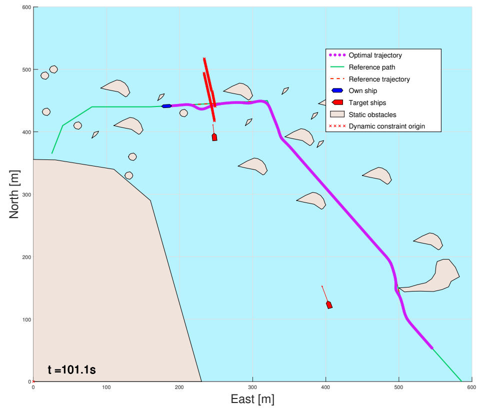
(e) caption



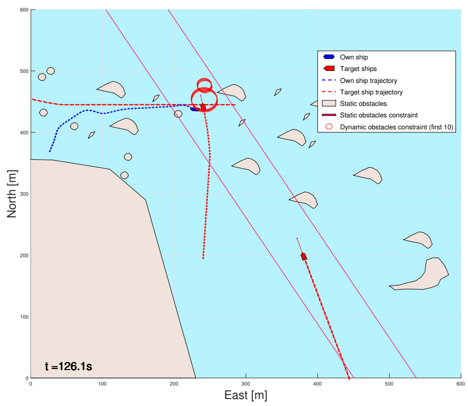
(f) mhm



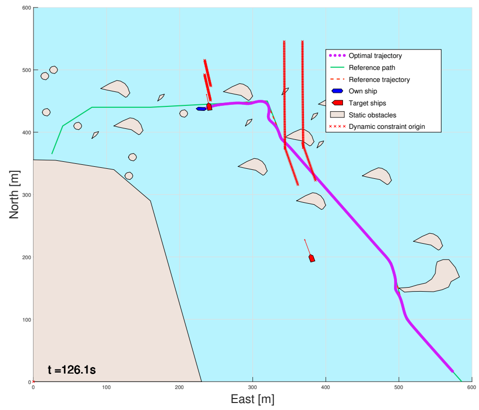
(g) caption



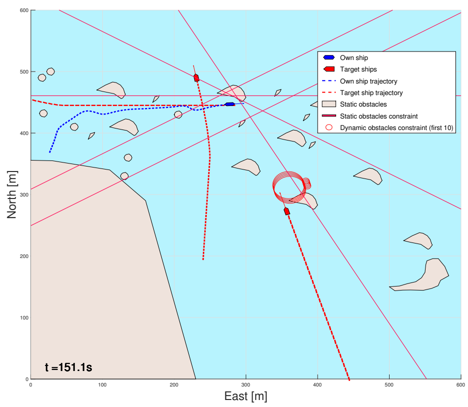
(h) mhm



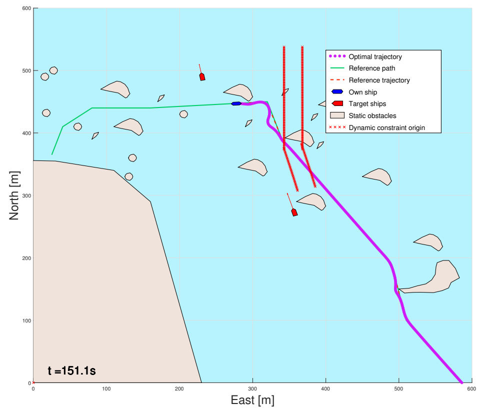
(i) caption



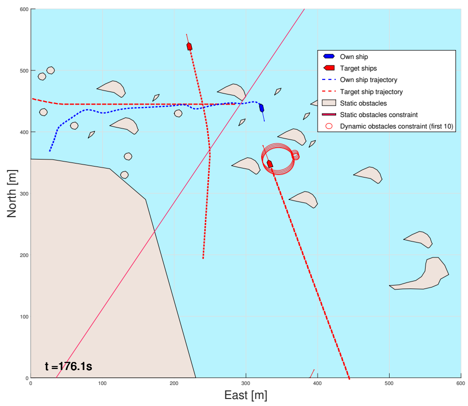
(j) mhm



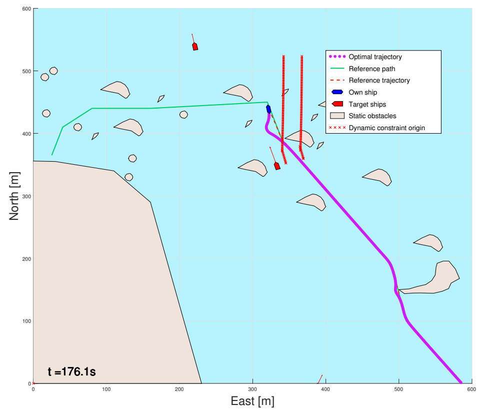
(k) caption



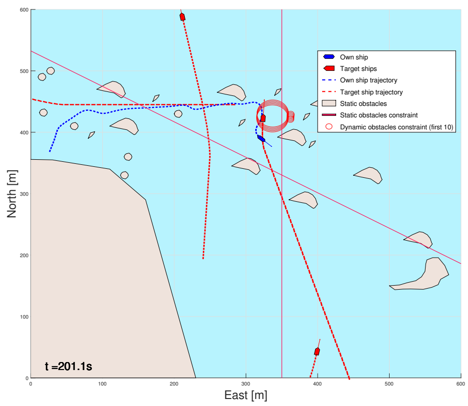
(l) mhm



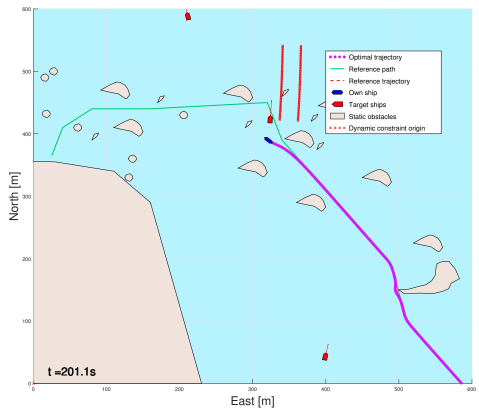
(m) caption



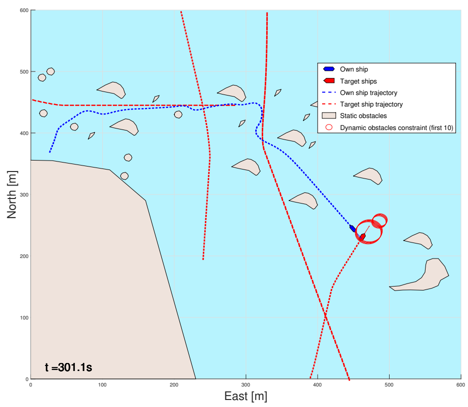
(n) mhm



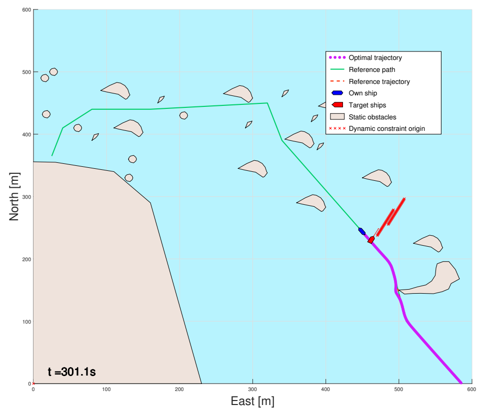
(o) caption



(p) mhm



(q) caption

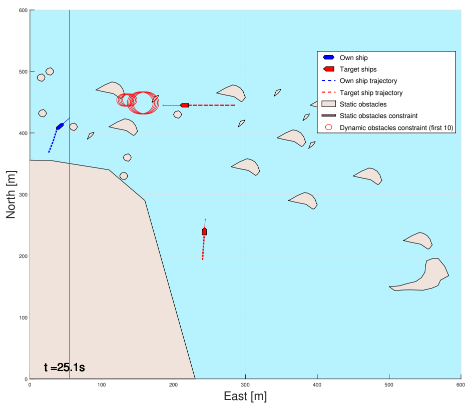


(r) mhm

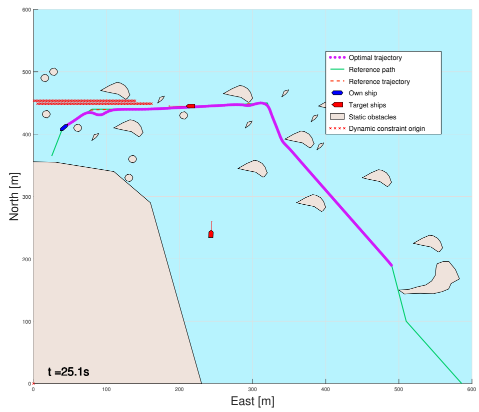
Figure 30: SKJÆRGÅRD WITH TRAFFIC. On the left shown with current active dynamic and static constraints. On the right seen with projected future trajectory

4.2.12 skjærgård without traffic

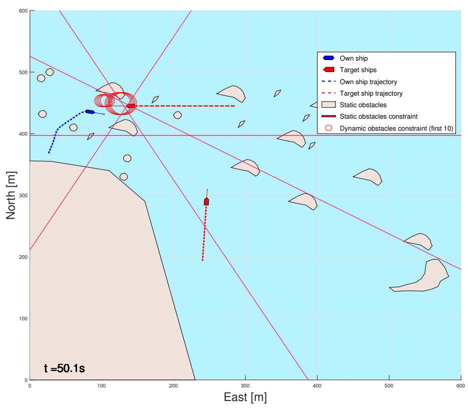
- Good path tracking.
- this is where I discovered a heading reference problem.
- able to dodge islands that are in the way.



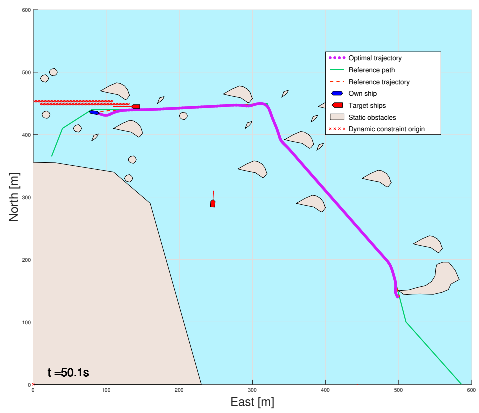
(a) caption



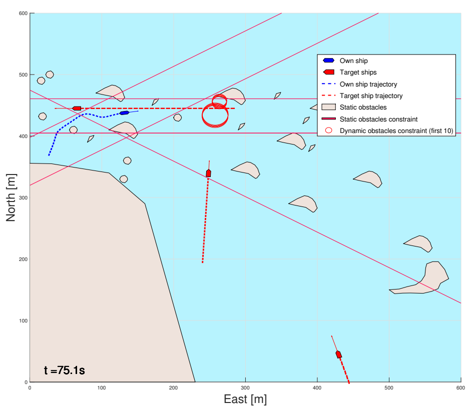
(b) mhm



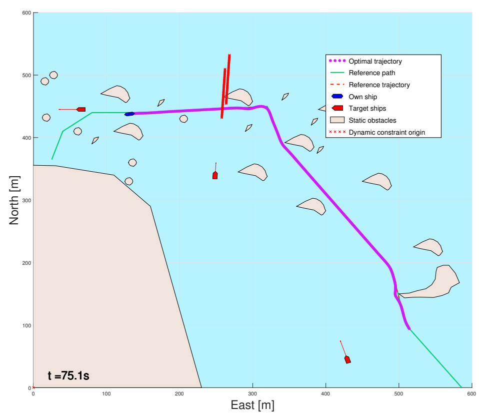
(c) caption



(d) mhm

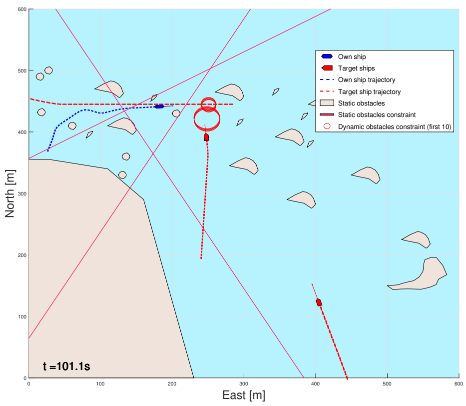


(e) caption

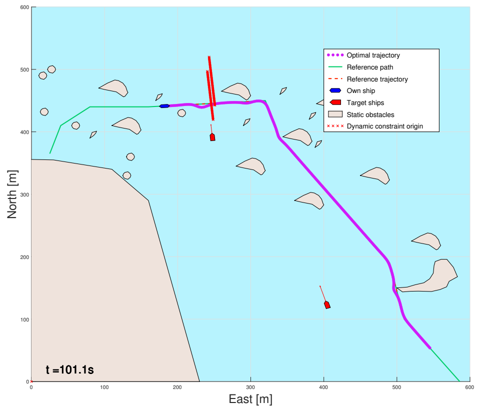


(f) mhm

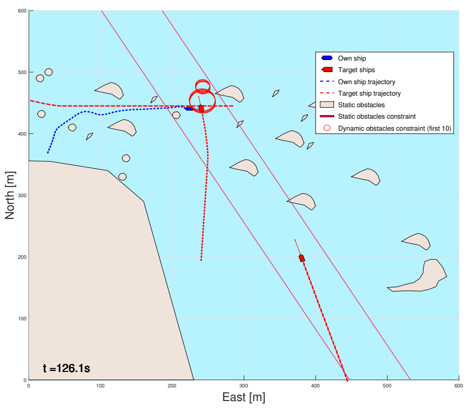
- computationally not too difficult either, algorithm exhibit reasonable execution times.



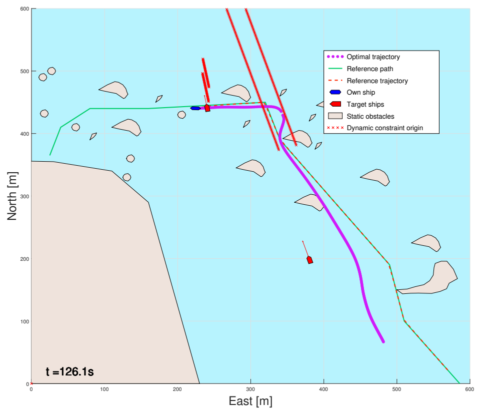
(g) caption



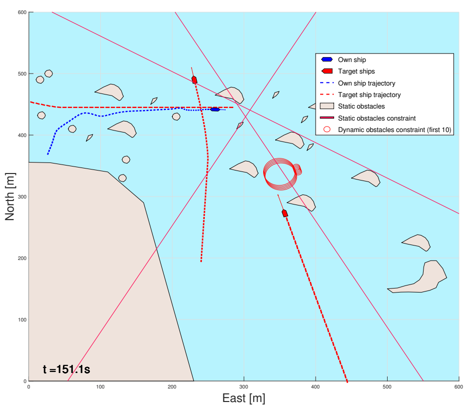
(h) mhm



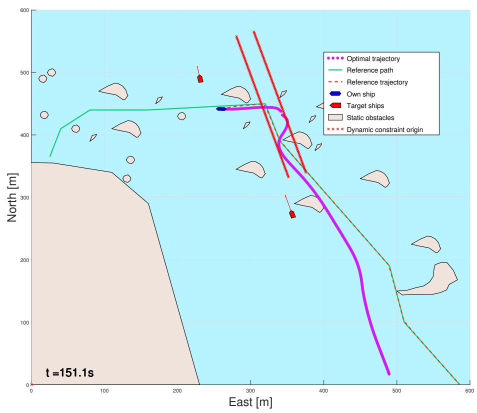
(i) caption



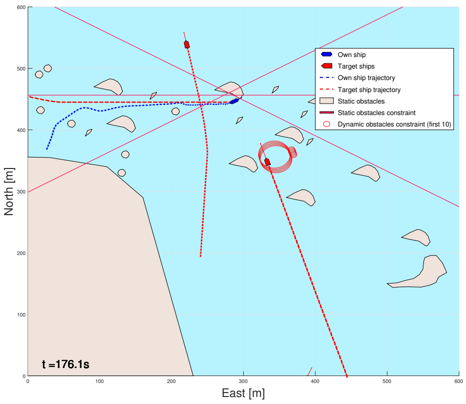
(j) mhm



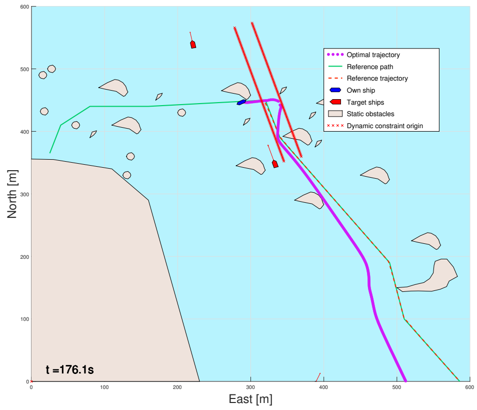
(k) caption



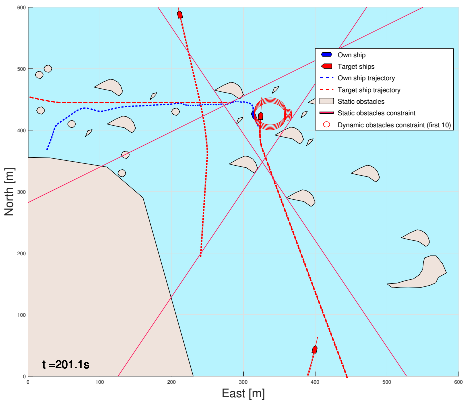
(l) mhm



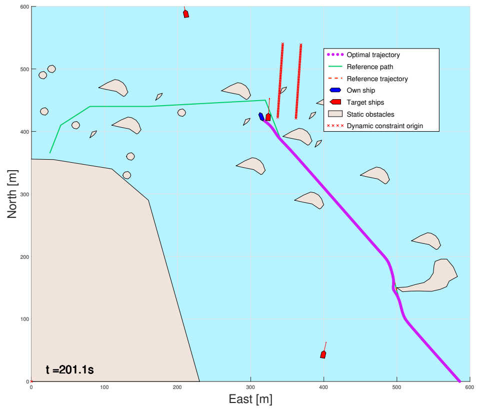
(m) caption



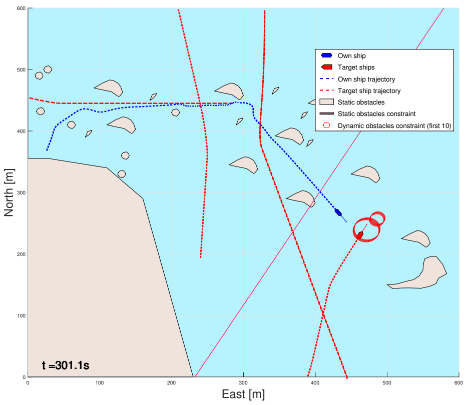
(n) mhm



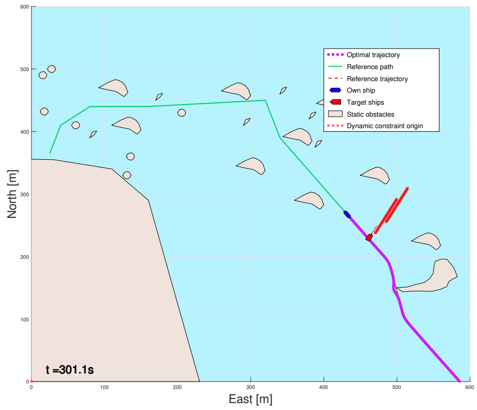
(o) caption



(p) mhm



(q) caption

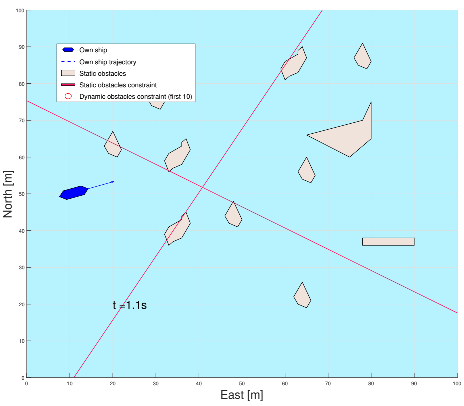


(r) mhm

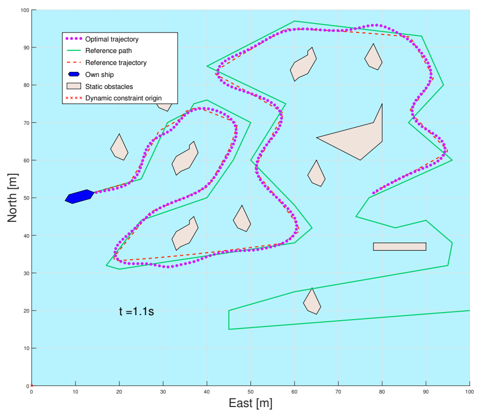
Figure 31: SKJÆRGÅRD WITH TRAFFIC. On the left shown with current active dynamic and static constraints. On the right seen with projected future trajectory

4.2.13 Miscellaneous

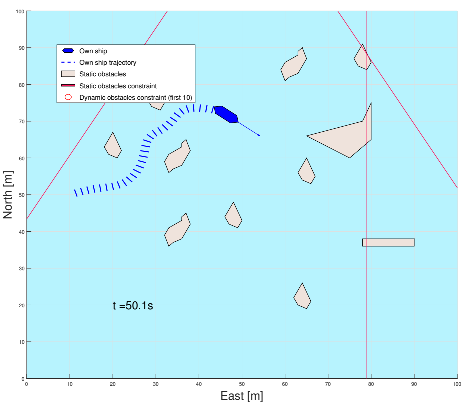
- Bad Prediction, what happens when the target ship does not follow the predicted path
- Blocked Path, a closer look at what happens when the path we intend to take is fully blocked.
- Wrong turn, observe that the optimal trajectory is often to turn the wrong direction slightly



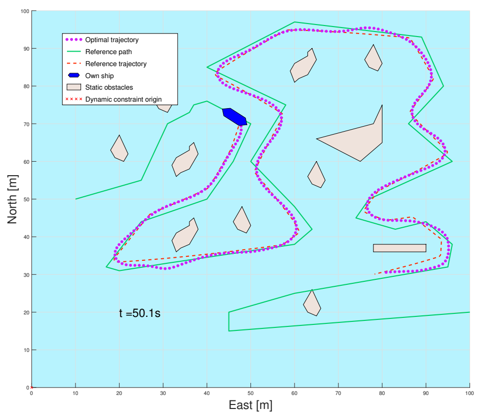
(a) caption



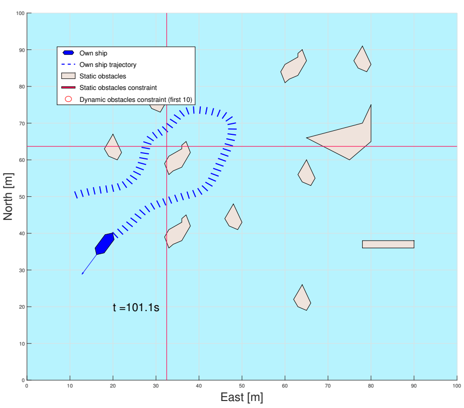
(b) mhm



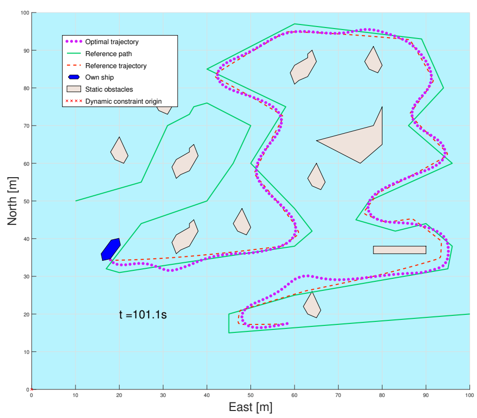
(c) caption



(d) mhm



(e) caption



(f) mhm

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.

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(~isempty(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.

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). *Maneuvers to avoid collision A Guide to the Collision Avoidance Rules (pp. 170-172).*

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.

Ruud, Kristian Amundsen, Edmund Førland Brekke and Jo Eidsvik (2018). 'Lidar extended object tracking of a maritime vessel using an ellipsoidal contour model'. In: *2018 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*. IEEE, pp. 1–6.

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.

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.

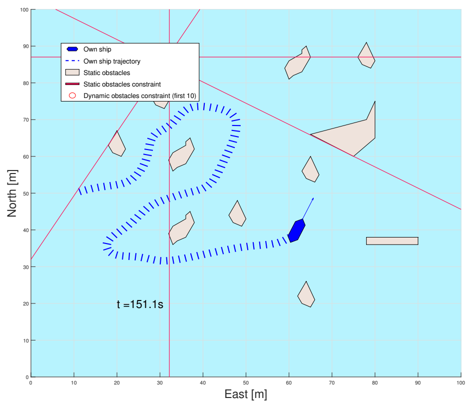
Wächter, Andreas and Lorenz T Biegler (2006). 'On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming'. In: *Mathematical programming* 106.1, pp. 25–57.

WÄRTSILÄ (2018). *Wärtsilä achieves notable advances in automated shipping with latest successful tests*. Web article. URL: https://www.wartsila.com/media/news/28-11-2018-wartsila-achieves-notable-advances-in-automated-shipping-with-latest-successful-tests-2332144.

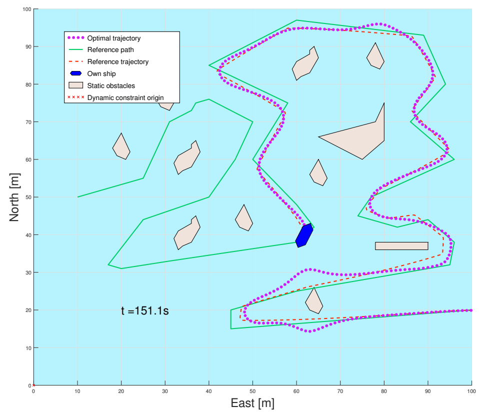
Woerner, Kyle (2016). 'Multi-contact protocol-constrained collision avoidance for autonomous marine vehicles'. PhD thesis. Massachusetts Institute of Technology, USA.

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

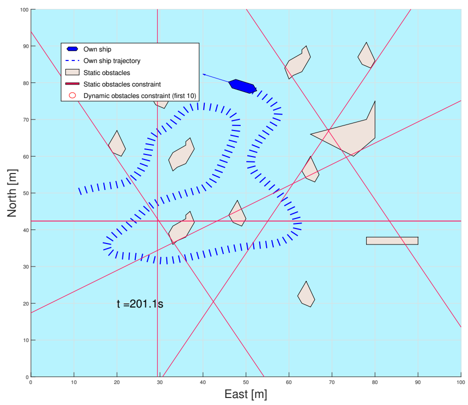
Zhang, Xinyu, Chengbo Wang, Lingling Jiang, Lanxuan An and Rui Yang (2021). 'Collision-avoidance navigation systems for Maritime Autonomous Surface Ships: A state of the art survey'. In: *Ocean Engineering* 235, p. 109380.



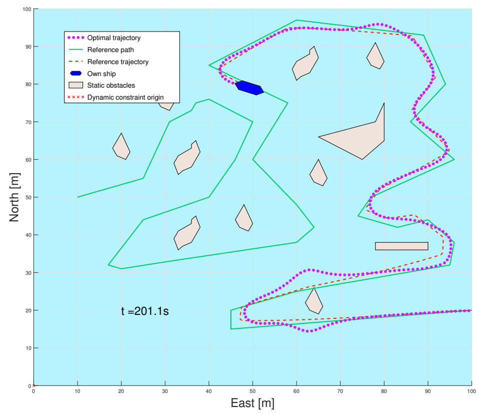
(g) caption



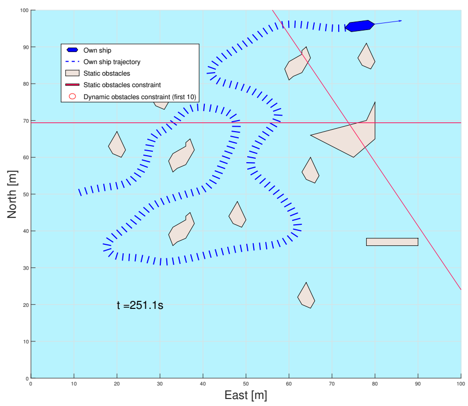
(h) mhm



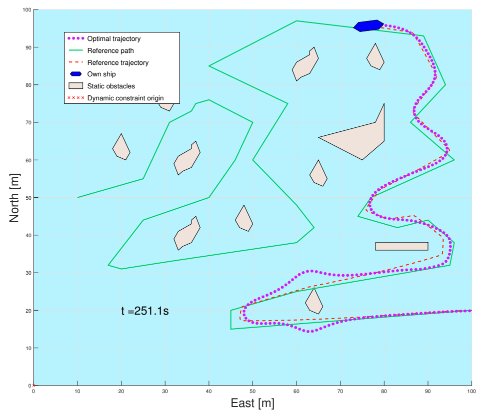
(i) caption



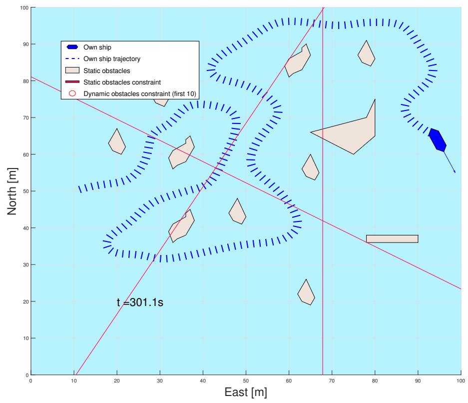
(j) mhm



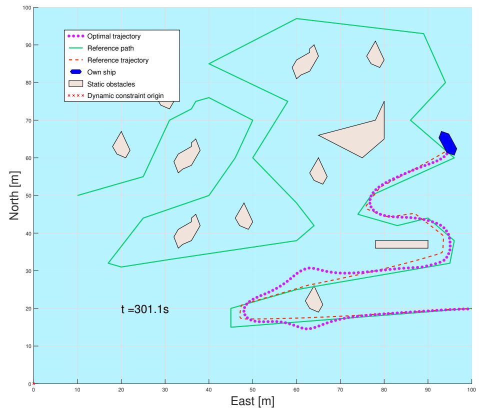
(k) caption



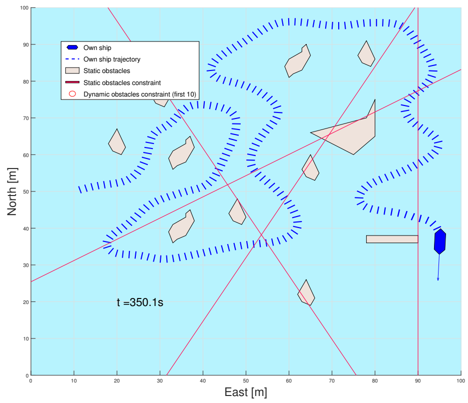
(l) mhm



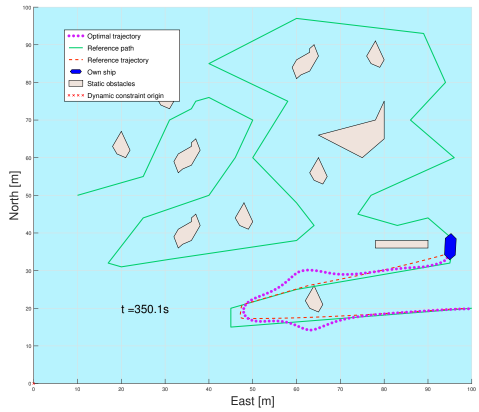
(m) caption



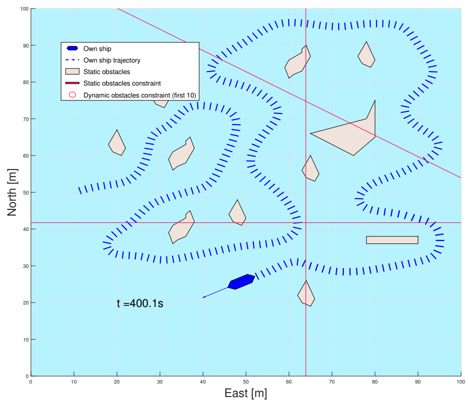
(n) mhm



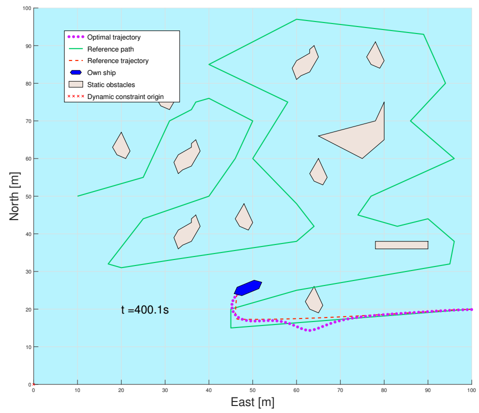
(o) caption



(p) mhm

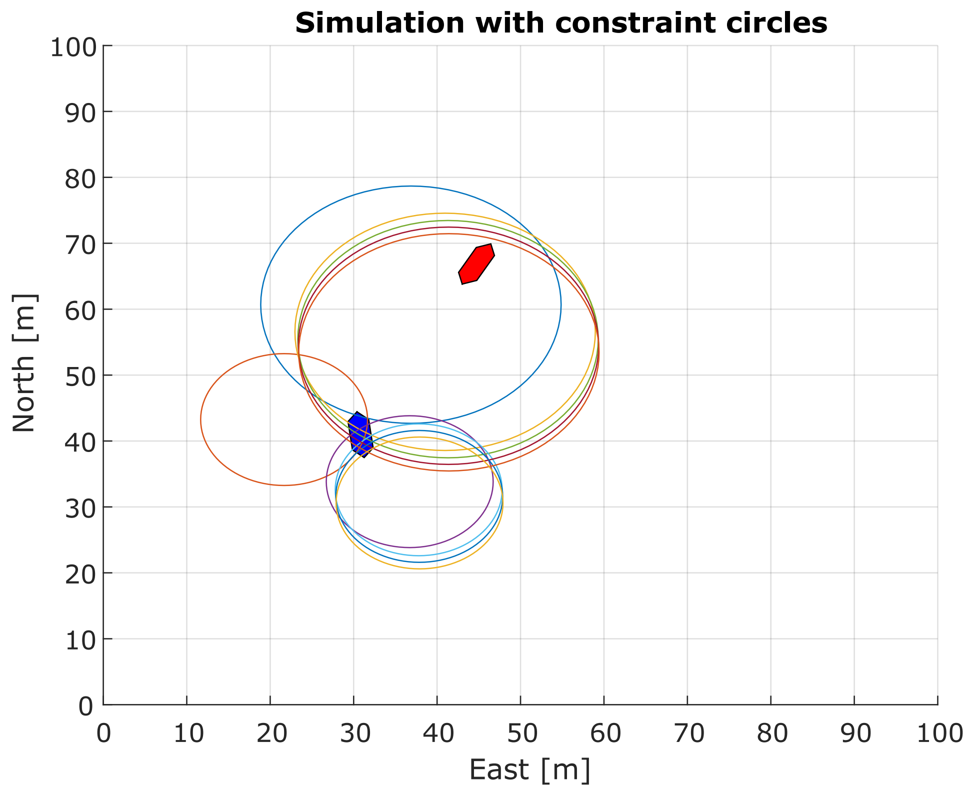


(q) caption

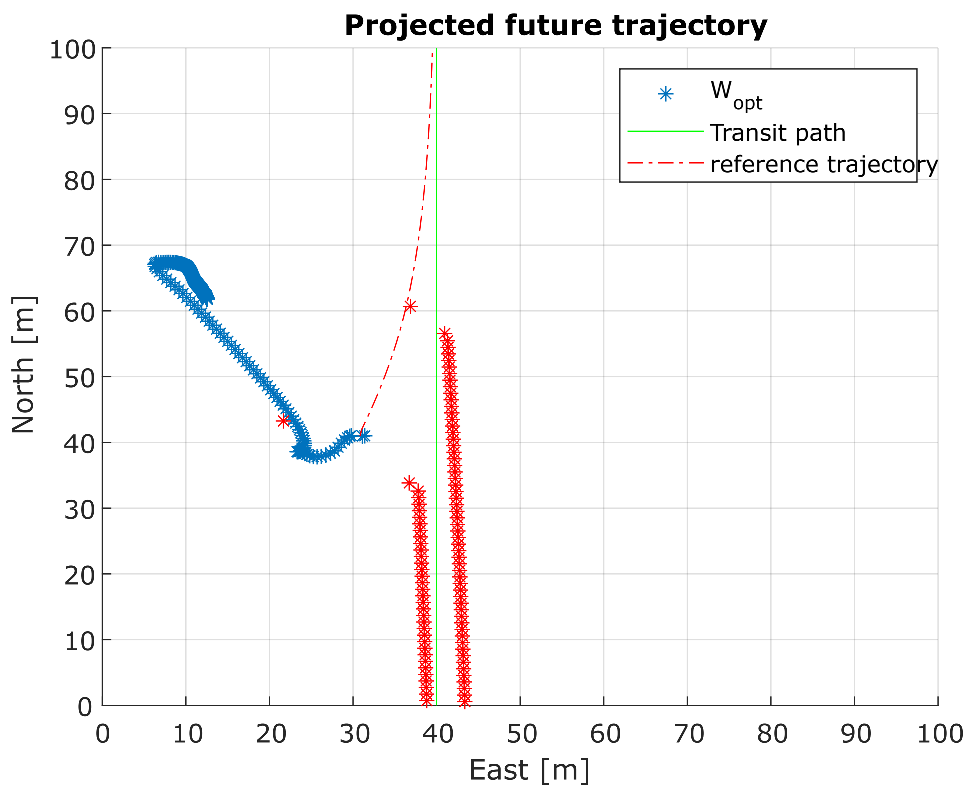


(r) mhm

Figure 32: SKJÆRGÅRD WITH TRAFFIC. On the left shown with current active dynamic and static constraints. On the right seen with projected future trajectory

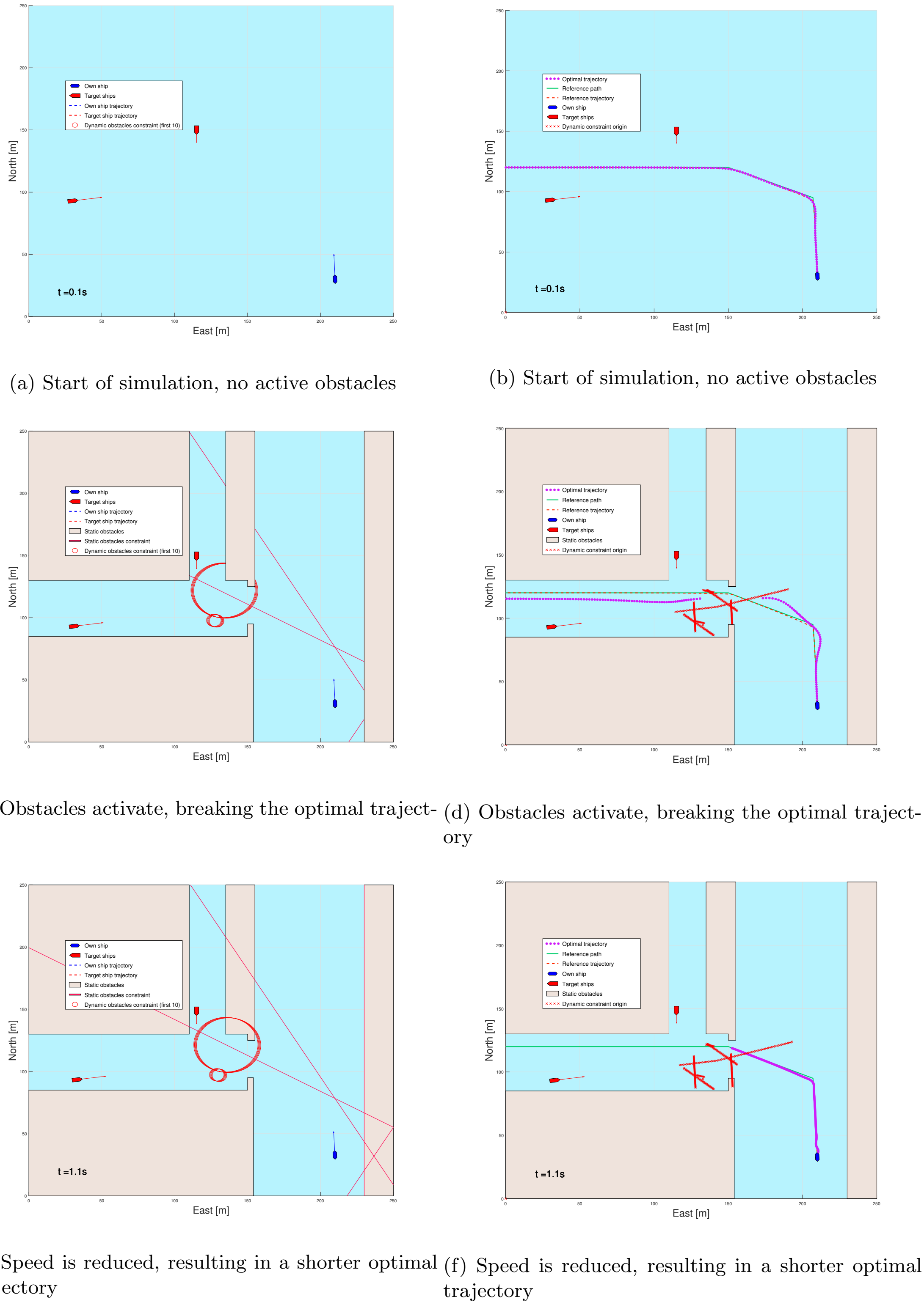


(a) When prediction goes wrong, the OS can get caught by moving constraints. (Old style figure)



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

Figure 33: 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

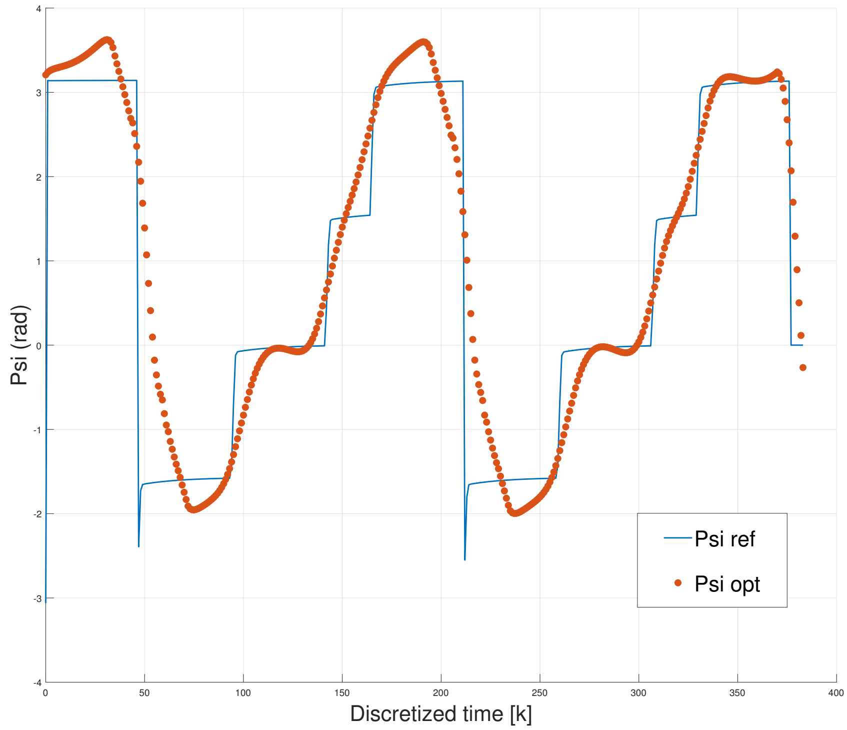
(c) Obstacles activate, breaking the optimal trajectory

(d) Obstacles activate, breaking the optimal trajectory

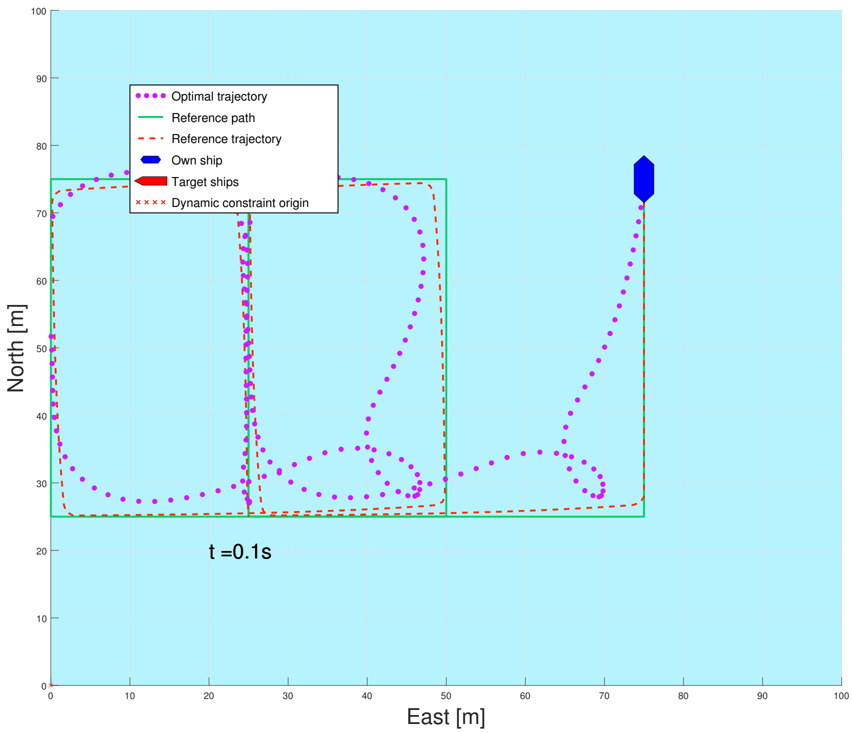
(e) Speed is reduced, resulting in a shorter optimal trajectory

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

Figure 34: How optimal path is calculated with lower speed when infeasibility is detected.



(a) ref



(b) wopt

Figure 35: this is a problem for sure.

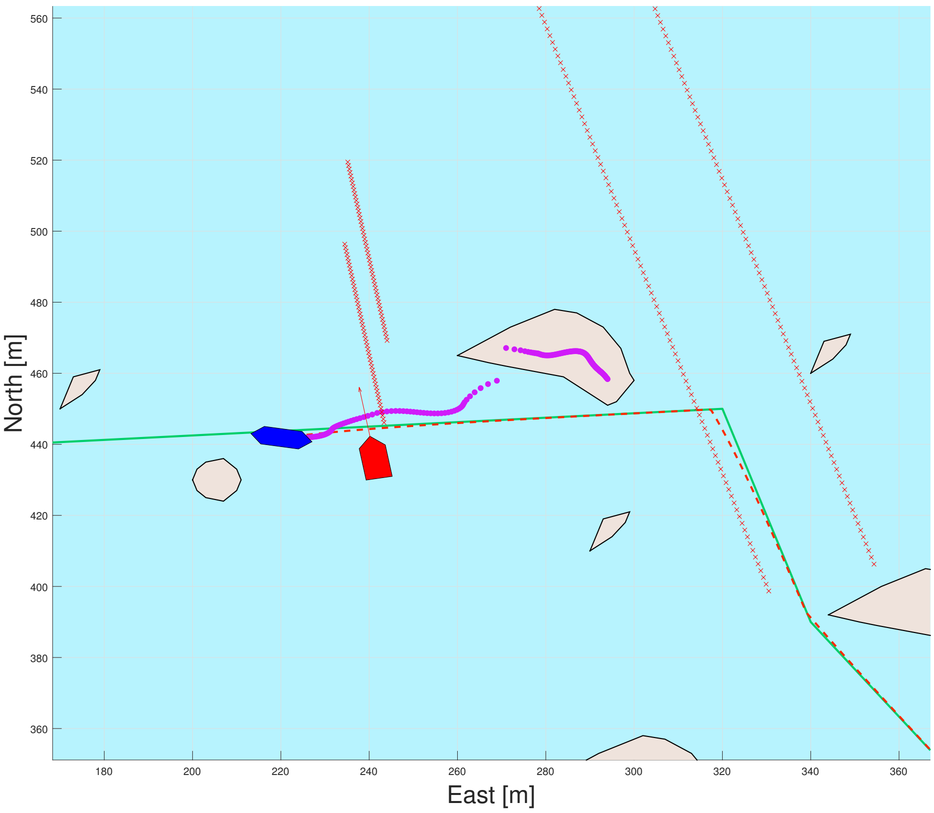
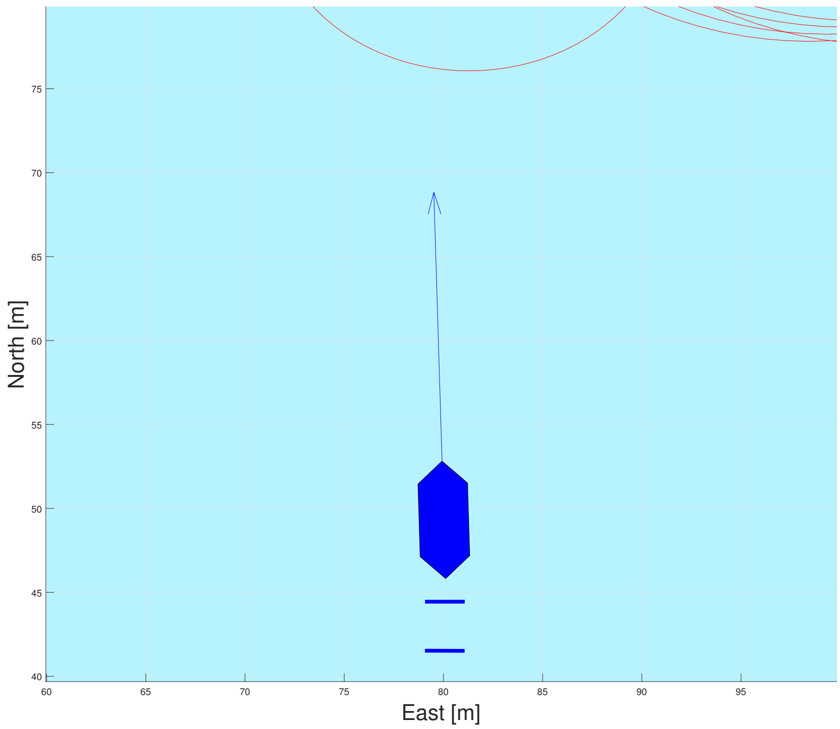
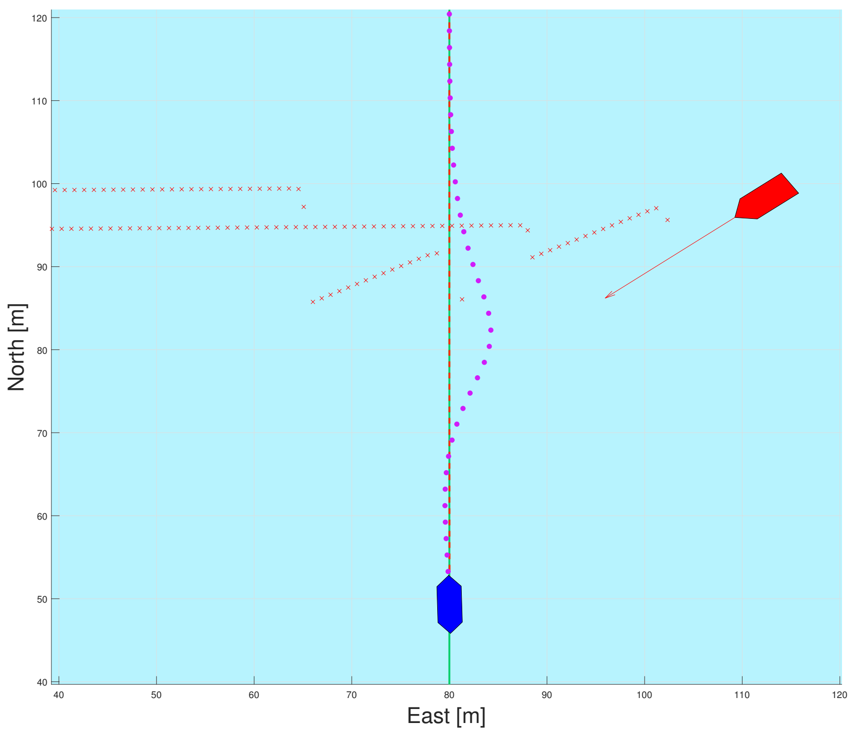


Figure 36: Stuck inside a static obstacle



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