Autonomous Escort with an Unmanned Surface Vessel Using Model Predictive Control

Axel Lagerstedt

Sean Small

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6310
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2026

Abstract

This thesis investigates the design and implementation of Model Predictive Control (MPC) and motion planning to enable an Unmanned Surface Vessel (USV) to perform autonomous escort missions. The objectives were to assess how the control architecture could utilize preexisting internal USV controllers and handle dynamic obstacles. Furthermore, how the controller architecture should be organized and how the controller performs when implemented on hardware were also explored. The proposed controller architecture consists of an MPC equipped with a generalized USV model to capture the unknown dynamics of the vessel where the model parameters are estimated online using an Extended Kalman Filter (EKF). The MPC is also equipped with a constraint which handles dynamic collision-avoidance, and a station keeping P/PI controller to ensure that the USV stays in the target position. The motion planner utilizes a custom A* search algorithm coupled with B-splines to plan long-term global paths to avoid obstacles and find the shortest route to the target position. Three different MPC variations, utilizing the motion planner to different degrees, were tested and validated through simulations and experiments on the Mini-Piraya USV platform by Saab Kockums AB. The results presented show that all three proposed controller variants are capable of steering the USV with precision, validating that the USV model paired with the EKF gives a good approximation of the system dynamics when operating at low velocities. The collision-avoidance implementation in the controller shows good performance and the USV is always kept at a safe distance when navigating around the escorted vessel. The experimental results also show that the controllers perform well when subjected to environmental disturbances, sensor noise, and lag proving that this more general USV model could easily be adapted and applied to a real marine vessel. Overall, the results from both the simulations and the experiments show that all three MPC implementations can be used to accomplish the escort mission, with each one bringing slightly differing advantages and limitations.

Acknowledgments

First and foremost, we would like to thank Saab Kockums AB and the Department of Automatic Control at LTH for giving us the opportunity to work with this thesis project. We would especially like to thank our supervisors at Saab Kockums, Erik Börjesson and Johan Silvander, for their continued support and keen interest in our thesis, for providing us with the resources we required, and for the many interesting discussion we have had. In addition, we would like to thank our colleagues in Karlskrona for welcoming us with open arms during all of our visits. We would also like to express our gratitude to our academic supervisor, Björn Olofsson, for his support, guidance, and valuable feedback during the entire thesis process. Lastly, a big thank you to our fellow Master Thesis students at the Lund office for the time we got to spend with them.

Contents

List of Abbreviations

USV	Unmanned Surface Vessel
MPC	Model Predictive Control
HVU	High Value Unit
EV	Escorting Vessel
SLOC	Sea Line of Communication
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
NED	North-East-Down
FRD	Forward-Right-Down
ENU	East-North-Up
RK4	Runge-Kutta 4
TMPC	Trajectory Tracking MPC
LMPC	Trajectory Tracking MPC Using Look-Ahead
PMPC	Time-Invariant Path Following MPC
ROS	Robot Operating System
VRX	Virtual RobotX

1

Introduction

Unmanned surface vessels (USVs) are autonomous or remote controlled naval vessels capable of navigating the surface of the ocean without crew onboard. These USVs can be used for a wide range of maritime missions such as environmental monitoring, oceanographic research, maritime logistics, and search and rescue operations. The common denominator for these use cases is that USVs can help minimize human risk while maintaining operational efficiency for missions in high-risk, remote, and logistically complex environments. USVs are, for these reasons, being used to a larger extent in government and military operations such as intelligence gathering, surveillance, maritime law enforcement, electronic warfare, and combat to increase presence in remote areas [Boretti, 2025]. There is, more specifically, a growing interest in using USVs for escorting missions to ensure that high value targets, such as cargo ships or military vessels, can travel through areas where a known threat is present. The USVs can then be used to repel threats and potentially intercept them without putting any human lives in danger [Liao et al., 2025]. While USVs help minimize human risk and reduce the cost of navigating the ocean, using them in military applications raises ethical concerns. Having a fully autonomous system capable of responding to a threat with lethal force without direct human control could potentially lead to unintended escalation and loss of human lives [Boretti, 2025]. The development and deployment of such systems therefore requires nuance and places a lot of responsibility on both developers and operators to ensure that USVs are used for ethical reasons.

Controlling a USV autonomously is, however, difficult since the ocean environment that it navigates through can vary immensely depending on the weather conditions, geographic location, and other vessels in its vicinity. This highly complex problem therefore requires advanced control systems which can take many aspects into account and steer the USV with precision [Liu et al., 2016]. This thesis will look more closely into the defence applications of USVs by investigating Model Predictive Control (MPC) design and implementation for autonomous escorting with a USV in two specific scenarios. In the first scenario, a surface vessel referred to as a high value unit (HVU) is escorted by an escorting vessel (EV) in a predetermined sea line of communication (SLOC) where the EV keeps a relative

position to the HVU. In the second scenario, an HVU is being escorted by an EV in an area where a third vessel, a potential threat, is present. The EV can then receive commands from the HVU to move to a position relative to the HVU and the potential threat. This position should be held, unless commanded otherwise, as a means to protect the HVU. In these scenarios a USV will be used as the EV while the HVU, SLOC, and threat are arbitrary.

1.1 Prior Work

MPC has previously been implemented and experimentally evaluated in [Kockum, 2022] on a Piraya USV, a vessel similar to the Mini-Piraya USV used for physical experiments in this thesis. This controller handled waypoint following and trajectory following at low speeds to steer the USV. The controller also handled obstacle avoidance to keep clear of shallow waters and docks to be able to safely perform autonomous docking of the USV. This MPC implementation minimized the Euclidean distance between the USV and the waypoint or trajectory to reach its target while ensuring that the USV kept a minimum distance to the obstacles. All of these features are of interest in this thesis with the difference that this thesis will focus on an MPC implementation for moving waypoints, trajectories and obstacles.

The performance of MPC on the Piraya was further explored and tested in [Svedberg, 2023] who developed and implemented a nonlinear path following MPC combined with system identification and reinforcement learning to improve the model-based controller performance and precision. The paths in this case were parameterizations of geometric curves in the output space and this MPC implementation then used a scalar path parameter as an optimization variable to choose the next reference point along the given path. This design allowed the controller to advance the system along the path at a variable pace and will be investigated if it can be applied to the use case in this thesis.

Both of the Master's Theses previously mentioned used offline identified model equations of the Piraya developed in [Ljungberg, 2021] for the MPC implementations. This is a three degrees of freedom (DOF) maneuvering model based on the ship model equations derived in [Fossen, 2021], which models the forces acting upon the ship. The control actuation for this model directly controls the thrust and angle of the motor to maneuver the USV, whereas this thesis will focus on an MPC design where the control signals are heading and velocity setpoints. For simulations, these works assumed that the known model equations were ideal and used them to calculate the movement of the USV. This thesis will, however, use a ship physics simulator to emulate the hydrodynamics of a USV with unknown dynamics to be able to investigate the more general MPC design.

Collision-avoidance for USVs using optimization-based methods was developed in [Wingqvist et al., 2024] and experimentally evaluated on the Piraya. This local planner could avoid static obstacles and other moving vessels by utilizing

model-based optimization to plan the USVs actions. The design and implementation of the dynamic collision-avoidance is of great interest in this thesis as the EV should avoid colliding with the HVU. For the optimization, a unicycle model was used to predict the movement of the USV where the control signals were heading and velocity setpoints similar to that of the use case in this thesis. In contrast to this thesis, which takes boat dynamics into consideration for simulations, this work simulated without dynamics meaning that the boats could turn instantly on the spot and keep a speed reference perfectly.

In [Cai et al., 2026] an Adaptive MPC escort controller was proposed for multi-USV escort systems. This paper shows, in simulations, how MPC can be used to escort and protect a master vessel with multiple USVs that can predict and intercept intruders. The AMPC escort scheme proposed in [Cai et al., 2026] is combined with a Kalman Filter enhanced Mixture Density Network to predict the movements of the intruders and an Improved Crested Porcupine Optimizer to effectively solve the non-convex and safety constrained optimization problem. Since this thesis focuses solely on protecting the HVU and not intercepting intruders, the main areas of interest from this paper are the USV model equations used in the optimization and how the formation keeping is achieved to ensure stable escorting.

1.2 Problem Formulation

This thesis is an investigation of how model-based controller design on a USV can be used to escort a moving surface vessel and react to a moving threat. It will explore how this can be implemented with Model Predictive Control and motion planning where collision-avoidance and preexisting internal controllers will be taken into account. The methods will be evaluated by performing tests virtually in simulations and field experiments on the Mini-Piraya USV. The purpose of this thesis is to further the development of escorting algorithms for autonomous USVs by investigating the following questions.

Research Questions

Q1 How can the MPC be designed and implemented to utilize existing controllers for velocity and heading?

Q2 How can the MPC be designed and implemented to handle dynamic spatial constraints?

Q3 How should the high-level controller architecture be organized? What are the functions of the motion planner and the MPC?

Q4 How does the controller perform implemented on real hardware, subjected to disturbances such as wind, waves, currents, and hardware noise?

1.3 Assumptions

For this thesis it is assumed that the EV has internal controllers which take forward velocity and heading as inputs, and steer the EV in the right direction while keeping a steady speed. The EV is also assumed to be an underactuated single hull vessel with one outboard motor located at the aft of the EV. The SLOC is assumed to be free from obstacles and that the only vessels present are the three previously mentioned. The positions of all three vessels are assumed to be known and given by GPS measurements and the EV is assumed to be equipped with an IMU for velocity and heading measurements.

1.4 Individual Contributions

Overall, both authors worked in close collaboration with each other during all parts of this thesis project. To streamline the initial phase of this project, the workload was divided into a few key areas of interest. Axel was primarily responsible for developing the A* search algorithm, the EKF implementation, and the code foundation for the MPC CasADi implementation. Sean was primarily responsible for developing the simulation environment in VRX, the internal controllers for the virtual USV and the real-time graphs displaying sensor measurements. Once a good foundation had been laid out, both authors worked together to develop, test, and evaluate different concepts in both simulations and field experiments to reach the final solution.

2

Background

This chapter presents the underlying theory for Model Predictive Control, A* search algorithms, and Kalman Filters used when developing the controller architecture. The USV platform and the hardware used in the field experiments are also presented.

2.1 Model Predictive Control

MPC is a form of optimal control which uses a dynamic model of the process to predict the future values of the controlled variables as a function of possible control actions. These control actions are obtained by solving a finite time horizon optimal control problem where the initial state is the current state of the process. With this, it is possible to optimize the process by choosing the best sequence of control actions at that time instant according to some criterion [Rawlings et al., 2017]. Once the optimization is complete, the first control signal in the sequence is applied to the process, and the MPC is executed again at the next sampling instant.

An example of a general discrete-time optimal control problem is

$$\begin{aligned}
\underset{\mathbf{x},\mathbf{u}}{\text{minimize}} \quad & J = \sum_{k=0}^{N-1} \ell(x_k, u_k) + V_f(x_N) && \text{(2.1a)}\\
\text{subject to} \quad & x(0) = x_0 && \text{(2.1b)}\\
& x_{k+1} = f(x_k, u_k) && \text{(2.1c)}\\
& (x_k, u_k) \in \mathbb{Z} && \text{(2.1d)}\\
& x_N \in \mathbb{X}_f && \text{(2.1e)}\\
& \text{for } k = 0, 1, \ldots, N-1 && \text{(2.1f)}
\end{aligned}$$

where J is the cost function which is minimized using $\mathbf{x} := (x_0, x_1, \ldots, x_N)$, the predicted state trajectories, and $\mathbf{u} := (u_0, u_1, \ldots, u_{N-1})$, the future control inputs [Rawlings et al., 2017]. The length of these trajectories is chosen by the prediction

horizon N which is a design variable that denotes how many steps into the future will be predicted.

The cost function (2.1a) is divided into two parts where $\ell(\cdot)$ is the stage cost and $V_f(\cdot)$ is the terminal cost. The stage cost is calculated each time step and summed up over the entire prediction horizon while the terminal cost is only calculated for the last step on the horizon. These costs can be designed in a number of ways, but in many cases are chosen as quadratic functions of the form $\ell(x,u) = x^T Qx + u^T Ru$. The weighting matrices Q and R serve as tuning parameters of the controller where, for example, larger values of Q relative to R penalizes the state variables more than the control actions resulting in larger costs for the state variables [Rawlings et al., 2017].

The constraints of the problem are described after the keywords "subject to" and restrict the search for the optimal solution. The initial condition (2.1b) ensures that the state $\mathbf{x}$ starts at x_0. Equation (2.1c) describes the system dynamics and is, in this case, a set of discrete-time differential equations. These equations ensure that the predicted state and control trajectories follow these dynamics for all time steps. The constraints (2.1d) shall guarantee that the state variables and control signals are contained in the set $\mathbb{Z}$ at each time step. Finally, the terminal state constraint (2.1e) requires the final state to be in a given set $\mathbb{X}_f$ [Rawlings et al., 2017].

The great advantage of MPC compared to other control techniques is that it is easy to include realistic constraints on the magnitude and rate of change of the state and control variables. The constraints in Equation (2.1d) could, for example, be inequalities of the form $|u(t)| \leq C_u$ and $|x(t)| \leq C_x$ determining the minimum and maximum values of the variables which are almost always present in applications [Glad and Ljung, 2000]. The model-based controller design is also suitable for multivariable systems and since MPC respects actuator limitations it allows the process to be run close to the constraints which is often cost-effective [Bernhardsson and Åström, 2016]. One challenge with MPC is the need for a model which describes the system well. Without a good model, the MPC predictions will be incorrect which will result in poor control.

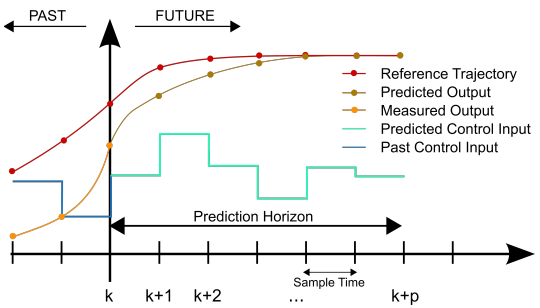


Figure 2.1 A basic working principle of a discrete-time MPC [Behrendt, 2009].

In Figure 2.1 is an example of a discrete-time MPC scheme. The reference trajectory here is predetermined for each time step. The MPC compares the reference with the measured state at time k and uses the model equations to predict the optimal control inputs required to reach the reference trajectory over the set prediction horizon.

2.2 Path Planning

Dijkstra's Algorithm

A method for finding the shortest path between nodes in a graph is to use Dijkstra's algorithm [Dijkstra, 1959]. It finds the optimal path from any node to another, but is computationally expensive with worst case performance $\mathcal{O}(|E|+|V|\log|V|)$, where V is the number of nodes and E is the number of edges, the connections between the nodes. It is a greedy algorithm as it always chooses what is locally best and it chooses to explore the unvisited node with the smallest distance from its current node. In each step, it performs a relaxation where it updates the distances of neighboring nodes if a shorter path is discovered using the just explored node. Since Dijkstra's algorithm uses no information about the goal's location, the search is performed radially around the source location. When it reaches the goal, the shortest path is constructed backwards from the goal to the source, this is enabled by each node keeping track of a parent which points to the shortest path to the node.

A* Search Algorithm

A* is based on Dijkstra's algorithm for finding the shortest path between nodes [Hart et al., 1968]. It improves upon Dijkstra's algorithm by using a heuristic, commonly Euclidean distance. Every node gets a cost associated with the heuristic and the shortest distance minimizes both the cost of moving and the cost of the heuristic which makes it explore around the heuristic first. The cost is defined as

$$f(n) = g(n) + h(n) \tag{2.2}$$

where $g(n)$ is the cost of the distance to the start node, $h(n)$ is the cost of the heuristic to the goal node, and n is the current node.

Although it has the same worst case performance as Dijkstra's algorithm, it achieves better average performance with a good heuristic. Compared to Dijkstra's algorithm which could be called an uninformed search since it needs no information about the goal, A* could be called an informed search since it uses the information about the target and the optimal route to improve the search.

B-spline

To make the calculated path from the A* search algorithm differentiable, it is approximated as a piecewise polynomial of order n by using a B-spline which is a

method to generate a smooth curve from points [Piegl and Tiller, 1997]. The curve is influenced by the points but does not necessarily pass through them. The reason to have a smooth curve is to avoid sharp turns and corners which ensures continuous forward and angular acceleration, which in turn prevents the acceleration state to go towards infinity that would occur at sharp turn.

2.3 Kalman Filter

Kalman filtering is a method of estimating states by estimating a joint probability distribution over the variables each step. The Discrete Kalman filter is summarized by the following algorithm [Simon, 2006]:

1. The dynamic system is given by the following equations:

$$x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1} \tag{2.3a}$$
$$y_k = H_k x_k + v_k \tag{2.3b}$$
$$E(w_k w_j^T) = Q_k \delta_{k-j} \tag{2.3c}$$
$$E(v_k v_j^T) = R_k \delta_{k-1} \tag{2.3d}$$
$$E(w_k v_j^T) = 0 \tag{2.3e}$$

2. The Kalman filter is initialized as follows:

$$\hat{x}_0^+ = E(x_0) \tag{2.4a}$$
$$P_0^+ = E\left((x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T\right) \tag{2.4b}$$

3. The Kalman filter is given by the following equations, which are computed for each time step $k = 1, 2, \ldots$

$$P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^T + Q_{k-1} \tag{2.5a}$$
$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{2.5b}$$
$$\hat{x}_k^- = F_{k-1}\hat{x}_{k-1}^+ + G_{k-1}u_{k-1} = \textit{a priori state estimate} \tag{2.5c}$$
$$\hat{x}_k^+ = x_k^- + K_k(y_k - H_k\hat{x}_k^-) = \textit{a posteriori state estimate} \tag{2.5d}$$
$$P_k^+ = (I - K_k H_k)P_k^-(I - K_k H_k)^T + K_k R_k K_k^T \tag{2.5e}$$

The equation for P_k^+, Equation (2.5e), is called the Joseph stabilized version of the covariance measurement update equation, which can be shown to be more stable and robust than the standard version. It is guaranteed to be symmetric positive definite, as long as P_k^- is symmetric positive definite. The algorithm follows the order given in the above equations where the control signals are required in a priori state estimate and measurements are required in a posteriori state estimate.

2.4 Extended Kalman Filter

To handle non-linear dynamics, the Extended Kalman Filter (EKF) uses linearized system dynamics in each time step. The Discrete Extended Kalman Filter is summarized by this algorithm [Simon, 2006]:

1. The system and measurement equations are given as follows:

$$x_k = f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1}) \tag{2.6a}$$
$$y_k = h_k(x_k, v_k) \tag{2.6b}$$
$$w_k \sim (0, Q_k) \tag{2.6c}$$
$$v_k \sim (0, R_k) \tag{2.6d}$$

   Note that the notation $w_k \sim (0, Q_k)$ means that w_k follows a Gaussian distribution with mean 0 and covariance Q_k.

2. Initialize the filter as follows:

$$\hat{x}_0^+ = E(x_0) \tag{2.7a}$$
$$P_0^+ = E\left((x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T\right) \tag{2.7b}$$

3. For $k = 1, 2, \ldots$, perform the following:

   a) Compute the following partial derivative matrices:

$$F_{k-1} = \left.\frac{\partial f_{k-1}}{\partial x}\right|_{\hat{x}_{k-1}^+} \tag{2.8a}$$
$$L_{k-1} = \left.\frac{\partial f_{k-1}}{\partial w}\right|_{\hat{x}_{k-1}^+} \tag{2.8b}$$

   b) Perform the time update of the state estimate and estimation-error covariance as follows:

$$P_k^- = F_{k-1} P_{k-1}^+ F_{k-1}^T + L_{k-1} Q_{k-1} L_{k-1}^T \tag{2.9a}$$
$$x_k^- = f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, 0) \tag{2.9b}$$

   c) Compute the following partial derivative matrices:

$$H_k = \left.\frac{\partial h_k}{\partial x}\right|_{\hat{x}_k^-} \tag{2.10a}$$
$$M_k = \left.\frac{\partial h_k}{\partial v}\right|_{\hat{x}_k^-} \tag{2.10b}$$

d) Perform the measurement update of the state estimate and estimation-error covariance as follows:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{2.11a}$$

$$\hat{x}_k^+ = x_k^- + K_k(y_k - H_k \hat{x}_k^-) \tag{2.11b}$$

$$P_k^+ = (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T \tag{2.11c}$$

2.5 USV Platform

The USV used as the EV for the field experiments was the Mini-Piraya USV shown in Figure 2.2. The Mini-Piraya is a small USV with a single propeller on a fixed axle driven by an electric motor. To steer the USV, a pair of rudders driven by servos are placed on either side of the propeller making the USV behave similarly to a vessel with a single aft outboard motor. This USV is equipped with a GPS and a Pixhawk controller running the Ardupilot software which could take velocity and heading setpoints and send control commands to the motor and servos. With the GPS and the Pixhawk's built in IMU, the Mini-Piraya provided measurements for position, heading, forward velocity, sideways velocity, and angular velocity. These measurements were estimated using a built in Kalman Filter which processed the raw sensor data to give more accurate measurements. The USV is also equipped with a Raspberry Pi used to run the remaining software.

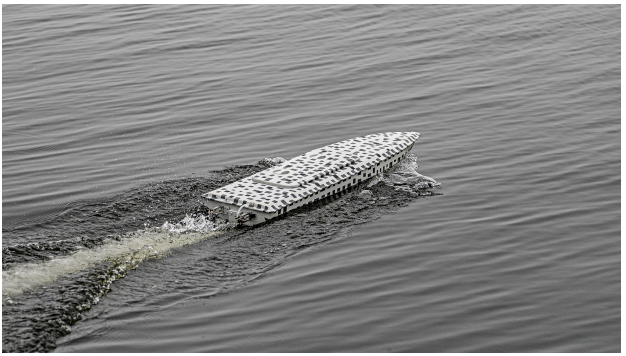


Figure 2.2 Mini-Piraya USV used for field experiments.

3

Modeling

This chapter presents and defines the coordinate frames and DOF used to model the EV. The model equations are then established and the process behind how they were developed is presented. Since USVs can vary from one vessel to another, a more general MPC model which utilizes preexisting USV controllers for velocity and heading regulation will be investigated. Lastly, the chosen discretization method and model parameter estimation is described.

3.1 Coordinate Frames

Three primary coordinate frames are used in this thesis to describe the equations of motion of the marine vessels, as shown in Figure 3.1. The first frame is the global inertial frame which has its origin fixed to earth at a defined position. Here, the x_n-axis points north, the y_n-axis points east, and the z_n-axis points down, and this can be referred to as a north-east-down (NED) system. The second frame is the body-fixed EV frame which has its origin located at the center of mass of the EV. Here, the x_b-axis points towards the front of the vessel, the y_b-axis points starboard (right), and the z_b-axis points down, and this can be referred to as a a forward-right-down (FRD) system. The third frame is the body-fixed HVU frame which is also an FRD system with its origin located at the center of mass of the HVU. The six DOF for the body-fixed frames are described in Table 3.1.

Table 3.1 The six DOF in the body-fixed frames.

Axis and Tait-Bryan angle	Velocity	Motion
x_b	u	surge
y_b	v	sway
z_b	w	heave
ϕ	p	roll
θ	q	pitch
ψ	r	yaw

However, since the vessels in this thesis are surface vessels the DOF can be reduced to three: surge, sway, and yaw. This assumes that heave, roll, and pitch are negligible which is most likely the case when navigating calm waters at low speeds since these three DOF are mostly excited by waves.

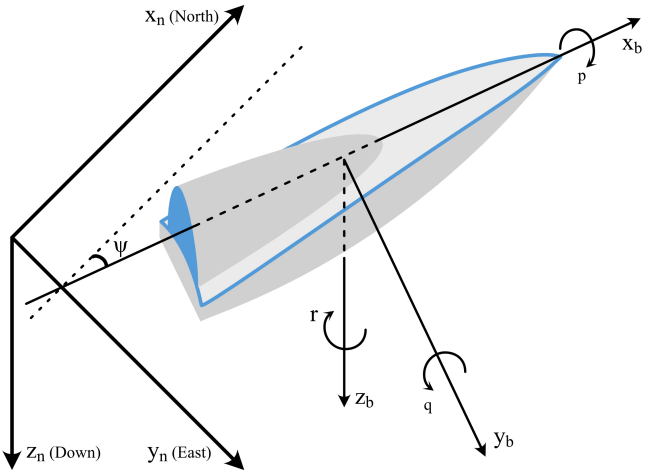


Figure 3.1 The body-fixed frame of one vessel in relation to the global inertial frame [Kockum, 2022].

3.2 Unicycle Model

To steer the EV, since it has onboard controllers for velocity and heading, setpoints for velocity and heading are sent via an interface. Due to this interface, the standard Fossen model for marine craft does not work since it depends on controlling the thrust and angle of the motor [Fossen, 2021]. Furthermore, a model with fewer parameters was more desirable for potentially testing on different USVs. This is why the dynamics were approximated to be unicycle kinematics, where rotation is fully decoupled from surge. The equations describing a unicycle model are as follows

$$\dot{x} = u \cdot \cos(\psi) \tag{3.1a}$$
$$\dot{y} = u \cdot \sin(\psi) \tag{3.1b}$$

First-Order Systems Extension

This model in Equation (3.1) has the drawback of not modeling forward acceleration, $\dot{u}$, and yaw rate, $\dot{\psi}$, meaning that it has issues with infinite acceleration and yaw

rate between the prediction steps in the MPC. To solve this issue, first-order systems for forward acceleration and yaw rate were implemented. This local approximation and simplification of the vessel dynamics could be used since the USV is assumed to be limited to operating at low velocities where the quadratic dampening of water has less of an impact. With the control signals ψ_{ref} and u_{ref} the extended unicycle model is

$$\dot{x} = u \cdot \cos(\psi) \tag{3.2a}$$
$$\dot{y} = u \cdot \sin(\psi) \tag{3.2b}$$
$$\dot{\psi} = \frac{1}{T_\psi}(\psi_{ref} - \psi) \tag{3.2c}$$
$$\dot{u} = \frac{1}{T_u}(u_{ref} - u) \tag{3.2d}$$

where T_ψ and T_u are time constants for each of the first-order systems. This was further extended to include a first order-system for yaw acceleration

$$\dot{x} = u \cdot \cos(\psi) \tag{3.3a}$$
$$\dot{y} = u \cdot \sin(\psi) \tag{3.3b}$$
$$\dot{\psi} = r \tag{3.3c}$$
$$\dot{u} = \frac{1}{T_u}(u_{ref} - u) \tag{3.3d}$$
$$\dot{r} = \frac{1}{T_r}(k_r(\psi_{ref} - \psi) - r) \tag{3.3e}$$

where ψ now depends on r, and the time constant T_r and gain k_r replace T_ψ.

Sway Model Extension

The unicycle model is a standard non-holonomic robotics kinematic model, but the boat does not move in a kinematic way. A boat has inertia, which means that it wants to continue in the direction it is going. A way to model this was to introduce dynamics for sway, sideways velocity, from Fossen's model as

$$\dot{x} = u \cdot \cos(\psi) - v \cdot \sin(\psi) \tag{3.4a}$$
$$\dot{y} = u \cdot \sin(\psi) + v \cdot \cos(\psi) \tag{3.4b}$$
$$\dot{\psi} = r \tag{3.4c}$$
$$\dot{u} = \frac{1}{T_u}(u_{ref} - u) \tag{3.4d}$$
$$\dot{v} = -d_1 \cdot v - d_2 \cdot v|v| - C \cdot u \cdot r \tag{3.4e}$$
$$\dot{r} = \frac{1}{T_r}(k_r(\psi_{ref} - \psi) - r) \tag{3.4f}$$

where d_1 is a linear dampening coefficient, d_2 is a quadratic dampening coefficient, and C is a Coriolis coupling coefficient [Fossen, 2021].

3.3 Discretization Using Runge-Kutta 4

The USV model equations are given in continuous time since it is natural to express dynamic relationships as differential equations. However, the MPC in this report was implemented in discrete-time. Therefore, to use the USV model effectively the system given in Equation (3.4) was discretized using Runge-Kutta 4 (RK4) which is defined as

$$w_{i+1} = w_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{3.5}$$

where

$$k_1 = f(t_i, w_i) \tag{3.6a}$$
$$k_2 = f\left(t_i + \frac{h}{2}, w_i + \frac{1}{2}k_1\right) \tag{3.6b}$$
$$k_3 = f\left(t_i + \frac{h}{2}, w_i + \frac{1}{2}k_2\right) \tag{3.6c}$$
$$k_4 = f(t_i + h, w_i + k_3) \tag{3.6d}$$

This method has a local truncation error of $O(h^4)$ which is better than simpler methods such as implicit and explicit Euler, but it is more computationally expensive since it evaluates four times the points [Sauer, 2012].

3.4 Parameter Estimation Using EKF

Since the model in Equation (3.4) contains the USV-dependent parameters $(d_1, d_2, T_u, T_r, k_r, C)$, these parameters needed to be estimated for the model to

be usable. This was done by treating the parameters as virtual states in an EKF which was implemented as described in Section 2.4. The state and input vectors became

$$\mathbf{x} = (x, y, \psi, u, v, r, d_1, d_2, T_u, T_r, k_r, C)^T \text{ and } \mathbf{u} = (\psi_{ref}, u_{ref})^T \tag{3.7}$$

and the model in Equation (3.4) was extended with

$$\dot{d}_1 = 0, \quad \dot{d}_2 = 0, \quad \dot{T}_u = 0, \quad \dot{T}_r = 0, \quad \dot{k}_r = 0, \quad \dot{C} = 0 \tag{3.8}$$

to model these parameters as constants. This new set of differential equations was also discretized using RK4. Artificial process noise is added to the parameters which enables the EKF to change them slowly.

4

Motion Planning & Model-Based Predictive Control

This chapter presents the methods used to develop the Motion Planner and the MPCs, and how they vary. The collision-avoidance constraint used in the MPCs to avoid the HVU is defined, and the target station keeping controllers are also presented. Lastly, the controller architecture is described which highlights what the different parts are intended for.

4.1 Motion Planning

To recall from Section¨ 2.2, the cost of an A* search is defined as

$$f(n) = g(n) + h(n) \tag{4.1}$$

where $g(n)$ is the cost based on distance from the start node and $h(n)$ is the heuristic based cost to the goal. A* is optimal when the heuristic is admissible, i.e.

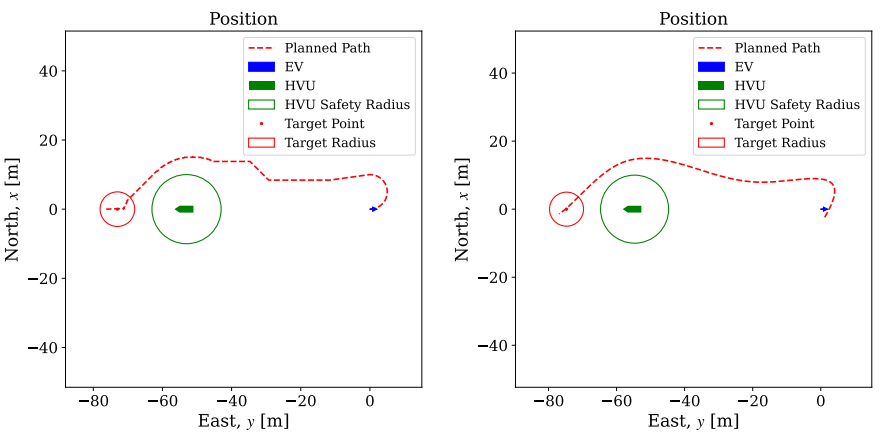
$$h(n) \leq h^*(n) \tag{4.2}$$

where $h^*(n)$ is the cost of the shortest path. In the case of this thesis, 2D space was discretized into a grid meaning that using the Euclidean distance is always admissible since it is the shortest distance between two points, which entails that A* will always give the optimal path. The search area was defined as a square with side length l with the HVU at its center and was discretized into a uniform grid of nodes with a distance h between each node in Manhattan distance. The search begins at the source, the EV's starting position inside the grid, and ends at the destination, the target position, which was assumed to always be located inside the search area. The A* search is allowed to move up, down, left, right, or diagonally in each direction

when moving from a node, where the diagonal distance is defined as $h\sqrt{2}$. To ensure a collision-free path, an area around the HVU with radius $r_{HVU} + r_{target}$ was marked as an obstacle and as not allowed.

Before starting the search, if the EV's heading was pointing away from the target point, a turn with radius r_{EV} aws added to the beginning of the path. This was added to allow for a smoother turn since the EV cannot turn around on the spot. If the EV was outside the search area, the optimal route was simplified to the Euclidean norm between the target and the EV. This route was also added to the beginning of the path and the point where the Euclidean norm intersected the search area was where the A* search proceeded from. Once inside the search area, the A* search was performed and the shortest path from the source to the destination was computed.

This motion planning was performed with a period of Δt to ensure that the path was up to date. Since the movement of the HVU was not taken into account during the planning, this update period needed to be short enough so that the path was always outside of the HVU radius, but longer than the MPC's update period. To aid the MPC reach the target point more effectively, the target's future movement was accounted for by predicting its trajectory over the period Δt and added to the end of the path. An example of a planned path is shown in Figure 4.1 where all parts of the motion planning are present.



(a) Planned path without a B-spline

(b) Planned path with a B-spline

Figure 4.1 Example planned path defined by points with and without a B-spline. The EV is shown as the blue arrow, the HVU as the green polygon, and the target as the red dot. The area in which the EV should avoid is displayed as the green circle, and the area in which the EV is aiming for is displayed as the red circle. The planned path is the red dotted line.

4.2 Trajectory Tracking MPC for Following a Moving Target

For the first approach, a single moving target position relative to the HVU is given with (x,y)-coordinates, velocity, and heading. Using the final discretized unicycle model with first-order systems and sway, the resulting state vector and input vector is

$$\mathbf{x} = (x, y, \psi, u, v, r)^T \text{ and } \mathbf{u} = (\psi_{ref}, u_{ref})^T \tag{4.3}$$

The target trajectory was constructed using the velocity and heading of the HVU, which was assumed to be constant over the prediction horizon. The trajectory was constructed as

$$(x_{k+1}, y_{k+1}) = (x_k + u_{HVU} \cdot \cos(\psi_{HVU}), y_k + u_{HVU} \cdot \sin(\psi_{HVU})) \tag{4.4}$$

Using this trajectory, state, and input vector this MPC design, also referred to as the TMPC, was formulated as

$$\begin{aligned}
\underset{\mathbf{x}_0,\ldots,\mathbf{x}_N,\mathbf{u}_0,\ldots,\mathbf{u}_{N-1}}{\text{minimize}} \quad & J = \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) && \text{(4.5a)}\\
\text{subject to} \quad & \mathbf{x}(0) = \mathbf{x}_0 && \text{(4.5b)}\\
& \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) && \text{(4.5c)}\\
& \mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max} && \text{(4.5d)}\\
& \mathbf{x}_{\min} \leq \mathbf{x}_k \leq \mathbf{x}_{\max} && \text{(4.5e)}\\
& |\mathbf{u}_k - \mathbf{u}_{k-1}| \leq \Delta\mathbf{u}_{\max} && \text{(4.5f)}\\
& \text{for } k = 0, 1, \ldots, N-1 && \text{(4.5g)}
\end{aligned}$$

where

$$\begin{aligned}
\ell(\mathbf{x}_k, \mathbf{u}_k) = {} & Q_x(x_{ref,k} - x_k)^2 + Q_y(y_{ref,k} - y_k)^2 \\
& + Q_\psi(1 - \cos(\psi_{ref,k} - \psi_k)) + Q_u(u_{ref,k} - u_k)^2 \\
& + Q_v \cdot v_k^2 + Q_r \cdot r_k^2
\end{aligned} \tag{4.6}$$

with Q as

$$Q = \text{diag}(Q_x, Q_y, Q_\psi, Q_u, Q_v, Q_r) \tag{4.7}$$

Here, the reference $\mathbf{x}_{ref,k} = (x_{ref,k}, y_{ref,k}, \psi_{ref,k}, u_{ref,k})$ is dependent on k since it is a trajectory with different references for each step in the horizon. For this MPC design, the cost function penalizes the Euclidean distance between the EV and the target trajectory meaning that it strives to minimize this distance for each step in

the horizon. The cost function also penalizes heading error where $\psi_{ref,k}$ is the target heading to ensure that the EV is moving in the same direction as the target. Surge error is also penalized where $u_{ref,k} = u_{max}$ to encourage the MPC to use the maximum allowed surge. Lastly, the cost function also penalizes sway and yaw rate differing from zero so that the EV only turns when necessary helping to reduce oscillatory behavior and promote slower turns.

Note also that J is a normal quadratic cost function with the exception that the cost for heading error is $Q_\psi(1-\cos(\psi_{ref}-\psi))$ which is a way to handle the periodicity of angles, i.e. that an angle $\alpha \equiv \alpha + 2\pi k$, $k \in \mathbb{Z}$. The issue, in this case, with a normal quadratic error is that it is minimized by going in the direction that makes it smaller. So, for example, if the EV's current angle is $-\frac{3\pi}{4}$ and the goal is $\frac{3\pi}{4}$ then the MPC would take the long way around because that minimizes the error instead of making the shorter turn by turning negatively and crossing the discontinuity $-\pi/\pi$.

Based on empirical testing, the output weight matrix is defined as

$$Q = \text{diag}(1, 1, 0.5, 0.5, 0.01, 0.01) \tag{4.8}$$

for all three MPCs meaning that the MPCs primarily prioritize the position error, then the heading and surge errors, and lastly the sway and yaw rate errors.

4.3 Trajectory Tracking MPC Using Look-Ahead for Path Following

The second approach is based on the TMPC formulation in Section 4.2 with the main difference being that this MPC design, also referred to as the LMPC, utilizes the planned path to reach the target position. Instead of choosing the final position to construct a trajectory, this MPC design finds the closest position along the planned path and moves it up along the path with some predetermined distance l. The LMPC then uses this position to create the trajectory in the same way as before, relative to the HVU's surge and heading. The trajectory of this new waypoint is used as the position reference $(x_{ref,k}, y_{ref,k})$ in the cost function J. If the distance between the EV and the target position is ever less than the distance l, then the target position becomes the next waypoint and the MPC uses the target trajectory as a reference instead. This way, the MPC utilizes the long-term planning of the motion planner.

4.4 Time-Invariant Path Following MPC With Look-Ahead

For the third approach, still using the final unicycle model, we have the same state vector as before with an extension of a third control variable θ which the planned path reference is parametrized in

$$\mathbf{x} = (x, y, \psi, u, v, r)^T \text{ and } \mathbf{u} = (\psi_{ref}, u_{ref}, \theta)^T \tag{4.9}$$

The idea is to let this MPC design, also referred to as the PMPC, minimize the error by not only choosing ψ_{ref} and u_{ref} but also which path references $\mathbf{x}_{ref,k}$ to minimize against [Faulwasser and Findeisen, 2016]. The optimization problem is then formulated as

$$\begin{aligned}
\underset{\mathbf{x}_0,\ldots,\mathbf{x}_N,\mathbf{u}_0,\ldots,\mathbf{u}_{N-1}}{\text{minimize}} \quad & J = \sum_{k=0}^{N-1} \ell(\mathbf{x}_k,\mathbf{u}_k) + V_f(\mathbf{x}_N) && \text{(4.10a)}\\
\text{subject to} \quad & \mathbf{x}(0) = \mathbf{x}_0 && \text{(4.10b)}\\
& \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k,\mathbf{u}_k) && \text{(4.10c)}\\
& \mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max} && \text{(4.10d)}\\
& \mathbf{x}_{\min} \leq \mathbf{x}_k \leq \mathbf{x}_{\max} && \text{(4.10e)}\\
& |\mathbf{u}_k - \mathbf{u}_{k-1}| \leq \Delta\mathbf{u}_{\max} && \text{(4.10f)}\\
& \theta_k > \theta_{k-1} && \text{(4.10g)}\\
& \text{for } k = 0,1,\ldots,N-1 && \text{(4.10h)}
\end{aligned}$$

This design differs from the others by having a terminal cost to help with convergence, and the forward constraint in Equation (4.10g). This constraint is used to guarantee forward movement in the prediction horizon by ensuring that the MPC always chooses a larger θ, i.e. a position further along the path. The first θ is calculated using the Euclidean norm and then moved some points further up the path to get some look-ahead to help prevent overshoot. The stage cost is defined as

$$\begin{aligned}
\ell(\mathbf{x}_k,\mathbf{u}_k) = {} & Q_x(x_{ref,k} - x_k)^2 + Q_y(y_{ref,k} - y_k)^2 \\
& + Q_\psi(1 - \cos(\psi_{ref,k} - \psi_k)) + Q_u(u_{ref,k} - u_k)^2 \\
& + Q_v \cdot v_k^2 + Q_r \cdot r_k^2
\end{aligned} \tag{4.11}$$

and the terminal cost is defined as

$$\begin{aligned}
V_f(x_N) = Q_f \Big(& Q_x(x_{ref,N} - x_N)^2 + Q_y(y_{ref,N} - y_N)^2 \\
& + Q_\psi(1 - \cos(\psi_{ref,N} - \psi_N)) \Big)
\end{aligned} \tag{4.12}$$

The stage cost in (4.11) is very similar to the previous definition in (4.6) with the main difference being that the reference vector $\mathbf{x}_{ref,k}$ is chosen using the parametrized path and the variable θ. The terminal weight $Q_f = 10$, which penalizes the MPC even more for being far away from the target position and differing from the reference heading in the final step of the horizon.

4.5 Collision-Avoidance Constraint

To ensure that the MPCs keep the EV from colliding with the HVU, a constraint was added to each MPC formulation. This constraint is formulated as

$$0 \leq \frac{(x_k - x_{HVU,k})^2 + (y_k - y_{HVU,k})^2}{r_{HVU}^2} - 1 + s_k \tag{4.13}$$

where s_k is a slack variable. Since the HVU is moving, its trajectory is predicted and used in this constraint which is why both x_{HVU} and y_{HVU} are dependent on k. To simplify these calculations, the distance between the EV and the HVU is normalized by dividing with r_{HVU}^2. So, if the EV is too close to the HVU, the sum will become negative and the MPC is forced to use the slack variable to fulfill the constraint. The slack variable s_k is then multiplied with a weight coefficient k_s and added to the cost function J in each step of the horizon. The coefficient k_s is in this case 10^6 so that the cost of being inside the HVU radius is very large. In turn, if the EV is outside of the HVU area then the constraint is fulfilled and no cost is added. Without the slack variable, if the EV entered the HVU area then all solutions found by the MPC would be infeasible resulting in undesirable behavior. This way, the MPC is penalized heavily if it enters the HVU radius, but can still get an optimal solution and steer the EV away from the HVU.

4.6 Switch to P/PI Controller for Target Station Keeping

Once the EV enters the target area, the controller switches from the MPC solver to an integrated P controller for heading and PI controller for velocity which are tuned to keep the EV close to the target position. This is done by regulating the position error of the EV in the target's body-fixed coordinate system where the target is in the origin. The error in the target's x-direction is coupled to u_{ref}, so if the EV is behind the target then u_{ref} will increase and vice versa if the EV is in front of the target. The error in the target's y-direction is then coupled to ψ_{ref}, so if the EV is to the right of the target then ψ_{ref} will decrease and vice versa if the EV is to the left of the target. Since the target is moving with the HVU with a certain heading and velocity, this body-fixed coordinate system is, in turn, also moving. The HVU velocity and heading are therefore used as feedforward signals so that when the position error is zero and the EV has stabilized on the target point, then the EV control signals will ideally be the same as u_{HVU} and ψ_{HVU}. An integral part was added to the velocity controller because of the large dampening effect that water has when traveling at different velocities.

4.7 Controller Architecture

The controller architecture was divided into two main parts, the Motion Planner and the MPC. The idea with the Motion Planner was to have an implementation capable of planning long-term movement at a low frequency while the MPC was capable of planning more short-term movement and adjustments at a higher frequency. The Motion Planner was only run for the MPC variants that utilized the path where the planned path was sent to the MPC as soon as it was updated. Both of these implementations required processed state information about the EV, HVU and target where $\mathbf{x}_{EV} = (x, y, \psi, u, v, r)^T$, $\mathbf{x}_{HVU} = (x, y, \psi, u)^T$, and $\mathbf{x}_{target} = (x, y)^T$. All of these state measurements were then sent to the controller at a higher frequency than both the MPC and Motion Planner were run at, to ensure that the measurements were up to date. Using this information the MPC then solved the optimization problem, and the control signals u_{ref} and ψ_{ref} were sent to the EV over the interface. The P/PI controllers for target station keeping were integrated into the MPC and once the EV had reached the target area the controller would switch to these P/PI controllers to produce the control signals. While in the target area the Motion Planner would also be turned off to minimize unnecessary computations.

5

Implementation

This chapter will cover the implementation of the controller as well as the software structure and the communication between different parts of it. The real-time User Interface (UI) used to test and evaluate runs, and the symbolic framework used to build the optimization problems are also presented.

5.1 Software Structure

Robot Operating System (ROS) is an open-source middleware for robotics applications used to handle the communication between different components in a system [Macenski et al., 2022]. A system can be organized by breaking down its functionality into different nodes, where each node communicates via topics using a publish and subscribe mechanism. In this case ROS2 Jazzy is used. A chart of the topics and nodes used for this implementation can be found in Figure 5.1.

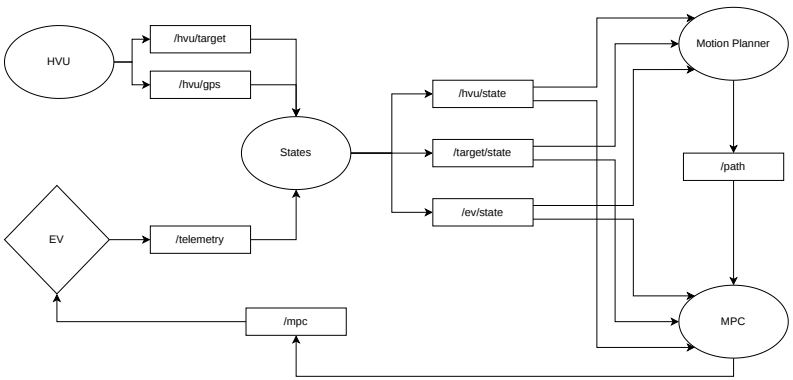


Figure 5.1 Chart of nodes and topics. Nodes are elliptical and topics are rectangular. The EV is different since it is an interface that is used rather than a node.

The nodes in Figure 5.1 each have subsystems that run inside them:

- **HVU:** Represents a simulated boat modeled as a point moving at a constant speed with a constant heading. Initializes with set values relative to the origin which are updated based on the speed and heading of the boat.

- **States:** Receives raw data from the EV and the HVU, and processes this data to be in the global inertial coordinate system and correct format for the MPC and Motion Planner. Initializes the origin of the inertial coordinate system in the EV's initial position.

- **Motion Planner:** Runs the A* search and B-spline creating the path used in the MPC.

- **MPC:** Runs the MPC and solves the optimization problem. Also runs an EKF as described in Section 3.4.

To support during simulations and experiments, a UI displaying real-time sensor measurements was created using Matplotlib in a separate ROS2 node [Hunter, 2007]. This UI subscribed to $/path$, $/mpc$, and all the state topics, and displayed these measurements in three different graphs. The first graph displayed the current position of the EV, HVU, and target point, the past trajectories of the EV and HVU, the predicted trajectory of the EV, the planned path, and the radius of both the HVU and target point. The second graph displayed the surge measurements of the EV and HVU, the surge reference, and the EVs predicted surge trajectory. The third graph displayed the yaw measurement of the EV, the yaw reference, and the EVs predicted yaw trajectory. It should also be noted that there is a topic called $/new\_target$ not shown in the chart which is used to manually update the target position in the HVU node.

5.2 CasADi

The open-source framework CasADi is used to formulate and solve the nonlinear optimization problem [Andersson et al., 2019]. It provides tools for symbolic modeling and gradient-based numerical optimization, which makes it well suited for optimal control problems.

MPC Formulation in CasADi

The optimization problem is formulated using `nlpsol` which is CasADi's primary tool for solving nonlinear programming problems. `nlpsol` generally solves problems on the form

$$\underset{x}{\text{minimize}} \quad F(x,p) \tag{5.1a}$$
$$\text{subject to} \quad x_{\min} \leq x \leq x_{\max} \tag{5.1b}$$
$$g_{\min} \leq G(x,p) \leq g_{\max} \tag{5.1c}$$
$$p = P \tag{5.1d}$$
$$\tag{5.1e}$$

where p are parameters.

Symbolic MX variables were created for each state and control signal u, and also for some model parameters. For the PMPC, two `interpolants` were also created to keep track of the paths parametrized in θ. The `interpolants` needed to be initialized for each path meaning that the PMPC needed to be recompiled each time the path was updated.

The problem is formulated using direct multiple shooting, which solves each time step separately. The model function $f(x,u)$ and the stage cost function $\ell(\cdot)$ was defined in continuous-time and a symbolic RK4 was used to discretize these equations symbolically. The decision variables $(\mathbf{x}_k, \mathbf{u}_k)$ were added to a decision vector $\mathbf{w}$ and the constraints for each state was added to a lower-bound vector and an upper-bound vector $[\mathbf{l}_{b,w}, \mathbf{u}_{b,w}]$ where each position in the vectors correspond to each state. The same was done for the initial guesses in a vector $\mathbf{w}_0$. A constraint vector $\mathbf{g}$ with corresponding vectors for initial guesses $\mathbf{g}_0$ and lower and upper bounds $[\mathbf{l}_{b,g}, \mathbf{u}_{b,g}]$ was also built to handle constraints other than bounds on states and control signals. An equality constraint defined as

$$\Phi(t_k, t_{k+1}, \mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}_{k+1} + s_k = 0 \tag{5.2}$$

where

$$\Phi(t_k, t_{k+1}, \mathbf{x}_k, \mathbf{u}_k) \tag{5.3}$$

is a numerical integration using RK4 in the interval $[t_k, t_{k+1}]$ was implemented to make sure that each state starts where the last one ends. s_k is a slack variable that was used to improve computation times and ensure that the MPC would not result in any infeasible solutions. If done properly, different slack variables should have been added to each constraint, but it was reasoned that they would not be used at the same time. The same slack variable could therefore be added to all constraints which requires fewer decision variables. The cost of the slack variable is many magnitudes higher than the other costs.

The collision-avoidance constraint was implemented as described in Section 4.5 and a soft constraint on surge defined as

$$0 \leq u + s_k \tag{5.4}$$

was implemented to ensure that the MPC can find a solution even if u is measured to be negative. Instead of using costs on $\Delta\mathbf{u}$, it was found that having a constraint on it gave better performance in this case, and this constraint was implemented as

$$\Delta\mathbf{u}_{min} \leq \Delta\mathbf{u} + s_k \leq \Delta\mathbf{u}_{max} \tag{5.5}$$

For the Path Following MPC, the constraint on θ was implemented as

$$\frac{1}{5000} \leq \theta_k - \theta_{k-1} + s_k \tag{5.6}$$

to ensure that the MPC always strives to choose a larger θ in each step. All of these constraints are added to $\mathbf{g}$.

To improve computational performance, a warm-start stategy is used, where the solution from the previous time step is used as the inital guess for the next opimization. This significantly reduces computional time after the first iteration. To further increase performance, just in time compilation is used which makes the first solution even slower since it has to first compile all the functions before it can start executing.

6

Simulations

This chapter presents the simulation environment used for testing and describes how it was modified to fit this thesis. The setup used during most simulations is then outlined and examples of how the different MPCs behave are shown. Finally, the simulated results displaying how the MPCs take on the two scenarios of the escort mission are shown.

6.1 Simulation Environment

Virtual RobotX (VRX) is the virtual environment used for simulations and is built on Gazebo which supports simulating the hydrodynamics and movement of USVs in marine environments [Bingham et al., 2019]. The boat, WAM-V, given in VRX is used as the virtual USV with modifications to the engine layout giving it a single aft outboard motor. Communication to the simulated USV is done via two ROS topics, $/wamv/thrusters/middle/thrust$ and $/wamv/thrusters/middle/angle$, which control the thrust and the angle of the motor. To simulate the velocity and heading controllers assumed to exist on the USV, PID controllers, tuned to have decent performance, were implemented in a separate ROS node as a middle stage between the MPC and the simulated boat. The MPC is then able to send velocity and heading setpoints to the PIDs which in turn send thrust and angle commands over the ROS topics. Sensor measurements from VRX are sent via the topics $/wamv/sensors/gps/gps/fix$ and $/wamv/sensors/imu/imu/data$, which give measurements for the GPS position, yaw, yaw rate, and accelerations along each axis. Surge and sway measurements are, however, not given so they were approximated using a combination of the position change and the acceleration both with regards to time. The VRX sensor measurements also uses the data convention east-north-up (ENU) meaning that these had to be converted to NED before being used.

6.2 Simulation Goals

Looking back at the two scenarios for the escort mission which the controllers are built for, there are a couple of criteria which they should achieve. Firstly, the controllers should be capable of moving the EV from any initial position to the target position without colliding with the HVU. Secondly, once the EV has reached the target position, the controllers should be capable of keeping the EV inside the target area for a longer period of time. Thirdly, the controllers should be capable of receiving a new target position and moving the EV from its current position to that new target. All of this should be achieved with the HVU constantly on the move.

6.3 Simulated Behavior Examples

The simulations show the different behaviors and goals of each MPC to visualize how each MPC predicts movement and what it minimizes towards. The predicted trajectories and planned paths will then not be shown in the simulation and experiment results to minimize clutter. The EV starts in the same position for each test and the HVU is initialized 0 meters north and -40 meters east relative to the EV with a heading of $-\frac{\pi}{2}$ rad and a velocity of 1.0 m/s.

Trajectory Tracking MPC for Following a Moving Target

The predicted trajectory for the TMPC is almost straight towards the target position, which is in line with the cost function for this MPC which penalizes Euclidean distance to the target trajectory shown as the red dotted line in Figure 6.1. Looking at the predicted EV trajectory in yellow, it seems as if the MPC wants to enter the HVU radius, violating the collision-avoidance constraint. However, since the MPC knows that the HVU is moving, it predicts that the HVU will have moved away from that spot once the EV gets there.

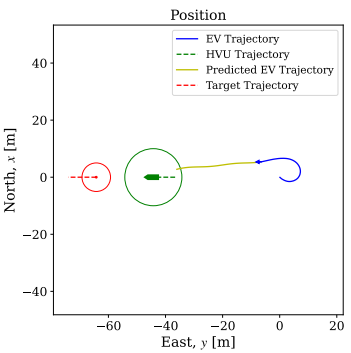


Figure 6.1 Example behavior of the TMPC.

Trajectory Tracking MPC Using Look-Ahead for Path Following

The LMPC has a cost function which mainly penalizes Euclidean distance to a goal position, exactly like the TMPC. The main difference here is that the goal position is instead a waypoint some predetermined distance ahead of the EV on the path as seen in Figure 6.2. The predicted MPC trajectory is clearly trying to move the EV towards this waypoint on the path.

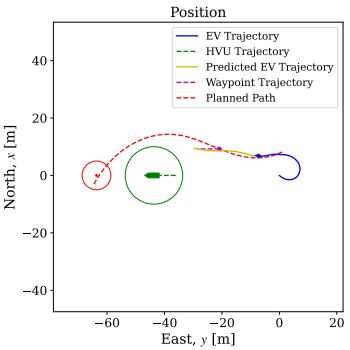


Figure 6.2 Example behavior of the LMPC.

Time-Invariant Path Following MPC with Look-Ahead

The predicted trajectory of the PMPC in Figure 6.3 follows the path almost perfectly which is reasonable since the references in each time step of the cost function are different points on the path with a tangential heading. This means that this MPC should always strive to follow the path perfectly until it has reached the target position.

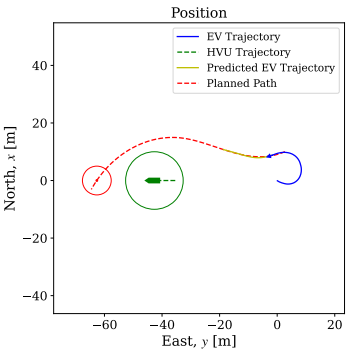


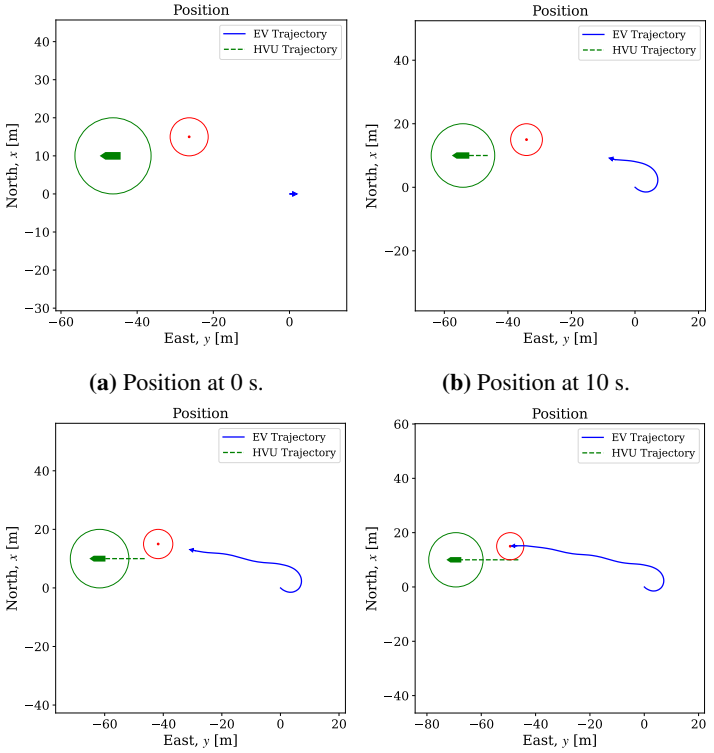
Figure 6.3 Example behavior of the PMPC.

6.4 Simulation Results for Reaching the Target Positions

The following simulations are setup as follows, the EV starts in the origin with a heading of $+\frac{\pi}{2}$ rad. The HVU is initialized 10 meters north and -40 meters east relative to the EV with a heading of $-\frac{\pi}{2}$ rad and a velocity of 1.0 m/s. The first target position is located 20 meters behind and 5 meters to the right of the HVU. Once the EV has reached the first target position, a new target position command is issued located 20 meters straight ahead of the HVU. This means that the EV has to keep clear of the HVU area in order to reach the second target position. All of the MPCs are updated with a sampling period of 200 ms and the Motion Planner is updated with a sampling period of 5 s. In these simulations the TMPC has a prediction horizon of $N = 50$, the LMPC has $N = 40$, and the PMPC has $N = 30$.

Trajectory Tracking MPC for Following a Moving Target

For the first target position in Figure 6.4, it is clear that the TMPC is able to perform a U-turn and reach the target efficiently. For the second target position, the MPC keeps clear of the HVU and reaches the second target position as well. In Figure 6.5c, it seems as if the EV has passed through the HVU radius, however, this is just the HVU that has caught up with the EV's past positions. From the telemetry plot in Figure 6.5e it can be seen how the MPC utilizes the maximum surge of 3 m/s when moving the EV to the first target position and then from the first to the second target position. In the surge plot, at around 80 s, it can be seen that the MPC slows down to stop the EV from entering the HVU radius, and waits a few seconds before continuing at max surge. At around 30 s and 110 s it can also be seen that the controller switches from the MPC to the station keeping PID where the heading converges to $-\frac{\pi}{2}$ rad and the surge converges to 1.0 m/s. It can also be seen that both the yaw rate and sway are kept around 0 except for the few necessary turns.



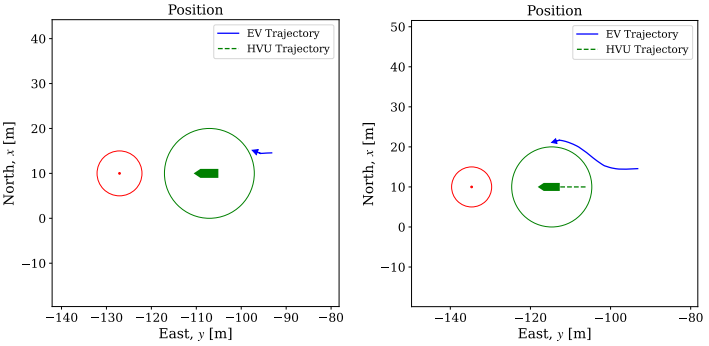
(a) Position at 0 s.

(b) Position at 10 s.

(c) Position at 20 s.

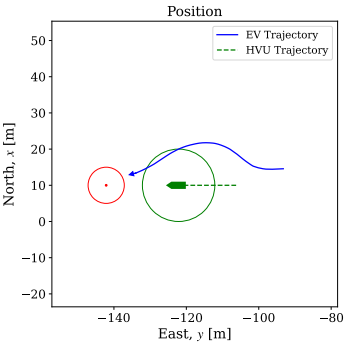
(d) Position at 30 s.

Figure 6.4 Simulation results of the USV reaching the first target position using the TMPC.

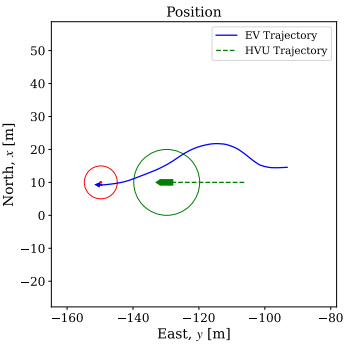


(a) Position at 80 s.

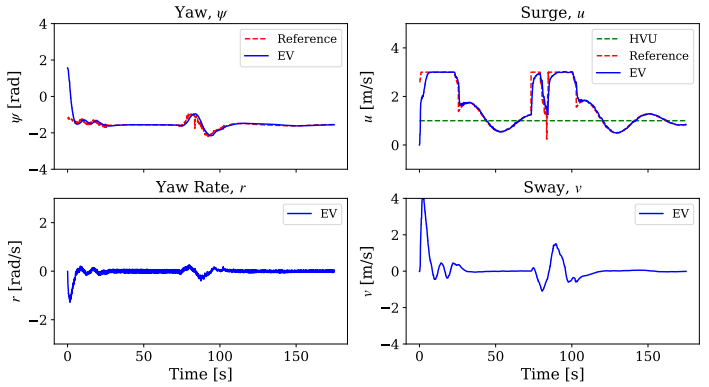
(b) Position at 90 s.



(c) Position at 100 s.



(d) Position at 110 s.

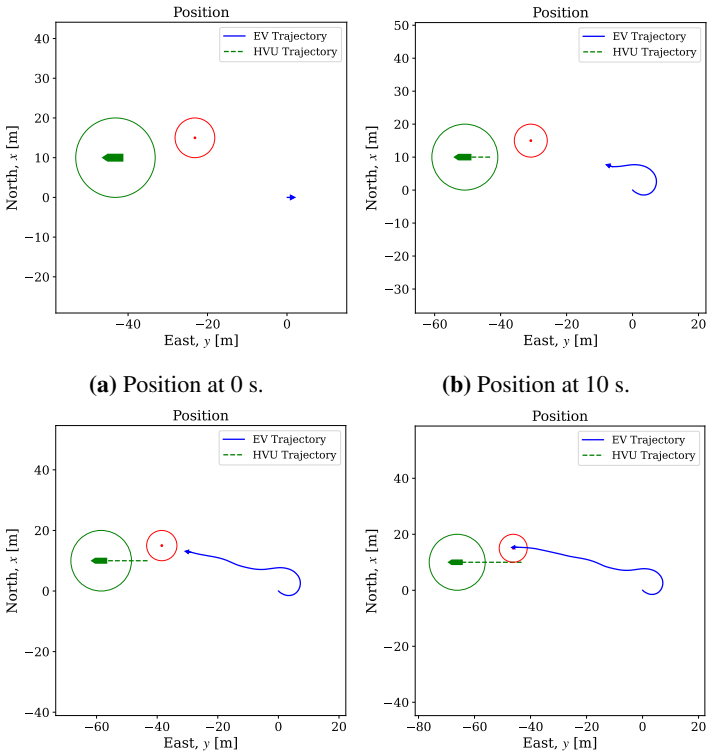


(e) Telemetry: Yaw, Surge, Yaw Rate, and Sway for the entire simulation.

Figure 6.5 Simulation results of the USV reaching the second target position using the TMPC.

Trajectory Tracking MPC Using Look-Ahead for Path Following

The LMPC reaches the first target position with no issues as seen in Figure 6.6. For the second target position, it can be seen in Figures 6.7b and 6.7c that this MPC passes the HVU with more of a margin than the TMPC before reaching the second target. The telemetry plots in Figure 6.7e show similar behavior as the TMPC where the surge is pushed to its limit before reaching the targets.



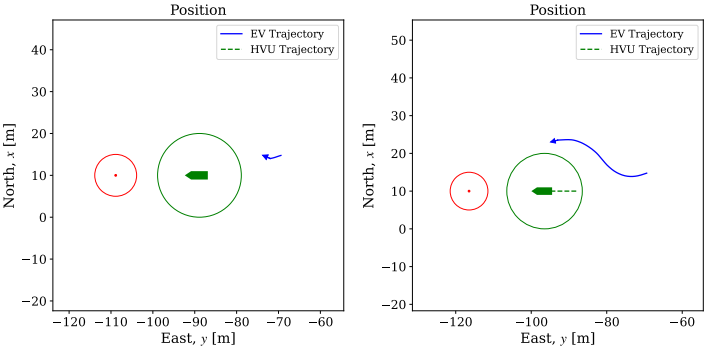
(a) Position at 0 s.

(b) Position at 10 s.

(c) Position at 20 s.

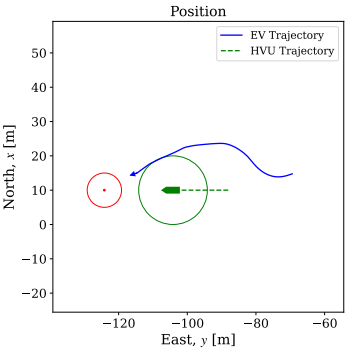
(d) Position at 30 s.

Figure 6.6 Simulation results of the USV reaching the first target position using the LMPC.

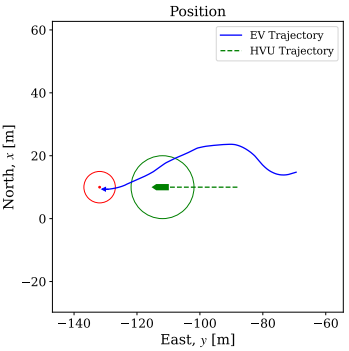


(a) Position at 60 s.

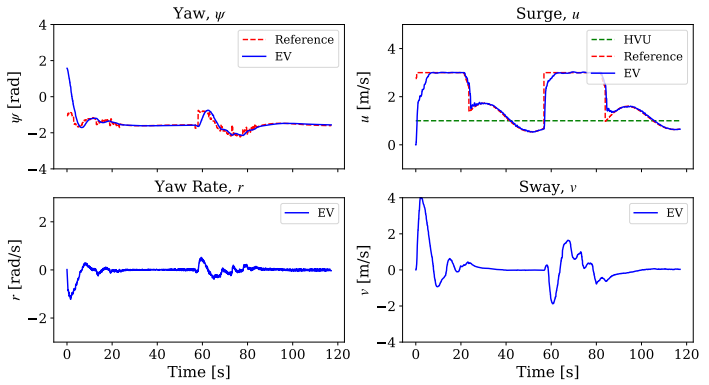
(b) Position at 70 s.



(c) Position at 80 s.



(d) Position at 90 s.

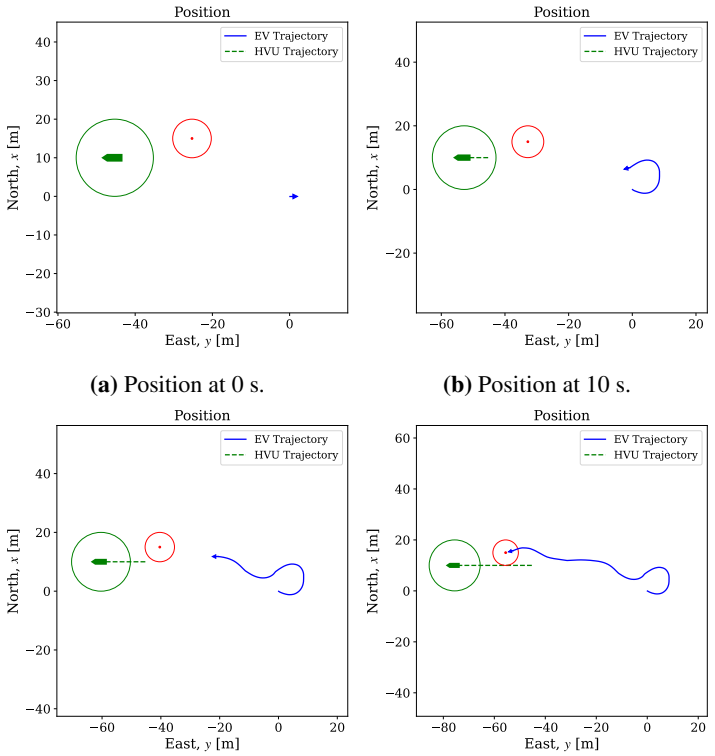


(e) Telemetry: Yaw, Surge, Yaw Rate, and Sway for the entire simulation.

Figure 6.7 Simulation results of the USV reaching the second target position using the LMPC.

Time-Invariant Path Following MPC with Look-Ahead

In Figure 6.8 it is shown that the PMPC also reaches the first target with no issues. For the second target in Figure 6.9 it has a similar approach as the LMPC where it passes the HVU radius with some margin before reaching the second target position. In the telemetry plots in Figure 6.9e, both references for yaw and surge have a couple sudden spikes while the PMPC is running. Otherwise, the plots for yaw rate and sway are both kept at reasonable levels.



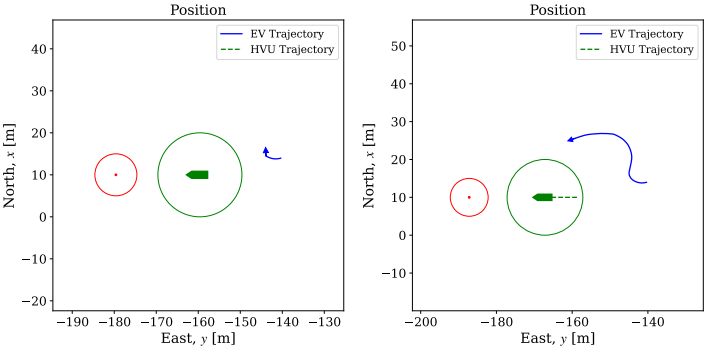
(a) Position at 0 s.

(b) Position at 10 s.

(c) Position at 20 s.

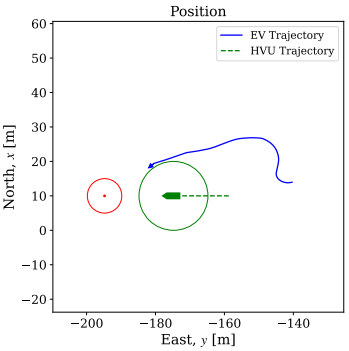
(d) Position at 40 s.

Figure 6.8 Simulation results of the USV reaching the first target position using the PMPC.

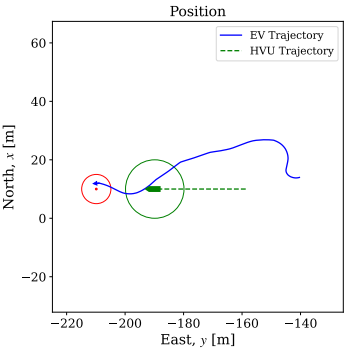


(a) Position at 150 s.

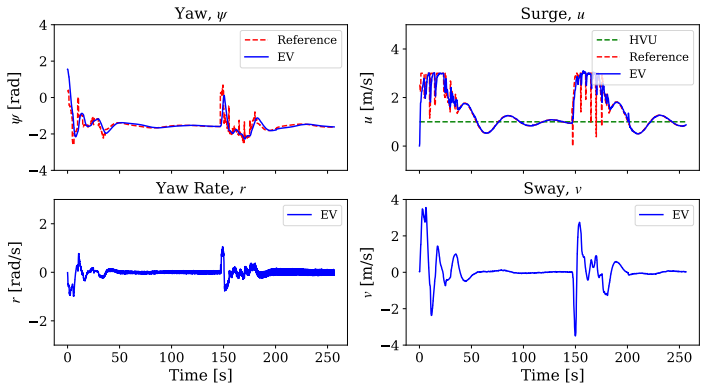
(b) Position at 160 s.



(c) Position at 170 s.



(d) Position at 190 s.

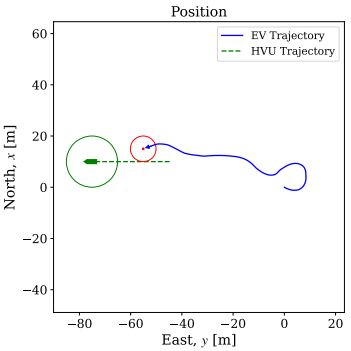


(e) Telemetry: Yaw, Surge, Yaw Rate, and Sway for the entire simulation.

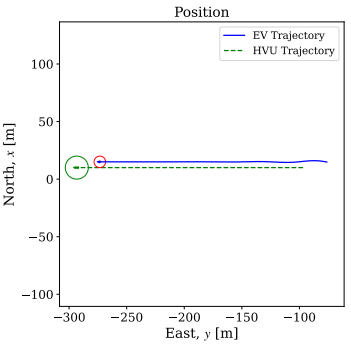
Figure 6.9 Simulation results of the USV reaching the second target position using the PMPC.

6.5 Simulation Results for Escorting Using P/PI Controllers

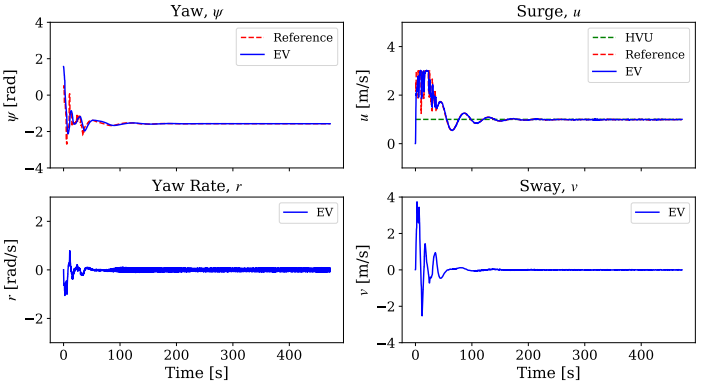
As seen in Figure 6.10a the EV reaches the target position at 40 s and then 5 minutes later in Figure 6.10b the EV has stayed inside the target area and followed the HVU for the entire duration. The telemetry plots in Figure 6.10c also reaches steady state after 40 s where yaw converges to $-\frac{\pi}{2}$ rad, surge converges to 1.0 m/s, and yaw rate and sway converge to 0.



(a) Position at 40 s.



(b) Position at 340 s.



(c) Telemetry: Yaw, Surge, Yaw Rate, and Sway for the entire simulation.

Figure 6.10 Simulation results of the USV target station keeping.

7

Experiments

Experiments were performed on 2026-04-23 in the harbor in Karlskrona shown in Figure 7.1 using the Mini-Piraya USV described in Section 2.5 as the EV. The test area also had a breakwater not seen on the map, where the experiments were performed on the shielded side of the breakwater. On the day of the experiments there were moderate winds and waves, but not a significant amount of either. The experiments were performed by manually steering the boat out to the starting position using a remote controller then running the controller remotely on a computer on land and sending messages between the boat and the computer via an internet connection. Originally, the controller was supposed to be deployed on the hardware during the experiments, but the MPCs were too computationally heavy to run effectively on the Raspberry Pi of the Mini-Piraya.

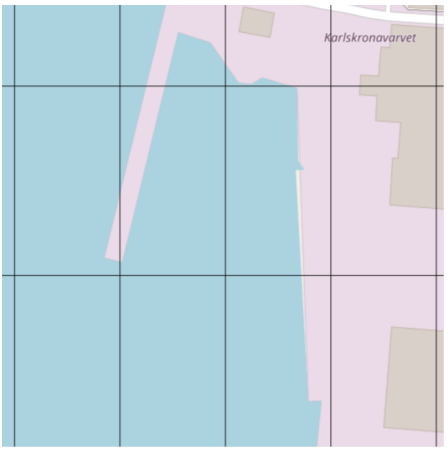


Figure 7.1 A map of the area where the experiments were performed [Open Sea Map, 2026].

7.1 Experiment Setup

The experiments were focused on getting the EV to the target position to be able to begin the escort. In the first experiment, the EV had to pass a stationary HVU and in the second experiment the HVU was instead moving. This focus was chosen since the MPC implementations are the most complex part of the controller and evaluating them through field experiments was prioritized. The station keeping P/PI controllers were developed after the experiments took place, so this part of the escort is only tested in simulations.

The HVU was entirely simulated and initialized to a spot in the middle of the harbor with a heading of 0 rad. The target position was located 20 m straight ahead of the HVU. The EV was manually placed south of the HVU pointing approximately north before every experiment. The controller was running remotely on a computer on land sending surge and yaw references to the EV via ROS messages. However, because of communication issues, thrust commands were used instead of surge setpoints. The surge references were therefore roughly translated to thrust references by comparing the surge references with the maximum surge of the EV. During the experiments this can be seen in the surge graphs where the actual surge is much higher than the reference surge which is an effect of using thrust as the control input. To prevent accidents, the EV surge reference upper bound was set to 1 m/s in all of the experiments. This was low enough to ensure a safe manual intervention.

7.2 Experimental Results with Stationary HVU

The first experiments were performed with a stationary HVU to make the scenario as simple as possible and to ensure that the MPCs could avoid the HVU well. This also gave good insight into how the EV behaved and if it worked having the controller running remotely instead of on the EV.

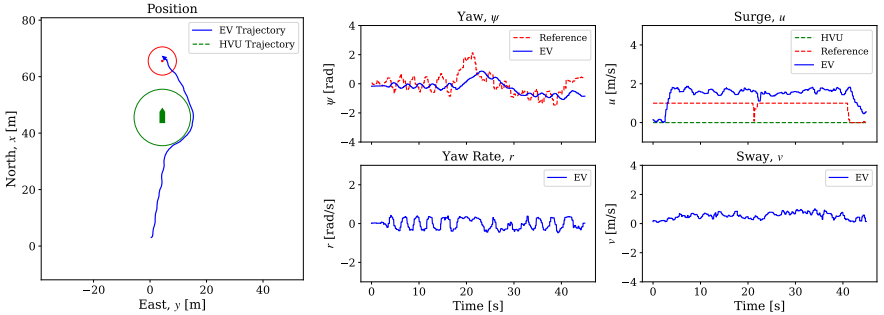
Trajectory Tracking MPC for Following a Moving Target

Since the HVU was stationary, the target position was also stationary making this TMPC equivalent to a target tracking MPC since the whole trajectory contained the same position. In Figure 7.2a the EV keeps right the to edge of the safety area and passes the HVU without ever entering it. In Figure 7.2b the yaw, yaw rate, and sway are a bit oscillatory during the entire experiment. The surge plot shows that the MPC saturates the surge reference when it can, and at around 20 s the reference dips, most probably, as a result of the EV approaching the HVU and needing to turn.

Trajectory Tracking MPC Using Look-Ahead for Path Following

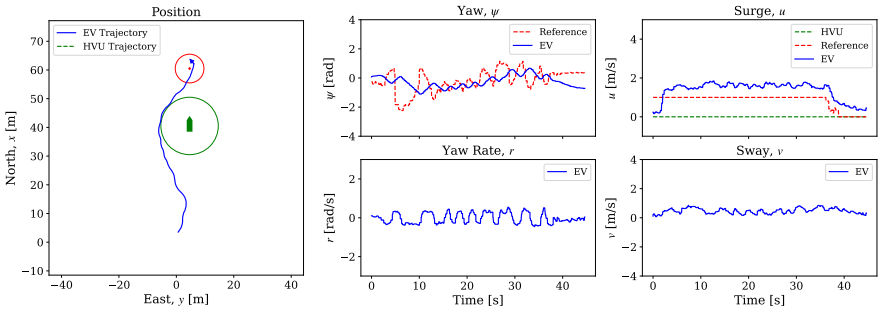
The trajectory in the LMPC will also contain the same position on the path for the entire prediction horizon. The EV in Figure 7.3a turns out slightly earlier than the

TMPC to pass the HVU, but also keeps quite close to the edge of the radius to minimize the distance it has to travel. The telemetry plots in Figure 7.3b show very similar behavior for all states when compared to the TMPC.



(a) Position. **(b)** Telemetry: Yaw, Surge, Yaw Rate, and Sway.

Figure 7.2 Experimental result of passing a stationary HVU using the TMPC.

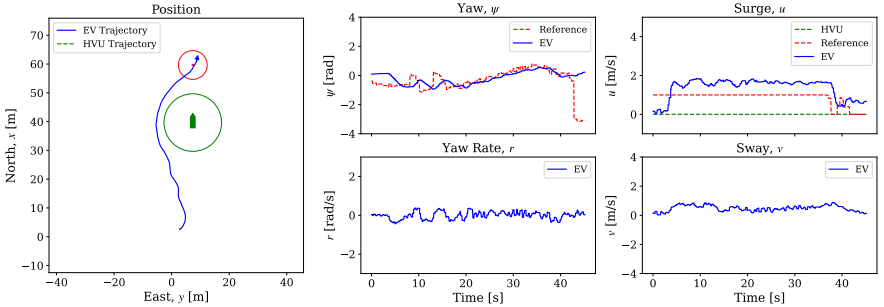


(a) Position. **(b)** Telemetry: Yaw, Surge, Yaw Rate, and Sway.

Figure 7.3 Experimental result of passing a stationary HVU using the LMPC.

Time-Invariant Path Following MPC with Look-Ahead

As shown in Figure 7.4a, the EV passes the HVU area with some margin. This is because of the PMPC following the paths more carefully and the paths being planned around a larger HVU area, as mentioned previously. The telemetry plots in Figure 7.4b show similar behavior for surge, yaw rate, and sway when compared to the other two MPCs. The yaw plot is, however, somewhat smoother for the second half of the experiment.



(a) Position. **(b)** Telemetry: Yaw, Surge, Yaw Rate, and Sway.

Figure 7.4 Experimental result of passing a stationary HVU using the PMPC.

7.3 Experimental Results with Moving HVU

The second experiments were performed under the same premise as described in the experiment setup with the exception that the HVU was also moving with a velocity of 0.5 m/s.

Trajectory Tracking MPC for Following a Moving Target

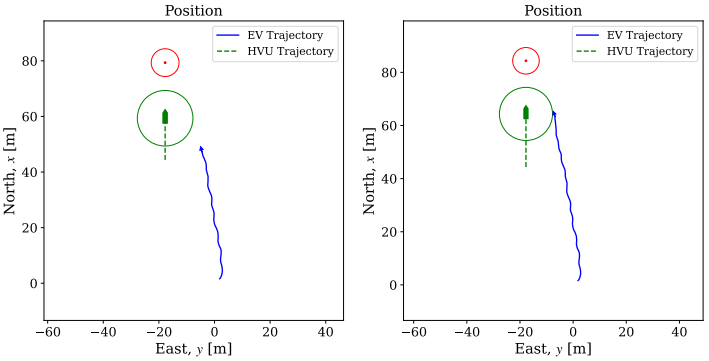
In Figure 7.5d it appears as if the EV passes through the HVU area, but looking at Figure 7.5c, it is clear that the EV has passed the HVU safely. The telemetry plots in Figure 7.5e are also somewhat oscillatory.

Trajectory Tracking MPC Using Look-Ahead for Path Following

The LMPC, once again, passes the HVU efficiently keeping a longer distance away from the HVU radius as shown in Figures 7.6c and 7.6d. The states in the telemetry plots in Figure 7.6e are in this case also somewhat oscillatory.

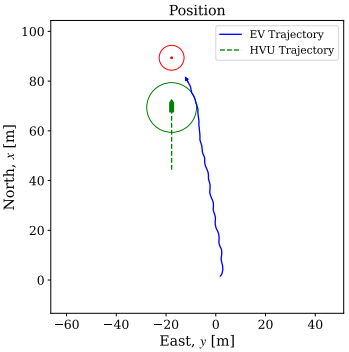
Time-Invariant Path Following MPC with Look-Ahead

The PMPC, like the LMPC, also keeps a safe distance from the HVU radius when passing it as shown in Figure 7.7c. The states in the telemetry plots in Figure 7.7e are in this case also somewhat oscillatory.

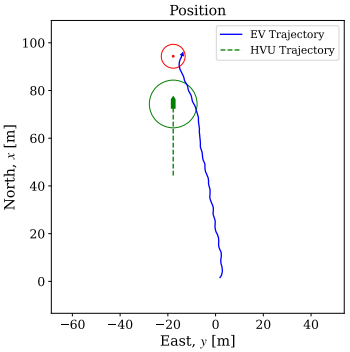


(a) Position at 30 s.

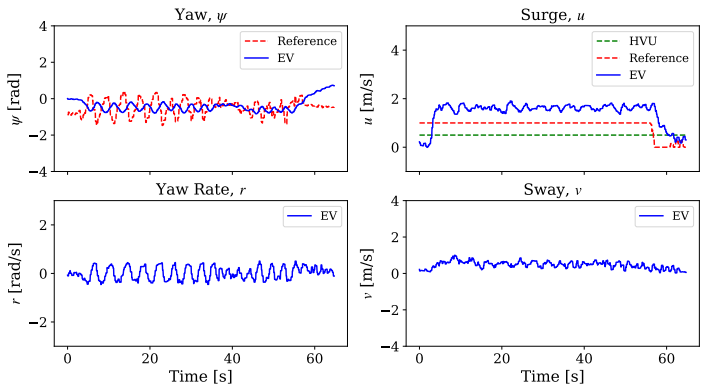
(b) Position at 40 s.



(c) Position at 50 s.

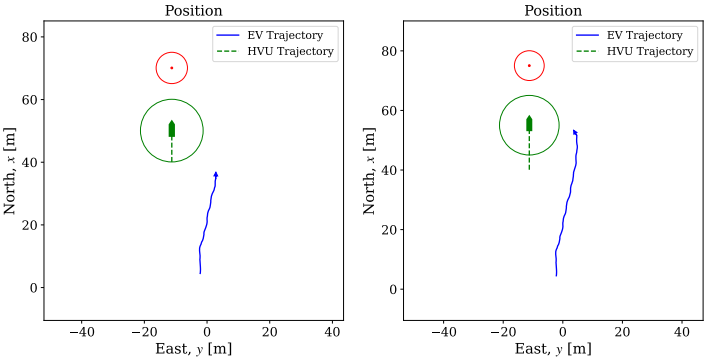


(d) Position at 60 s.



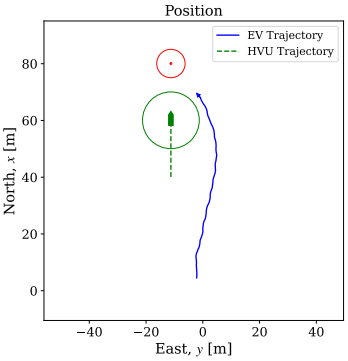
(e) Telemetry: Yaw, Surge, Sway, and Yaw Rate.

Figure 7.5 Experimental result of passing a moving HVU using the TMPC.

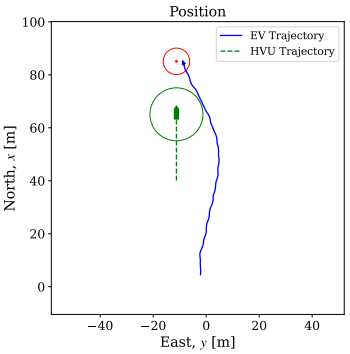


(a) Position at 20 s.

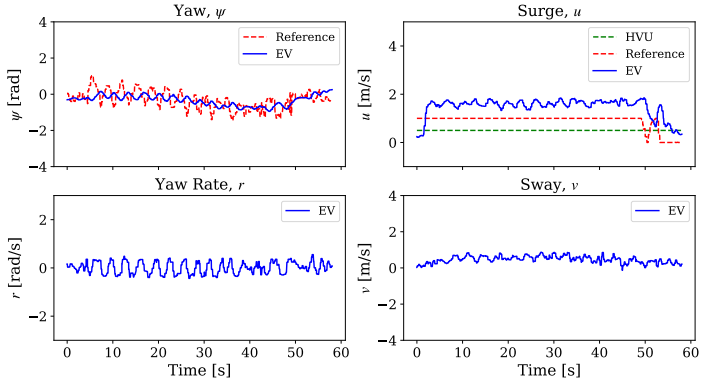
(b) Position at 30 s.



(c) Position at 40 s.

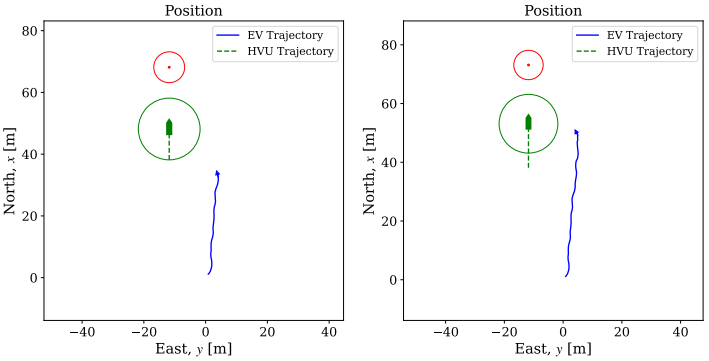


(d) Position at 50 s.



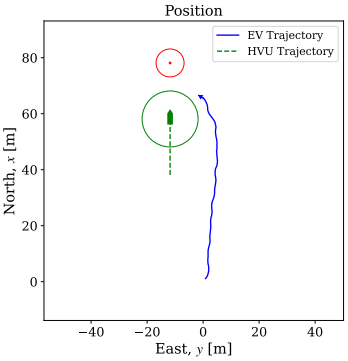
(e) Telemetry: Yaw, Surge, Sway, and Yaw Rate.

Figure 7.6 Experimental result of passing a moving HVU using the LMPC.

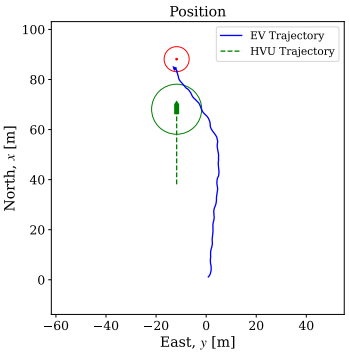


(a) Position at 20 s.

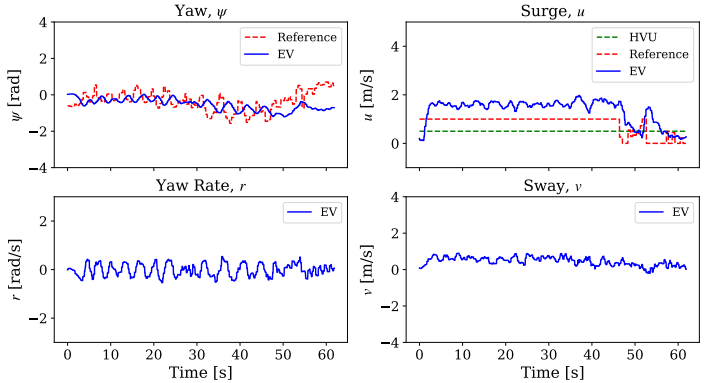
(b) Position at 30 s.



(c) Position at 40 s.



(d) Position at 60 s.



(e) Telemetry: Yaw, Surge, Sway, and Yaw Rate.

Figure 7.7 Experimental result of passing a moving HVU using the PMPC.

8

Discussion

In this chapter, each of the MPC implementations, the path planning algorithm, and the EKF are reviewed based on the research questions as well as their respective strengths, limitations, and potential improvements observed in the simulations and experiments. To recall from Section 1.2, the research questions are:

Q1 How can the MPC be designed and implemented to utilize existing controllers for velocity and heading?

Q2 How can the MPC be designed and implemented to handle dynamic spatial constraints?

Q3 How should the high-level controller architecture be organized? What are the functions of the motion planner and the MPC?

Q4 How does the controller perform implemented on real hardware, subjected to disturbances such as wind, waves, currents, and hardware noise?

8.1 Q1: Integration With Existing Controllers

The standard way to model a marine vessel using Fossen's model could not be used since this model describes the forces in the system, where the forces are controlled by the unknown internal controllers. To utilize these existing controllers for velocity and heading, the dynamics were approximated as first-order systems, as described in Section 3.2. This allowed the MPC to treat the system dynamics and the low-level PID controllers as a black-box, which the MPC could send setpoints to rather than controlling the forces directly. Approximating the USV dynamics to this simpler kinematic model turned out to be an easier way of describing the movement for vessels operating at low speeds. Now, a purely kinematic model is an imperfect representation of the system since marine vessels inherently have a lot of inertia. Extending the model with the sway dynamic equation helped mitigate this and reduced oscillatory behavior in the MPCs predictions. Furthermore, since yaw is two integration steps away from angular acceleration, the PID controlling this is inherently

slow and introduces lag to the system. This requires a longer prediction horizon to be able to find a solution better than not moving, but also increases the computation time. Estimating the model parameters with the EKF proved useful and more robust than empirical testing, effectively bridging the gap between the simplified model and the vessel's actual motion.

The EV trajectories, in Section 6.4, have shown that the MPCs can steer the EV with precision to the target positions implying that the model equations are a sufficiently good model of the system. Furthermore, the MPCs have also shown, in both simulations and experiments, that they are capable of staying very close to the HVU radius without crossing the line, further validating this model. Before adding the sway dynamics, the TMPC was prone to entering the HVU area, and the PMPC would oscillate around the path resulting in poor path following. Looking at the telemetry graph in Figure 6.7, both surge and yaw follow the references very well for all simulations. The controllers had more issues with both yaw and surge in the experiments, which is to be expected since the measurements were not ideal and the thrust was controlled instead of surge, meaning that the model used here was not entirely correct. However, the MPCs performed fine even though the internal surge controller had some stationary error.

8.2 Q2: Dynamic Spatial Constraints

A safety area around the HVU is implemented as a constraint in all of the MPC implementations where surge, yaw, and position is measured from the HVU and sent to the EV. The MPCs then predict the movement of the safety area in each step in the prediction horizon and if the EV ever enters this area the corresponding slack variables in the MPCs are used. In both simulations and experiments, the MPCs never violated this constraint and showed good performance in getting around the HVU safely, as can be seen in Figure 7.5. Now, this works well as long as the EV stays outside of this safety area, but when the EV enters the area and the slack variable is used, a very large cost is added to the cost function. Having too large costs could lead to numerical instability in the MPCs which could, in turn, lead to the solver becoming worse and computing bad solutions. However, not having the slack variable and its corresponding cost at all would instead result in infeasible solutions if the EV ever entered the area. This situation would be even worse since the MPCs would not be able to compute any viable solutions if the constraint were to be broken resulting in the control signals not being updated. A back-up controller could potentially be added to the controller architecture as another safety measure. Overall, the current implementation of this constraint works well, but could be improved to make the MPCs more robust to all scenarios.

The motion planner also plans around the HVU safety area acting as a secondary safety measure to ensure that the EV does not enter this area. However, since the HVU is assumed to be stationary when planning, the restricted radius in the mo-

tion planner is made larger to avoid situations where the HVU would move and potentially block the path. The path is also only updated every couple of seconds as changing the path too often could lead to instability in the MPCs. The motion planner could have been improved to handle a moving obstacle, but this was not prioritized since the constraint in the MPCs handled it well for this use case. Taking motion into consideration would also greatly increase the complexity and computation time of the motion planner, but could result in a better path meaning that the path could be updated less frequently.

8.3 Q3: Controller Architecture

This section evaluates the structure of the control system, and more specifically the task division between the motion planner and the MPC. The TMPC does not utilize any path planning meaning that all the work is allocated to the MPC. Since MPCs are, in general, quite short sighted the TMPC tended to go straight towards the target position and could not see the HVU until it was within the prediction horizon. This means that the TMPC would get very close to the HVU radius and need to slow down in order to turn and steer clear of the HVU. This results in a suboptimal route in regards to time and distance. Another major drawback is that it can become numerically unstable when being far away from the target position because of the quadratic cost on the position error.

The LMPC design is quite similar to the TMPC with the exception that the LMPC utilizes the path created by the motion planner for more long-term planning. This means that the position that the LMPC minimizes towards is always within the horizon, thus removing the issue of numerical instability. Using the path to maneuver also results in a smoother route since the MPC is guided to turn preemptively when passing the HVU.

The third design, the PMPC, places a lot more responsibility on the motion planner. Two major strengths with this structure is that this MPC is also more numerically stable since it always has the references within the horizon, and it will always follow the shortest path to the goal. Since this MPC strives to follow the path perfectly it is, however, very reliant on the path being good. This implementation also has the downside of needing to be recompiled each time the path is updated which can take a little bit of time meaning that the motion planner cannot update the path all too often.

Overall, the MPCs which utilize the motion planner maneuver the EV more proactively than the TMPC, but are also very dependent on the path begin good. The LMPC and the PMPC are better at staying outside of the HVU safety area since they utilize both the MPC constraint and the motion planner constraint, but the TMPC is faster at reacting to changes and is more versatile. Despite the differing complexity, with the TMPC being the simplest implementation and the PMPC being the most complex, all three implementations can be used to accomplish the escort mission

scenarios described previously.

8.4 Q4: Performance Subject to Disturbances on Hardware

When comparing the simulations to the experiments, all experimental results had some oscillatory behavior. They all avoided the restricted zone and got to the target position despite not having an ideal internal controller for surge. The oscillatory behavior could be a consequence of not having ideal internal controllers, measurement noise being introduced by sensors, environmental disturbances such as wind and waves being present, or even lag being introduced when running the software remotely on a different computer. Overall, the performances in the experiments are almost as good as in the simulations and proved that this more general USV model could easily be adapted and applied to a real marine vessel.

9

Conclusion

This thesis has investigated how model predictive control and motion planning can be designed and implemented on a USV to enable autonomous escorting of another marine vessel. Three different MPC implementations paired with an EKF, a motion planner, and a station keeping P/PI controller have been presented to enable the autonomous escort mission. The method has been tested and evaluated in simulations using the VRX environment, and in field experiments on the Mini-Piraya USV hardware. The results from both the simulations and the experiments show that all three MPC implementations can be used to accomplish this mission, with each one bringing slightly differing advantages and limitations.

9.1 Future Work

Firstly, this thesis mainly focuses on MPC designs and therefore uses a very simple motion planner. The motion planner is something that could be further improved to handle moving spatial constraints, taking motion into consideration where hybrid A* or lattice planning could potentially be used. Secondly, the scenarios investigated in this thesis all contain only one EV, although realistically there could be many. Both MPC and motion planning could be improved to handle situations where there could be multiple EVs present at once, and all of them knowing or not knowing each others goals. Lastly, the model used in this thesis is designed to work well with onboard controllers and on varying USV platforms. The model and the EKF could be improved upon to truly work on any USV without modifications needed.

Bibliography

Andersson, J. A. E., J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl (2019). "CasADi – A software framework for nonlinear optimization and optimal control". *Mathematical Programming Computation* **11**:1, pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

Behrendt, M. (2009). *MPC Scheme Basic*. Wikimedia Commons. Available at https://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg. Licensed under CC BY-SA 3.0. (Visited on 2026-04-16).

Bernhardsson, B. and K. J. Åström (2016). *Model Predictive Control (MPC)*. URL: https://www.control.lth.se/fileadmin/control/Education/DoctorateProgram/ControlSystemsSynthesis/2016/MPC.pdf.

Bingham, B., C. Aguero, M. McCarrin, J. Klamo, J. Malia, K. Allen, T. Lum, M. Rawson, and R. Waqar (2019). "Toward Maritime Robotic Simulation in Gazebo". In: *Proceedings of MTS/IEEE OCEANS Conference*. Seattle, WA. DOI: 10.23919/OCEANS40490.2019.8962724.

Boretti, A. (2025). "Navigating the future: the expanding role of unmanned surface vehicles in maritime security". *Journal of Transportation Security* **18**:24. DOI: 10.1007/s12198-025-00314-x.

Cai, Q., Q. Huang, M. Ye, Q.-L. Han, and L. Ding (2026). "An AMPC and Intruder Maneuver Prediction Based Algorithm for Multi-USV Escort". *IEEE Transactions on Vehicular Technology*. DOI: 10.1109/TVT.2026.3655697.

Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs". *Numerische Mathematik* **1**:1, pp. 269–271. ISSN: 0945-3245. DOI: 10.1007/BF01386390.

Faulwasser, T. and R. Findeisen (2016). "Nonlinear Model Predictive Control for Constrained Output Path Following". *IEEE Transactions on Automatic Control* **61**:4, pp. 1026–1039. DOI: 10.1109/TAC.2015.2466911.

Fossen, T. I. (2021). *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley. DOI: 10.1002/9781119575016.

Glad, T. and L. Ljung (2000). *Control Theory: Multivariable and Nonlinear Methods*. Taylor & Francis. ISBN: 9780748408771.

Hart, P. E., N. J. Nilsson, and B. Raphael (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *IEEE Transactions on Systems Science and Cybernetics* **4**:2, pp. 100–107. DOI: 10.1109/TSSC.1968.300136.

Hunter, J. D. (2007). "Matplotlib: A 2D Graphics Environment". *Computing in Science & Engineering* **9**:3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

Kockum, S. (2022). *Autonomous Docking of an Unmanned Surface Vehicle using Model Predictive Control*. MA thesis TFRT-6164. Department of Automatic Control, Lunds University. URL: https://lup.lub.lu.se/student-papers/search/publication/9096721.

Liao, Y., Y. Ge, Z. Song, R. Qian, W. Yu, Y. Xu, Y. Li, and S. Pang (2025). "Research on improved null space based unmanned surface vehicle formation maneuver control method for merchant ship escort". *Ocean Engineering* **319**, pp. 120–225. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2024.120225.

Liu, Z., Y. Zhang, X. Yu, and C. Yuan (2016). "Unmanned surface vehicles: An overview of developments and challenges". *Annual Reviews in Control* **41**, pp. 71–93. ISSN: 1367-5788. DOI: https://doi.org/10.1016/j.arcontrol.2016.04.018.

Ljungberg, F. (2021). *Systemidentifiering för Piraya*. Tech. rep. Internal report, proprietary data. Saab Kockums AB.

Macenski, S., T. Foote, B. Gerkey, C. Lalancette, and W. Woodall (2022). "Robot Operating System 2: Design, architecture, and uses in the wild". *Science Robotics* **7**:66, eabm6074. DOI: 10.1126/scirobotics.abm6074.

Open Sea Map (2026). *OpenSeaMap - The free nautical chart*. Available at https://map.openseamap.org/. Licensed under CC BY-SA 2.0. (Visited on 2026-05-05).

Piegl, L. and W. Tiller (1997). *The NURBS Book*. 2nd. Springer-Verlag Berlin Heidelberg. DOI: 10.1007/978-3-642-59223-2.

Rawlings, J. B., D. Q. Mayne, and M. M. Diehl (2017). *Model Predictive Control: Theory, Computation, and Design*. 2nd ed. Nob Hill Publishing. ISBN: 9780975937730.

Sauer, T. (2012). *Numerical Analysis*. 2nd ed. Pearson Education, Inc. ISBN: 9780321783677.

Simon, D. (2006). *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. John Wiley & Sons, Inc., Hoboken, New Jersey. ISBN: 978-0-471-70858-2.

Svedberg, M. (2023). *Data-Driven Adaptive Control of Unmanned Surface Vehicles Using Learning-Based Model Predictive Control*. MA thesis TFRT-6205. Department of Automatic Control, Lunds University. URL: https://lup.lub.lu.se/student-papers/search/publication/9136756.

Wingqvist, B., B. Olofsson, J.-O. Lindh, A. Robertsson, and R. Johansson (2024). "Collision Avoidance for ASVs in Archipelagos—A COLREGs-Aware Optimization-Based Method". In: *2024 European Control Conference (ECC)*, pp. 1139–1146. DOI: 10.23919/ECC64448.2024.10591135.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
June 2026

Document Number
TFRT-6310

Author(s)
Axel Lagerstedt
Sean Small

Supervisor
Erik Börjesson, Saab Kockums AB, Sweden
Johan Silvander, Saab Kockums AB, Sweden
Björn Olofsson, Dept. of Automatic Control, Lund University, Sweden
Yiannis Karayiannidis, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Autonomous Escort with an Unmanned Surface Vessel Using Model Predictive Control

Abstract

This thesis investigates the design and implementation of Model Predictive Control (MPC) and motion planning to enable an Unmanned Surface Vessel (USV) to perform autonomous escort missions. The objectives were to assess how the control architecture could utilize preexisting internal USV controllers and handle dynamic obstacles. Furthermore, how the controller architecture should be organized and how the controller performs when implemented on hardware were also explored. The proposed controller architecture consists of an MPC equipped with a generalized USV model to capture the unknown dynamics of the vessel where the model parameters are estimated online using an Extended Kalman Filter (EKF). The MPC is also equipped with a constraint which handles dynamic collision-avoidance, and a station keeping P/PI controller to ensure that the USV stays in the target position. The motion planner utilizes a custom A* search algorithm coupled with B-splines to plan long-term global paths to avoid obstacles and find the shortest route to the target position. Three different MPC variations, utilizing the motion planner to different degrees, were tested and validated through simulations and experiments on the Mini-Piraya USV platform by Saab Kockums AB. The results presented show that all three proposed controller variants are capable of steering the USV with precision, validating that the USV model paired with the EKF gives a good approximation of the system dynamics when operating at low velocities. The collision-avoidance implementation in the controller shows good performance and the USV is always kept at a safe distance when navigating around the escorted vessel. The experimental results also show that the controllers perform well when subjected to environmental disturbances, sensor noise, and lag proving that this more general USV model could easily be adapted and applied to a real marine vessel. Overall, the results from both the simulations and the experiments show that all three MPC implementations can be used to accomplish the escort mission, with each one bringing slightly differing advantages and limitations.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language	*Number of pages*	*Recipient's notes*
English	1-61	
Security classification		

http://www.control.lth.se/publications/