University of South-Eastern Norway

www.usn.no

FMH606 Master's Thesis 2022

Trajectory following using model predictive control for SeaDrone collision avoidance using Robot Operating System

Syed Sami Al Haq

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606 Master's Thesis, 2022

Title: Trajectory following using model predictive control for SeaDrone collision avoidance using Robot Operating System

Number of pages: 29

Keywords: MPC, USV, GAZEBO, CASADI, VRX, ROS, rudderless double thruster, 3DOF dynamic model

Student: Syed Sami Al Haq

Supervisor: Fabio Augusto de Alcantara Andrade

External partner: N/A

Summary:

Navigation of unmanned surface vehicle (USV) is much important now a days due to much improvement in communication system. Details survey information can be collected using USV by passing it over wireless communication. Even remote and natural disaster effected places can be greatly helped by this. This paper focuses on collision avoidance self-control of USV using model predictive control. For testing and simulation GAZEBO is used. ROS is also used to communicate with USV sensor.

This experiment is worked out mainly in two steps. First stability of control and mathematical model is checked using mathematical simulation. Later it is tested in GAZEBO which simulates real life effect on experiment object in terms of forces acting on object.

Satisfactory outcome is achieved in these steps. Using CasAdi for MPC was stable.

Preface

This master's thesis in done in University of South-Eastern Norway (USN), Department of Electrical engineering, Information Technology and Cybernetics on autumn semester of 2022. This work is related to Valkyrie Project which is funded by EU with many partners. Main goal of this project is to fast response in post disaster scenarios. USN took the responsibility to develop autonomous navigation of SeaDrone to recon, search and rescue. This work in USN also used to attend SeaDrone competition held by autodrone.no. This paper mainly focuses on the development of trajectory following with collision avoidance. Final testing is done in GAZEBO.

I would like to thank my supervisor Fabio Augusto de Alcantara Andrade very first, and co-supervisor Carlos Pfeiffer for their valuable guidance and support to complete the work. Every step of work was very smooth only because of them sharing all valuable resources that they have had along with their experience.

Porsgrunn, 28th of May 2022

Syed Sami Al Haq

Contents

List of Figures

Acronyms

ASV Autonomous Surface Vehicle

CAS Collision Avoidance System

CasADi

COLREGS International Regulations for Preventing Collisions at Sea

DOF Degrees of Freedom

ENU East North Up

GPS Global Positioning System

IMU Inertial Measurement System

MPC Model Predictive Control

PID Proportional Integral Derivative

ROS Robotic Operating System

SB-MPC Simulation-Based Model Predictive Control

UAV Unmanned Aerial Vehicle

USV Unmanned Surface Vehicle

VO Velocity Obstacle

1 Introduction

History shows that industrial revolution was happened when engine was invented; that is to say automation. It is no wonder that automation has made a way for us to reach upto this high level of comfort in life. After the digital era has started and with much more computational power, automation has changed its course and currently solving much more difficult task which was never thought of before.

Advancement of computational power has opened up way to solve problems faster than ever. Time consuming calculation like iterations or decision based on large data are no more obstacle to handle. In similar fashion, control system also reached reliable and stable position. Simulation of model to predict future behavior is no wonder now a days.

This project tries to solve such a problem, automatic control of USV, by model predictive control. The USV selected for this project is rudderless double thruster. The reason behind rudderless is to minimize the model complexity which will consequently reduce the pressure of optimizing.

Main idea behind this work is to reduce computational power in optimization. Providing too much constrain i.e. obstacle avoidance or keeping safe distance etc., makes the MPC too complex and vulnerable. As we know that solvers are sometimes stuck in local minima unless constrains are widen up. In contrast widening up constrain also lead to high computational requirement. To attend this problem, this paper aims to separate all the calculation of obstacle detection and path planning funcitons from main MPC solver. MPC solver will only be given target to reach. And this target will be updated time to time to maneuver through obstacle.

1.1 Motivation

Continues development in automation lead to verities of controlling methods. Although PID was initially well stablished and widely used method, but now Model Predictive Control has taken much of space because of its accuracy compared to PID. Despite of having dependencies, still MPC seems stable and promising.

In many field MPC has made its way as control method. For example, in autonomous vehicle, aircraft, robots etc. It is also used widely in water vessels to assist navigation or to suggest economic route. Although many works has been done and implemented in these fields, but not so significant improvement has been done for low speed fully automated vessels or unmanned surface vehicle (USV).

The need of USV has recently became concern due to demand of survey in natural disastrous places or any remote or hazardous places where it may be dangerous for humans to access. This work is motivated by such a cause, to survey flood or any victim areas immediately for rescue service. European Union and some other partners has initiated the idea.

1.2 Literature Review

Article [1] has come up with a concept of separating control application into 3 layers. Where first layer does long term planning, second layer or mid-level COLAV is responsible for trajectory planning and local adaptation. Finally reactive layer follows the command from top layer and execute control signal. This paper has used simplified 3 DOF for ASV. With control signal vehicle velocity and yaw rate. MPC is used to get optimized control output. CASADI library is used with IPOPT solver. The paper has taken account for COLGERS rule number 8, 13, 14, 15, 16 and 17 while planning path.

Paper [2] focus on implementation of PID on rudderless two thruster USV. The implementation is done up to physical USV. Controller input and output are RPS of thruster and movement in x,y and r (rotation) direction. The report derive the mathematical model of thruster propeller as it is relevant to input. Then it is combined with the 3DOF model of USV. Furthermore, system identification is implemented to collect specific model parameters. In testing, different type of trajectory following task done to calculate its accuracy.

Main objective of paper [3] is to explore advance path of ship. To get future path, it is depended on the current turn rate. This future path is send to UAV for explore. MPC is used to control UAV. Limit of flight length is set in constrain. Significant improvement of error is observed when ship model is included in the calculation. For simulation, ships velocity was considered as constant. It was also concluded that test of changing ship velocity and heading was satisfactory.

This [4] paper's main objective is to minimize or avoid conventional MPC computation to implement collision avoidance system. COLREGS rules are coupled with the cost function in constrain. Instead of dealing full range in mathematical models to come with optimized solution, it simulates probable scenario and try to find out optimized solution. Control signal is also segmented to reduce probable scenario. Result demonstrate that the controller is capable to handle multi obstacle or complex scenario with ease.

The thesis paper [5] designs MPC controller to maintain COLREGS. Implementation was done in C++ within ROS framework. Comparison between linear and non-linear ship model is being done. Also SB-MPC and VO algorithm is compared. Results indicates that controller is capable to handle complex scenarios with ease. It also demonstrates that controller prefers to change its course rather than changing speed. It is also mentioned that controller is heavily depended on tuning of parameters for specific scenario.

1.3 Assumption

This project focuses on the control system which should be simple and should not contain many constrain. To achieve that some related process was assumed and mocked due to scope and time limit. Assumption and limitations can be considered as follows.

1. All sensor data are available readily and accurately without any delay.
2. No disturbance considered in the simulation. i.e. wave or wind
3. Obstacle position is known and movement and direction of movement is known.
4. Planning of route and to calculate optimal temporary trajectory is available.

1.4 Contribution

Control of USV is wide area. Although many topics has to be addressed to establish a working concept, this paper mainly focus on use of MPC theory. CasAdi [6] library is also used to have an experience on it. Use of ROS is another important software (framework) which is studied and heavily used in this work.

Simulation in GAZEBO is done to have a pre-assessment before implementation. And finally writing code in python was second main focus to address collision avoidance while automatic control.

1.5 Thesis structure

Work of this paper presented in this report mainly in chapter 2, 3, 4 and 5. Chapter 6 discuss some scope of future work.

In chapter 2, theories are discussed. These theories are considered as basic and stable working principle of the experiment. Moreover, description and fundamental information of software and simulators are also briefly discussed.

Chapter 3 elaborates how the idea behind this paper is implemented. This chapter also includes flow and logic behind the application.

Chapter 4 demonstrate testing and result of the implementation. Using graphs results are presented.

Chapter 5 explains the reason and limitations those are observed in result.

Chapter 6 discuss about very relevant future work of this paper.

Finally, chapter 7 summarize the work.

2 Theoretical Background

In this chapter theories will be briefly discussed. Based on this theories the experiment is done. State space model of USV is the main equation, on top of that all prediction of controller is done as well as simulation.

2.1 Surface Vessel Model

The USV model used in this paper is simplified 3 DOF model which is derived in [7]. Instead of 6 DOF, 3 DOF is chosen to minimize the complexity of the control equation. And this simplified 3 DOF is very much correct for USV as roll and pitch movement are barely effective. Furthermore is discussed in the paragraph below.

Generally, motion of a USV is divided into kinematics and kinetics. Kinematics deals with geometrical movements and kinetic deals with forces acting on the object.

2.1.1 Equation of motions

Kinematic: When considering kinematic, any water vessel is capable to move in 6 DOF i.e. surge, sway, heave, roll, pitch and yaw. USV having stability in heave, roll and pitch due to its construction, the equation of those DOF can be eliminate to reduce the complexity of equation. First of all, heave can be ignored due to stable position of having buoyancy and less weight. Roll and pitch movement can also be ignored due to its smaller dimension[8].

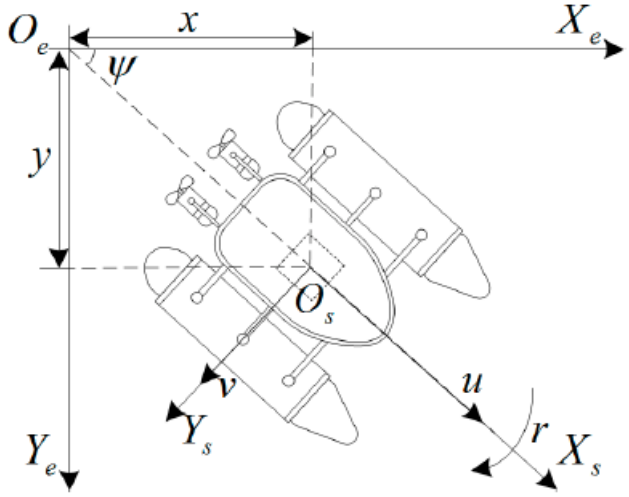


Figure 2-1 : 3 DOF USV Source: [2]

Two coordinate systems are considered when describing water vessels. One is fixed or inertial to earth $O^eX^eY^e$. Another one is USV's structure based $O^sX^sY^s$. Movement of USV is defined based on the earth frame. Direction of movement is considered based on the USV frame. Where X^s and Y^s defines direction of forward velocity u and left side velocity v respectively. Yaw rate of angular movement is based on O^s denotated as r.

Without considering any wave or wind present while experiment, typical kinematic motion can be written as[7],

$$\dot{\eta} = \mathrm{J}(\eta)\upsilon \qquad 2\text{-}1$$

Where $\eta = [\, x \;\; y \;\; \psi]^T$, x and y are position and ψ is heading of the USV. $\dot{\eta} = [\, \dot{x} \;\; \dot{y} \;\; \dot{\psi}]^T$ symbols velocity on these axes in the earth frame. $\upsilon = [\mathrm{u} \;\; \mathrm{v} \;\; \mathrm{r}]^T$ where u, v and r are surge, sway and yaw velocity. Transformation matrix $\mathrm{J}(\eta)$ can be define as,

$$\mathrm{J}(\eta) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad 2\text{-}2$$

Kinetic part:

To calculate the effect of force acting on the USV, kinetic model is also formulated. According to [2] vectorial model is chosen rather than Tylor-series expansion in Abkowitz model. Which was possible because of symmetric shape of the vessel. Kinetic model is defined as equation

$$M\dot{\upsilon} + C(\upsilon)\upsilon + D(\upsilon)\upsilon = \tau + \tau_E \qquad 2\text{-}3$$

$$M = \begin{bmatrix} m - X_{\dot{u}} & 0 & -my_g \\ 0 & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ -my_g & mx_g - N_{\dot{v}} & I_Z - N_{\dot{r}} \end{bmatrix} \qquad 2\text{-}4$$

$$C(\upsilon) = \begin{bmatrix} 0 & 0 & -m\left(x_g r + \upsilon\right) + Y_{\dot{v}}\upsilon + \frac{Y_{\dot{r}} + N_{\dot{v}}}{2} r \\ 0 & 0 & (m - X_{\dot{u}})u \\ m\left(x_g r + \upsilon\right) - Y_{\dot{v}}\upsilon - \frac{Y_{\dot{r}} + N_{\dot{v}}}{2} r & -(m - X_{\dot{u}})u & 0 \end{bmatrix} 2\text{-}5$$

$$D(\upsilon) = D + D_n(\upsilon) = \begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_\upsilon & Y_r \\ 0 & N_\upsilon & N_r \end{bmatrix} - \begin{bmatrix} X_{u|u|}|u| & 0 & 0 \\ 0 & Y_{\upsilon|\upsilon|}|\upsilon| + Y_{\upsilon|r|}|r| & Y_{\upsilon|\upsilon|}|\upsilon| + Y_{\upsilon|r|}|r| \\ 0 & N_{\upsilon|\upsilon|}|\upsilon| + N_{\upsilon|r|}|r| & N_{r|\upsilon|}|\upsilon| + N_{r|r|}|r| \end{bmatrix}$$

2-6

Where, M in mass matrix. USV body-fixed frame center of gravity is x_g and y_g. I_Z is Moment of inertia about Z-axis. C(υ), D(υ) represents Coriolis and centripetal matrix, drag matrix.

Force creating on the thruster is demonstrated as τ.

τ_E represent vector of forces caused by the disturbance.

As USV is considered as rudderless double thrusters, the thrust acting on the USV is always in forward or backward direction. The thrusters of port and starboard can vary its force and thus movement on left or right is achieved. Vector of τ is considered as following.

$$\tau = [\tau_u \quad 0 \quad \tau_r]^T \qquad 2\text{-}7$$

In equation $\tau = [\tau_u \quad 0 \quad \tau_r]^T$ 2-7, $\tau_u = X_{P1} + X_{P2}$, Where X_{P1} and X_{P2} represents force on port and starboard thruster.

$$\tau_r = (X_{P1} - X_{P2})d_p$$

Where, $\tau_u \, and \, \tau_r$ represents the result of thrusters forward/backward and rotational movement. If both thrusters are applying force in same direction with same amount, then the force will be added and there will be no movement on Y-axis and in yaw. In contrast, if powers are different and/or opposite, it will contribute to movement on Y-axis and yaw.

USV by construction is symmetric and the floating mechanism is designed in such a way that movement on heap can be ignored. Also because of its flat shape and floating position change in pitch angle and roll angle can be ignored. Considering these characteristics the model of USV is simplified into simpler kinematic equation by this author [2].

$$M\dot{v} + C(v)v + D(v)v = \tau \qquad 2\text{-}8$$

$$M = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & 0 \\ 0 & 0 & I_Z - N_{\dot{r}} \end{bmatrix} \qquad 2\text{-}9$$

$$C(v) = \begin{bmatrix} 0 & 0 & -(m - Y_{\dot{v}})v \\ 0 & 0 & (m - X_{\dot{u}})u \\ (m - Y_{\dot{v}})v & -(m - X_{\dot{u}})u & 0 \end{bmatrix} \qquad 2\text{-}10$$

$$D(v) = D = -\begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & 0 \\ 0 & 0 & N_r \end{bmatrix} \qquad 2\text{-}11$$

According to paper [2] Final model of USV sums up to,

$$\begin{cases} \dot{x} = u\cos\psi - v\sin\psi \\ \dot{y} = u\sin\psi + v\cos\psi \\ \dot{\psi} = r \\ (m - X_{\dot{u}})\dot{u} - (m - Y_{\dot{v}})vr + X_u u = X_{P1} + X_{P2} \\ (m - Y_{\dot{v}})\dot{v} - (m - X_{\dot{u}})ur + Y_v v = 0 \\ (I_Z - N_{\dot{r}})\dot{r} - \big((m - X_{\dot{u}}) - (m - Y_{\dot{v}})\big)uv + N_r r = (X_{P1} - X_{P2})d_p \end{cases} \qquad 2\text{-}12$$

Reducing the USV model to this 3DOF, which is rudderless double thruster, makes the calculation of the force acting on the USV much simpler.

GAZEBO model

Model in GAZEBO [9] used in this work takes two inputs as a control signal as for two thrusters. The model matches with the previous mathematical model so previous model was used in MPC while simulating and testing in GAZEBO

Diagram of input output can be as below.

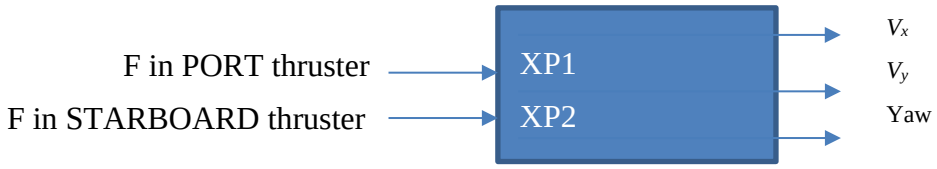


Figure 2-2: Input output diagram

2.1.2 Model of USV in GAZEBO

The VRX model [9] in GAZEBO is constructed in similar fashion as the above structure in Figure 2-13. Parameters of the USV can also be set on the XACRO file which is provided while compiling the model for GAZEBO.

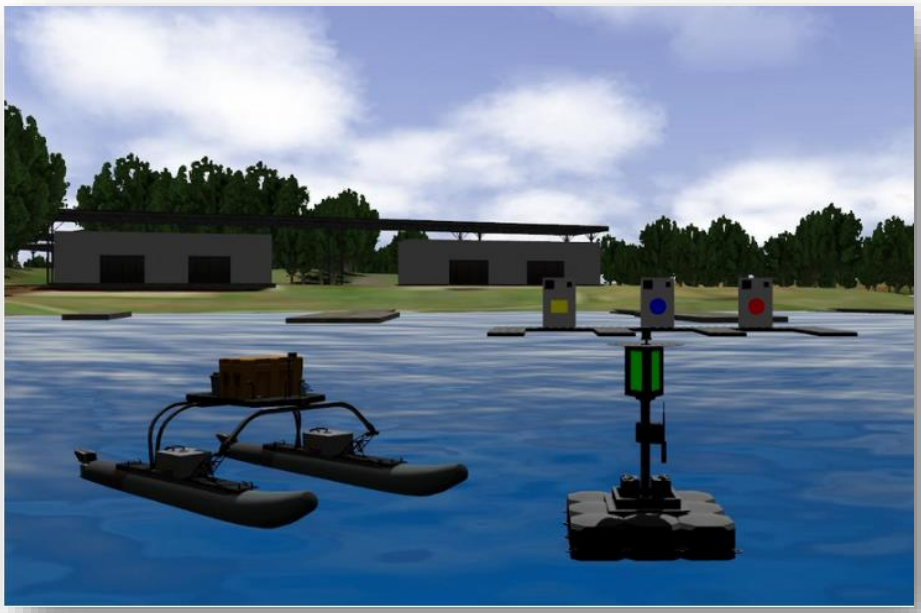


Figure 2-3: VRX model Source: [10]

GAZEBO VRX model powered by Openrobotics [10] is constructed along with the ROS. Subscribing to the proper topic can fetch GPS and IMU data. To send command, topic of thruster is used to publish control data.

2.2 Model Predictive Control

Model predictive controller is the only controller which takes account of future situation. Which drastically increase overshooting. It is dependent on dynamic model of the system with real time input from sensor so it could provide best control signal. MPC uses solvers to come up with optimized control signal with the help of optimization solver. And repetition of the fixed time frame makes it possible to solve or optimize, along with consideration of uptodate state values.[11]

In contrast to many benefits, there are also some drawbacks of this controller. This controller depends highly on dynamic model so fine model is needed to have a good control output. This controller demands high computation power because it have to solve optimization problem in each iteration.

2.3 GAZEBO

Simulation often save money and time when it comes to testing a very new concept. Gazebo is such a simulation tools or platform where testing can be done for any physical object considering different force acting upon it. The real power of Gazebo is that it can mimic almost accurately all the aspect of physics in terms for dynamics, force, gravity etc. It can also create world where effect of surrounding environment can be simulated. For example, in this case effect of wave and wind on the water vessel. On top of that simulation can be controlled with many parameters such as method of iteration and/or iteration time, speed. Overall, Gazebo can handle the dynamics in such a details that usually no significant difference is experienced when the system is tested in real life.

2.4 ROS

Reading status and controlling robot was never easy because of communicating with many sensors at the same time. Often people used to spend lots of time only to prepare communication system for every robot project. Robot operating system was developed to overcome this problem and gave a common system which can be used in any robotic project[12]. It is an application that give full structure of communication. Basic structure of ROS is based on Topic and Noes. Topic holds place of value of sensor whether to read or write. Alongside Nodes helps to get or set the value on topic. ROS keeps and make sure the readiness of values in Topic (in high frequency) so control system or any function can read or write value reliably.

2.5 CasADi

Casadi is a tool library to manage variables for easily solving optimal control problem. Casadi works mainly as a place holder for variable so designer of code doesn’t have to concentrate on arranging those. Casadi also integrates popular solvers for example qpoas, ipopt etc. which makes it more dynamic and one stop solution provider.

3 Implementation

3.1 System overview

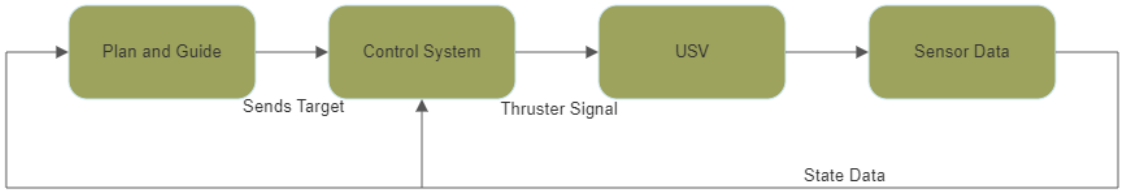


Figure 3-1 : Overview of System

Control system for this USV follows regular workflow of controlling. Sensors are read and data are fed to both Control system and Plan & Guide module. Control System receives updated goal from Plan & Guide module and try to reach that. Model Predictive Control (MPC) is used to reach latest target. Hence dynamic model of the system is embedded in Control System. Finally output from the Control System controls two thrusters to guide USV to reach its goal.

Two implementation is done for this work,

1. At first, model in implemented and simulated in python code and all the state data is generated from mathematical model iteration. Output is plotted on graph.
2. Secondly, GAZEBO is used to simulate USV and all state data are collected from GAZEBO sensors.

To collect data in GAZEBO, ROS is used and corresponding variables were updated other modules to fetch.

Some module and function are mocked and those can be developed in future projects. Following are those modules.

1. Using visual sensors, Lidar, camera and sonar, data will be analyzed to detect position, velocity and heading of obstacle.
2. Function to detect obstacle position (port/starboard side). Which is used to conclude whether obstacle is passed or not.

This project creates dummy or mock of those to focus on control problem as that is the scope of this work.

3.2 Collision Avoidance System

This paper deal with COLREG RULE 14, 15, 17: where obstacle coming from left should be given way and to cross vessels on port side when head to head situation occurs. Two type of obstacle is considered, one stationary and one moving body. Both are tested in separate scenario. Position, heading and velocity of the object is considered to be available accurately and control system act upon that.

3.3 Software Flow and logic

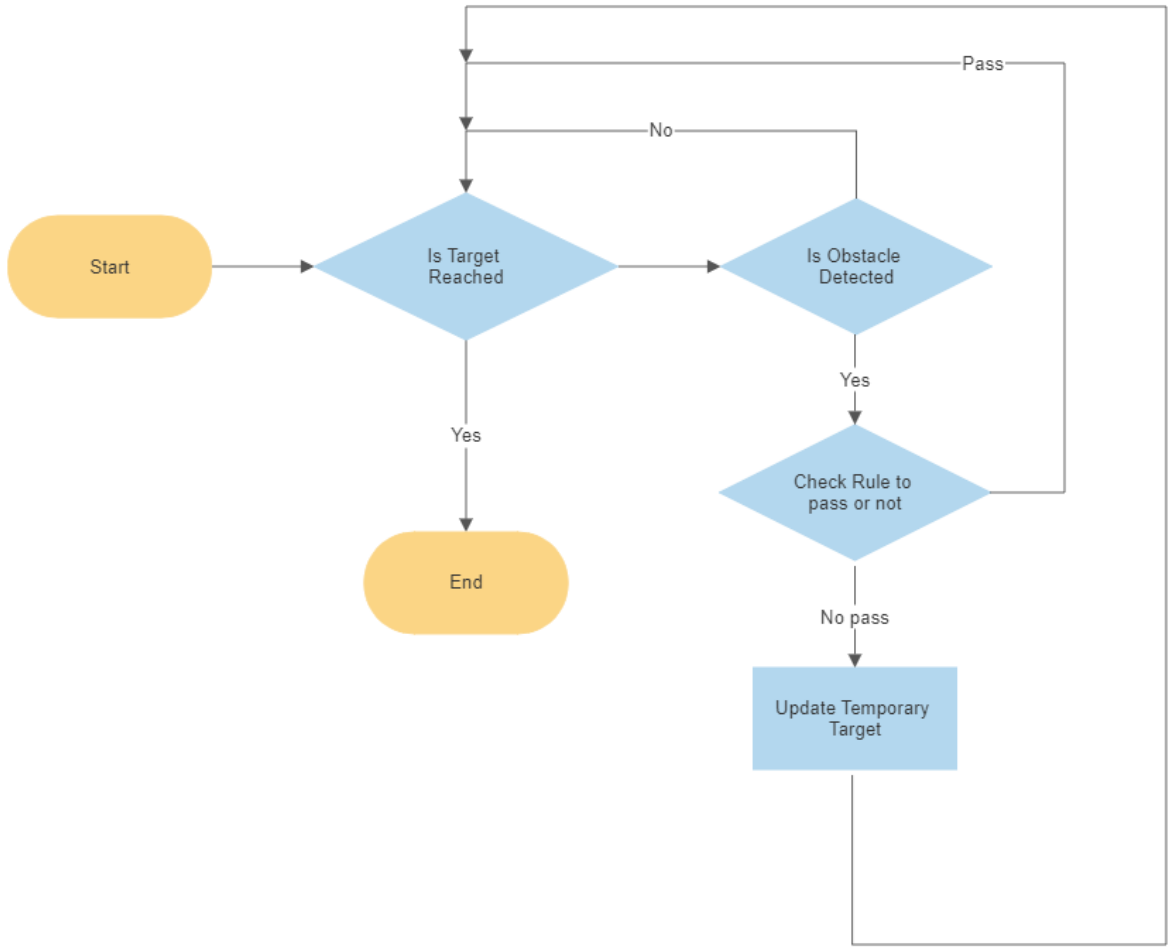


Figure 3-2 : Software Flow

As described in Figure 3-2 : Software Flow**isTargetReached** in the main loop with condition to exit when satisfactory level of error is reached between current state and target. **isObstacleDetected** and **checkRuleToPass** are mocks. **updateTemporaryTarget** is the function where logics are implemented to update target periodically based on the obstacle information. For example, there are two different scenarios dealt inside this function both stationary and moving obstacle. For stationary object target is updated once and for moving obstacle it is updated periodically till it reaches the position which is safe from obstacle and heading to final target.

3.4 Implementation of model

As kinetic model deals with forces, some parameters were necessary to set. Following are the parameters which were considered for the model. All parameters are aligned with the GAZEBO USV model, so later it can be used to simulate in GAZEBO. Parameters are collected from the XACRO files which are used as a parameters input file in GAZEBO USV model simulation.

PARAMETER	VALUE
MASS (M)	180 kg
INERTIA ($\boldsymbol{I_Z}$)	446 kg-m^2
$\boldsymbol{(X_u)}$	53.1 kg-s^{-1}
$\boldsymbol{(Y_v)}$	40 kg-s^{-1}
$\boldsymbol{(N_r)}$	446 N-mθ^{-1}
$\boldsymbol{(X_{\dot{u}})}$	0
$\boldsymbol{(Y_{\dot{v}})}$	0
$\boldsymbol{(N_{\dot{r}})}$	0
$\boldsymbol{(d_p)}$	2.4 m

Table 1: Kinetic model parameters

3.5 MPC design and implementation

Cost Function

Cost function is designed to minimize difference between target and current state as following.

$$min \sum_{k=0}^{N} ([f(y_i)]^T Q[f(y_i)] + [f(z_i)]^T R[f(z_i)])$$

3-1

Where, $f(y_i)$ is function of $[x \quad y \quad \psi \quad u \quad v \quad r]$ which are the sates of USV. And $f(z_i)$ is function of control inputs as $[X_{P1} \quad X_{P2}]$.

Q and *R*, are weight matrices to tune optimization. Point to note that weight on $x \quad y \quad \psi$ are emphasis and rest are set to zero as for this experiment main concern is to reach the goal in correct orientation.

Setting up MPC parameters was finalized by trial and error of different values for parameters. With step size of 0.1 sec and 10 prediction horizon was good enough to control USV while using mathematical simulation. For GAZEBO step size was chosen 1 because the simulation updat was every 1 sec for states.

To manage the solver and optimization variables, CasAdi [6] library is used. The non-linear equation was optimized using popular solver ipopt which was configured in CasAdi tools.

Sum of cost function for full prediction horizon was done using Raunge Kutta 4^{th} order method. Objective was to minimize the weight cost function for whole horizon.

Constrains

Constrains are set according to the physical limitation of USV, these values were also collected from the GAZEBO model of USV. But after that it was widened up to make it easy for solver to reach minima.

Table 2: Constrains

CONSTRAIN	**MIN**	**MAX**
Velocity in u direction	$-10ms^{-1}$	$10ms^{-1}$
Velocity in y direction	$-10ms^{-1}$	$10ms^{-1}$
Thruster Force Limit	-100N	250N
Angular Velocity	$-30\ \theta s^{-1}$	$30\ \theta s^{-1}$

3.6 GAZEBO code

USV simulation in GAZEBO runs with ROS. Out of many rostopics, only GPS, IMU and thruster topic was subscribed to get status data and to publish thruster command.

Figure 3-3: GAZEBO VRX model

GPS

Raw data of GPS (latitude and longitude) is converted to cartesian coordinated, ENU. That make it easy to work with position and target, as the movement of the experiment is done within very small range.

IMU

From the IMU sensor, velocity and orientation data was collected. Orientation data was converted from quaternion to Euler as model used Euler angle in rad. From Euler angle only rotation of z-axis is taken as the USV model consider rotation in only yaw.

Thruster topic

Control signal is continuously published in thruster topic (left and right two separate topics). UVS model in GAZEBO takes value only in range from 1 to -1. Therefore, in the code control signal is scaled down to this range before publishing in topic.

Control code

Control code in GAZEBO simulation is repetition of previous MPC loop. Only two parameters are changed and those are mass and inertia. The reason being, in the simulation extra components are being added in the USV such as lidar, GPS, camera, shooter etc.

4 Results

As mentioned earlier, this experiment is simulated in two different ways. First the code is ran in plain python and movement path is plotted. Parallel to MPC, USV mathematical model is also ran to get state values.

In second experiment GAZEBO is used to simulate USV model. Some target is given and movement path is collected for plotting graph along with video capture of GAZEBO simulation.

Do same test is done in both way.

4.1 First experiment using mathematical model

4.1.1 No obstacle (Mathematical Simulation)

	Initial State	Target	Result
x	0	30	30.13
y	0	40	40.37
ψ	90°	270°	259°

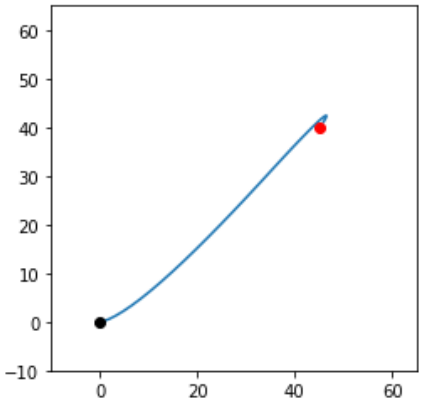


Figure 4-1: No obstacle test 1

	Initial State	Target	Result
x	0	0	-0.34
y	0	40	39.99
ψ	90°	90°	89.9°

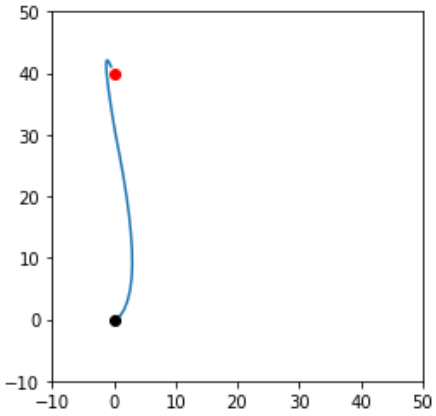


Figure 4-2: No obstacle test 2

These tests indicates that designed controller is capable to reach its target.

4.1.2 With Obstacle

Weight: 180, Inertia: 446

	Initial State	Target	Obstacle
x	0	45	15
y	0	40	10
ψ	0°	45°	0

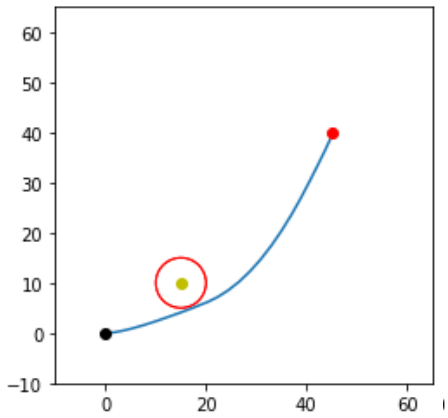


Figure 4-3: Stationary Obstacle USV 180kg

This test is done by keeping COLREG RULE 17 in mind; which define Actions by stand-on vessel: It states that if possible, course should be kept as it is and if necessary USV should pass the obstacle by keeping obstacle on USV's own port side.

Weight: 250, Inertia: 495

	Initial State	Target	Obstacle
x	0	45	15
y	0	40	10
ψ	0°	45°	0

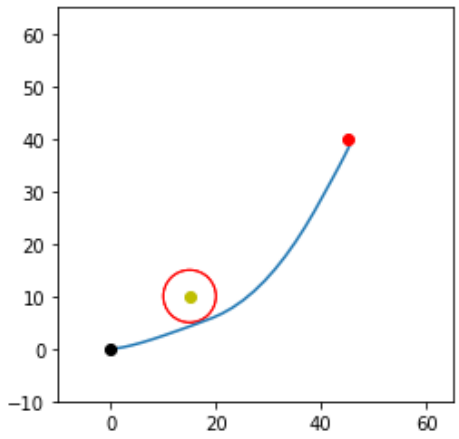


Figure 4-4: Stationary Obstacle USV 250kg

This test demonstrate that change of weight and inertia did not have much effect on control.

	Initial State	Target	Obstacles
x	0	70	(15,15) (40,40) (65,65)
y	0	80	
ψ	0°	45°	

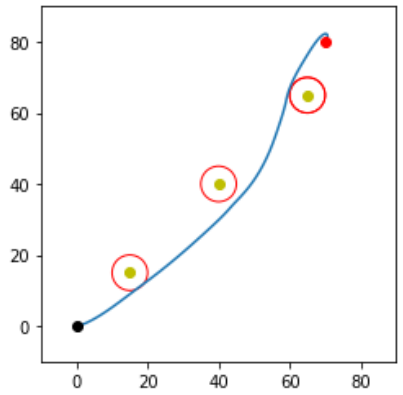


Figure 4-5: Multiple stationary obstacle

4.1.3 Moving obstacle (Mathematical Simulation)

	Initial State	Target
x	0	45
y	0	40
ψ	0°	45°

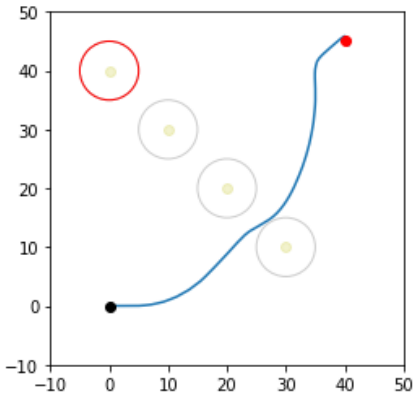
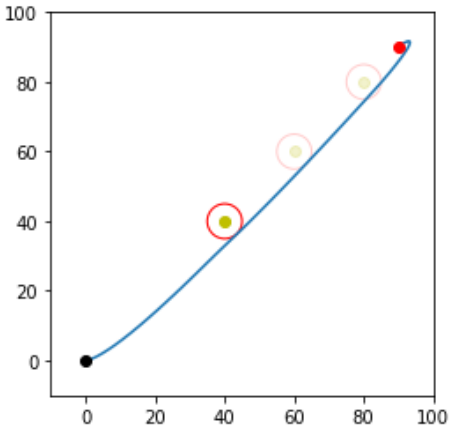


Figure 4-6: Moving obstacle

In this test, movement of obstacle is simulated from [30,10],[20,20],[10,30],[0,40]. USV travelling upto point (10,0) chases target [30,10]. At this moment it is considered that updated target is provided by trajectory module(not implemented in this project) based on latest movement of obstacle. From this point USV starts chasing new target which is [20,20]. At (20,15) target is being updated to ultimate target as it is already detected that obstacle is crossed or on the lift hand side(not implemented in this project). This testing and simulation is done considering COLREG RULE 15. Which states that crossing vessel should be given way if the vessel is coming from own starboard side.

	Initial State	Target
x	0	90
y	0	90
ψ	0°	45°



This test is done to simulate COLREG RULE 14: which states that in head to head situation vessels should pass each other on port side.

4.2 GAZEBO TEST

Similar test above has been done in GAZEBO simulation using VRX model [9]. As mentioned before, weight and inertia on z-axis is increased to match with VRX model.

4.2.1 No Obstacle (GAZEBO Simulation)

	Initial State	Target
x	0	7
y	0	60
ψ	45°	90°

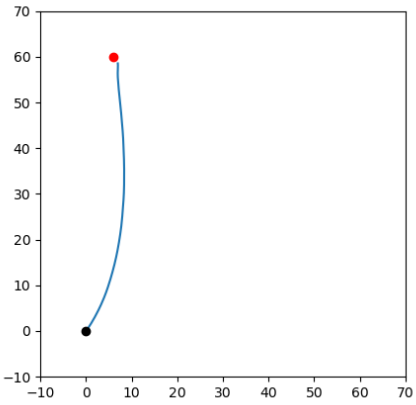


Figure 4-7: GAZEBO path No obstacle

4.2.2 Stationary Obstacle (GAZEBO Simulation)

	Initial State	Target
x	0	7
y	0	60
ψ	45°	90°

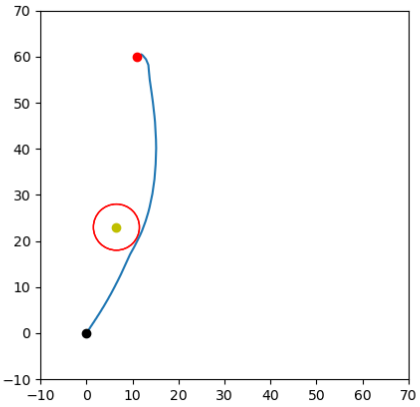


Figure 4-8: Crossing obstacle in GAZEBO sim

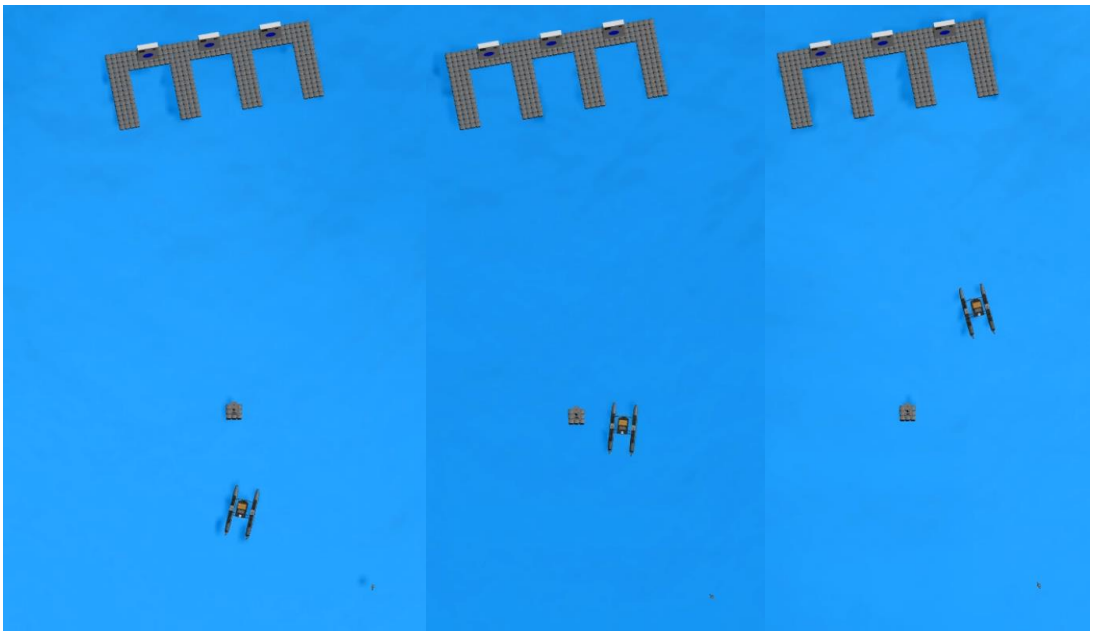


Figure 4-9: GAZEBO visual result

Animation of simulation is shared via this link in youtube.

https://youtu.be/JVyHBpnyh0c

Comparison of same route with and without obstacle is shown in this youtube video.

https://youtube.com/shorts/cavdPzeXrUI

5 Discussion

Objective of this work was to automate USV to reach its goal using model predictive control. And on the way it has to avoid collision with obstacle. The obstacle can be stationary or moving.

Implementation of control system was done in three steps. At first, using only kinematic model, control was achieved without much error (No plot is shown in result). At second stage kinetic model was added, in which control signal was shifted from velocity (u, v) to force (X_{P1} , X_{P2}), in other word control input changed. Some weight and inertia value, i.e weight, inertia etc, were set to simulate the mathematical model.

Testing without any obstacle, to reach goal was very precise and accurate. There were error at level ~0.1% when the target orientation was set to odd position. For example, in no obstacle scenario target orientation was set to 270° but x and y target was same. In test 2, target for vertical movement also was achieved without any difficulties. But overshooting was observed when goal was reached.

When obstacle is placed on the way, shortest path was not achieved. The reason is the weight (Q matrix) on the orientation parameter. Weight (Q matrix) on the orientation is set to double compared to axis-x, y . The reason behind emphasizing orientation is to keep the heading correct, as USV is not capable to move only on Y direction heading to the target keeps the USV to move in correct direction.

Similar behavior is also observed in moving obstacle. When USV has just crossed the moving obstacle, it slightly turned clockwise to keep up with target orientation (Which is not valid when reaching obstacle i.e. temporary target). It would be more appropriate to keep on changing orientation along with x, y target.

For multiple stationary objects, continuously update of target seems to work fine. Although in this test, periodically target is being updated in mock which is expected to happen from visual sensor module in future project. Note can be taken that if multiple stationary objects are detectable at early stage or as far as possible to detect, optimized trajectory can be planned.

Expected behavior is also observed in GAZEBO simulation. Despite of having wind and wave, controller was able to reach target properly.

6 Future Work

As highlighted in result that detecting obstacle position was mocked in this thesis. On of the very immediate future work can be to implement those function which will be feeding the planning module with information of obstacle.

Another future work of this project can be to implement more COLREG RULEs. For example based on the size of obstacle or by analyzing image, if it can detect whether the obstacle is motor operate or not then few other COLREG RULEs can be immediately implemented.

7 Conclusion

In this thesis implementation of MPC to avoid collision has been successfully tested. Well stablish simplified 3DOF model of USV was stable and was also able to predict states of USV which was fed into MPC to optimize for control signal.

Apart from plotting graphs with mathematical simulation of system, implementation in GAZEBO was also done. MPC controller seems to work well in GAZEBO which increases the chance to run error free in physical USV.

Software architecture of the system is done such a way that in future same architecture can be use to further implement the COLREG rules. As MPC module is separate there will be no hassle to introduce new functionality.

References

[1] B. Olav, “MPC-based Mid-level Collision Avoidance for ASVs using Nonlinear Programming.”

[2] L. Chunyue, “Modeling and Experimental Testing of an Unmanned Surface Vehicle with Rudderless Double Thrusters,” Tianjin University, China, 2019.

[3] F. Andrade, “Autonomous UAV surveillance of a ship’s path with MPC for Maritime Situational Awareness.”

[4] T. Johansen, “Ship Collision Avoidance and COLREGS Compliance using Simulation-Based Control Behavior Selection with Predictive Hazard Assessment.”

[5] I. Hagen, “Collision Avoidance for ASVs Using Model Predictive Control,” NTNU, 2017.

[6] A. Joel A E, “{CasADi} -- {A} software framework for nonlinear optimization and optimal control,” *Math. Program. Comput.*, vol. 11, pp. 1–36, 2019.

[7] F. Thor, *Handbook of Marine Craft Hydrodynamics and Motion Control*, 1st ed. West Sussex, UK: Jhon Wiley & Sons Ltd, 2011.

[8] F. Thor, “Nonlinear modelling and control of underwater vehicles,” NTNU, 1991.

[9] B. Bingham, “Toward Maritime Robotic Simulation in Gazebo,” Seattle, WA, 2019.

[10] “Openrobotics.” https://www.openrobotics.org/

[11] R. Sharma, *Mpc Lecture*. [Mp4]. Available: https://web01.usn.no/~roshans/mpc/videos/lecture1/introduction.mp4

[12] “ROS.” https://www.ros.org/

[13] “COLREGs - convention on the international regulations for preventing collisions at sea, international maritime organization (IMO),.” 1972.

Appendices

Appendix A

Snippet of MPC code

```
def chaseTarget():
    global args, state_init, state_target, n_states,\
        n_controls, N, f, solver, nlp_prob, mpc_iter, u0,\
        X0, t0, t, cat_states, cat_controls, cont_XP1,\
        cont_XP2, times
    t1 = time()

    args['p'] = ca.vertcat(
        state_init,    # current state
        state_target   # target state
    )

    args['x0'] = np.zeros(n_states*(N+1) + n_controls*N)
    # print(args['p'])

    sol = solver(
        x0=args['x0'],
        lbx=args['lbx'],
        ubx=args['ubx'],
        lbg=args['lbg'],
        ubg=args['ubg'],
        p=args['p']
    )

    u = ca.reshape(sol['x'][n_states * (N + 1):], n_controls, N)
    X0 = ca.reshape(sol['x'][: n_states * (N+1)], n_states, N+1)


    print(DM2Arr(X0)[0,0], DM2Arr(X0)[1,0], DM2Arr(X0)[2,0])
    # cont_XP1 =
    cont_XP1 = np.vstack((cont_XP1,DM2Arr(u[0, 0])))
    cont_XP2 = np.vstack((cont_XP2,DM2Arr(u[1, 0])))


    t0, state_init, u0 = shift_timestep(step_horizon, t0, state_init,
u, f)
```

Appendix B

Snippet of ROS code

```
def gpsCallback(data):
    # print(data.latitude, data.longitude)
    global x_init, y_init, z_init,\
         init_counter,latitude_ori, longitude_ori, altitude_ori

    latitude = data.latitude
    longitude = data.longitude
    altitude = data.altitude
    #To capture initial lat long for ENU reference
    #Executes only once at beganing
    if init_counter:
        latitude_ori = latitude
        longitude_ori = longitude
        altitude_ori = altitude
        init_counter = False

    x_init_temp, y_init_temp, z_init_temp = pm.geodetic2enu(latitude,
longitude,\
                            altitude, latitude_ori, longitude_ori,
altitude_ori)
    x_init = x_init_temp
    y_init = y_init_temp
    z_init = z_init_temp

def imuCallback(data):
    global r_init, theta_init
    (roll,pitch,yaw) = euler_from_quaternion([data.orientation.x,\
        data.orientation.y,data.orientation.z,data.orientation.w])
    theta_init = yaw
    # print(theta_init_copy)
    r_init_temp = data.angular_velocity.z
    r_init = r_init_temp

def fixVelCallback(data):
    # print(data)
    global u_init, v_init
    u_init_temp = data.vector.x
    v_init_temp = data.vector.y
    u_init = u_init_temp
    v_init = v_init_temp
```

Full code github link

https://github.com/samiusn/Masters-Thesis

Appendix C

COLREG RULES

This section provides a brief overview of the main technical and operational requirements from COLREGS, [13]:

Rule 6 - Safe speed. The following should be considered: Visibility, traffic density, stopping distance and turning ability, wind/waves/current, navigational hazards, draught vs. depth, radar/sensor state.

Rule 8 - Actions to avoid collision. Actions shall be made in ample time. If there is sufficient sea-room, alteration of course alone may be most effective. Safe distance required. Reduce speed, stop or reverse if necessary. Action by the ship is required if there is risk of collision, also when the ship has right-of-way.

Rule 13 - Overtaking. Any vessel overtaking any other shall keep out of the way of the vessel being overtaken. A vessel shall be deemed to be overtaking when coming up with another vessel from a direction more than 22.5 degrees abaft her beam.

Rule 14 - Head-on situation. When two power-driven vessels are meeting on nearly courses so as to involve risk for collision, then alter course to starboard so that each pass on the port side of each other.

Rule 15 - Crossing situation. When two power-driven vessels are crossing so as to involve risk of collision, the vessel which has the other on her own starboard side shall keep out of the way.

Rule 16 - Actions by give-way vessel. Take early and substantial action to keep well clear.

Rule 17 - Actions by stand-on vessel. Keep course and speed (be predictable) if possible. If it is necessary to take action, then the ship should try to avoid to alter course to port for a vessel on her own port side.

Rule 18 - Responsibilities between vessels. Except for Rules 9, 10, and 13, a power-driven vessel shall keep out of the way of: a vessel not under command, a vessel restricted in her ability to manoeuvre, a vessel engaged in fishing, and a sailing vessel.

Rule 19 - Conduct of vessels in restricted visibility. Avoid alteration of course to port for a vessel forward of the beam, and avoid alteration of course towards a vessel abeam or abaft the beam, if possible.

University of South-Eastern Norway

Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: SeaDrone trajectory following with model predictive control using robot operating system

USN supervisor: Fabio Andrade / Carlos Pfeiffer

External partner: N/A

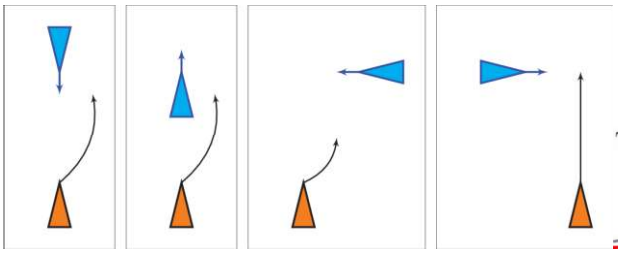
Task background:

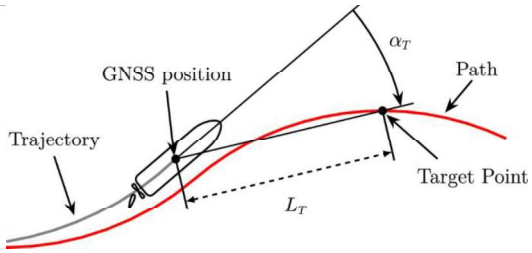
The Autonomous Systems Research Group has a SeaDrone which is used for research (Valkyrie project) and competitions (autodrone.no). Some tasks involve collision avoidance, path planning, path following, remote sensing among other methods.
The Valkyrie project is an EU funded project with many partners about fast response in post disasters scenarios. USN's task in the project is to develop autonomous solutions to help recon, search and rescue. Such missions rely on path following algorithms.
In the Autodrone competition, as the SeaDrone is meant to perform its tasks autonomously, the path following is present in most tasks.

Currently, a simple waypoint navigation is implemented for path following and more advanced solutions are needed.

The platform currently has a Nvidia Jetson Xavier, which is responsible for processing some of the sensor data and a Raspberry Pi, which is connected to the control unit and is responsible for the navigation. The Software solution is built with Robot Operating System.

Task description:
The objective of the project is to develop and test a real time path following solution for the SeaDrone using Model Predictive Control. The model should account for the kinematic model and constraints of the SeaDrone. Cost functions should be developed for the main tasks, such as collision avoidance, where the SeaDrone should change its course with respect to a moving boat that is being tracked.





Student category: IIA

Is the task suitable for online students (not present at the campus)? Yes

Practical arrangements:
The SeaDrone is located in campus Vestfold. The solution can be tested using the SeaDrone or in simulation tools such as Gazebo or Unreal Engine if the water test is not possible due to logistical challenges.

Supervision:
As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

Signatures:

Supervisor (date and signature):
10.11.2021

Student (write clearly in all capitalized letters): SYED SAMI AL HAQ

Student (date and signature): 30/5/2022