BRAUDE
College of Engineering, karmiel

Capstone Project Phase A

SeaSafe

Maritime Collision Avoidance Simulator With COLREG Compliance

Members

Guy Pariente

Almog Elbaz

Supervisors

Dr. Edi Shmueli

Prof. Miri Weiss Cohen

Introduction

Maritime transportation is essential to global trade, facilitating the movement of goods across oceans efficiently and reliably. It accounts for over 80% of the volume of international trade[12], making it a cornerstone of the global economy.
While maritime transportation addresses the growing demand for efficient trade, the safety of these operations remains a critical concern. Studies have shown that 80-85% of maritime incidents are attributed to human error [13]. As the volume of maritime traffic increases, the potential for such incidents rises, posing significant risks to human life.

One promising solution to enhance safety and efficiency is the development of autonomous shipping. Autonomous vessels have the potential to reduce human error by relying on advanced algorithms and sensors to make real-time decisions. These systems can consistently process massive amounts of data to navigate safely, even in complex scenarios involving multiple vessels. Autonomous vessels can operate continuously without the need for rest, optimize routes for fuel efficiency, and respond rapidly to changing conditions.

Despite these advantages, a critical challenge remains: ensuring that autonomous vessels can avoid collisions while complying with established maritime regulations. Central to maritime safety is the Convention on the International Regulations for Preventing Collisions at Sea (COLREG), established by the International Maritime Organization (IMO) in 1972. COLREG sets out the rules that all vessels must follow to prevent collisions, specifying how ships should behave in various situations such as head-on encounters, crossings, and overtaking maneuvers. It's important to note that COLREG was originally designed to address interactions between two vessels at a time. The rules primarily focus on 2-ships scenarios, providing clear guidance on how two ships should navigate to avoid collisions.

COLREG has many rules. In this work we focus on rules that apply to “power-driven vessels”. Fig. 1 shows the COLREG rules we focused on:

1. Overtaking – a ship is trailing another ship and attempts to overtake it.
2. Head on situation – ships are approaching each other head on, and each vessel will steer to its right.
3. Crossing situation – a ship that sees another vessel on its starboard side will turn right and pass behind the ship that was on its right.

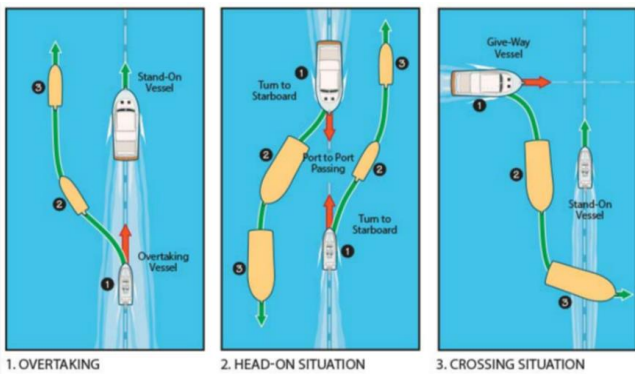


Fig. 1. Visual explanations for the 3 rules

COLREG also assigns roles to ships in these scenarios, designating one as the stand-on vessel (which should maintain its course and speed) and the other as the give-way vessel (which must take clear action to avoid collision).
On Head-on scenario - both vessels are considered “give-way” ships, on crossing scenario, the ship that has the other on its starboard (right) side is the “give-way” ship, the other ship is “stand-on” one.
In the Overtaking scenario, the ship overtaking another is the “give-way” ship and must keep out of the way of the ship being overtaken, which is the “stand-on” ship.

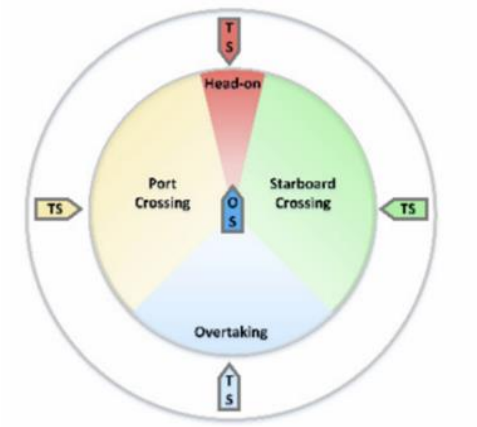


Fig 2. - Port and Starboard sides visualization, from [7]

This project aims to develop an algorithm for collision avoidance that ensures autonomous vessels operate in full compliance with COLREG. By focusing on the rules applicable to power-driven vessels, we seek to address the complexities involved in real-time decision-making in maritime environments. An effective collision avoidance algorithm must not only detect and predict potential collision scenarios but also decide on maneuvers that are compliant with COLREG. This involves complex decision-making processes, especially when multiple scenarios may need to be addressed simultaneously and for the reason that COLREG is well defined for 2 ships scenarios, and we may encounter multi-vessel collision scenarios.

To test and validate our algorithm, we plan to develop a simulator that models various maritime scenarios involving multiple autonomous vessels. The simulator will allow us to create controlled environments where we can examine how the algorithm performs under different conditions. By simulating these scenarios, we aim to assess the algorithm's effectiveness in ensuring safe and efficient navigation while maintaining full compliance with maritime regulations. To guide the development process and ensure that the simulator and algorithm meet our objectives, specific requirements have been established outlining the functional and non-functional aspects of the project. These requirements will be detailed further in a later section.

Over the years, numerous papers have discussed ship collision avoidance and proposed various solutions and predictive methods, In reviewing existing literature, we found several approaches that have been demonstrated in the design of collision avoidance systems, with a

variety of algorithms being considered such as Ant Colony Algorithm (Tsou, Ming-Cheng and Hsueh (2010)) [2], Evolutionary Algorithms(Roman and Zbigniew (2000)) [3], Bayes Nets (Yonghoon, Jungwook and Jinwhan (2018))[4], DCOP - Distributed coordination optimization (Rudy, Shijie and Jialun (2019))[5], ETAD - Extended timeframe for AI decisions (Andreas,Magne and Ole (2022))[6]. We will expand on these approaches in greater detail in the Literature Review section.

One of the major challenges for autonomous shipping solutions is the need to operate in coexistence with conventional manned ships. This is a problem because manually steering adheres to the captain's experience and instantaneous decisions. In our project we assume a fully autonomous environment – all vessels are autonomous, which we believe is the future.

We can divide the goals of our project into practical and theoretical ones. From a practical (SW development) perspective, we aim to have the simulator running with the algorithm we will develop. Theoretically, we have defined an objective function to finish the scenarios as fast as possible (last vessel to reach its destination). This allows us to examine the scalability of the algorithm by testing how the finish time is affected by the number of vessels in the scenarios.

An ideal algorithm should be able to handle multiple “nested” situations - that is, solve collisions that arise from previous moves made by the algorithm to avoid collisions. This is really taking COLREG to the extreme. But there could be cases the algorithm cannot handle. Identifying those, and proposing solutions is part of the research.

Literature Review

Maritime collision avoidance has been the subject of intensive research over the last decade. Solutions range from genetic algorithms (2000) to use of deep learning and RL in recent years. The solutions vary in the problem they intend to solve, and assumptions they take on the environment and rules that should be complied.

Negenborn et al.[5] proposed a solution based on a distributed coordination mechanism with dynamic calculation of collision risk parameters using accurate maneuverability models. The paper defines the problem as a ‘many to many’ situation - where the objective is to optimize all the trajectories of the involved ships, and not using a point of view of one ship only, Fig .3 shows the ‘many-to-many’ situation described.

The proposed solution involves two phases as shown in Fig. 4: In the first phase, predictions of ship trajectories are made based on ship dynamics, considering different candidate rudder angles, and potential collision risks that may be caused by each rudder angle selection are

evaluated based on calculations of collision risk parameters; secondly, an optimization strategy is adopted to find the most efficient collision avoidance plan for the ships, namely, the rudder angles that each ship should take, and the corresponding operation time for rudder steering, with the overall objective to minimize the sum of time that each ship spends in avoiding collisions with the other ships. The suggested trajectories don't apply COLREG rules.

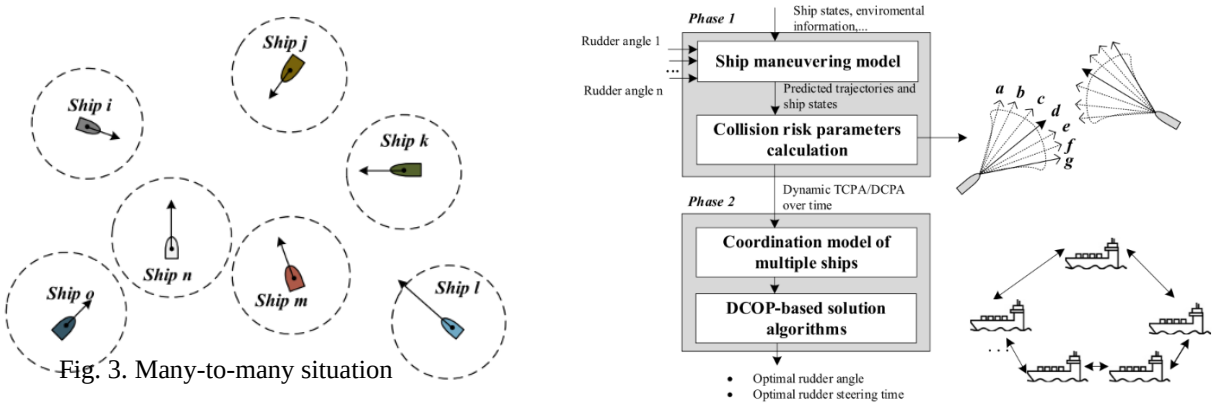


Fig. 3. Many-to-many situation

In order to assess and analyze the effectiveness of the proposed method, simulation experiments are carried out, demonstrating that the method minimizes collision risks and optimizes ship trajectories. The results show that the DPOP(Dynamic Programming Optimization Protocol) algorithm provides the shortest computation time, while AFB(Asynchronous Forward Bounding) requires the largest information exchange. SyncBB had the longest computation time.
However, since the proposed method optimizes trajectories without assigning specific roles to ships (e.g., "give-way" or "stand-on" vessels) as required by COLREGs, it does not guarantee compliance to these vital navigational rules, In addition, the authors mentions that more research should be taken in order to assess the compliance of the solution with COLREG.

Our project aims to address this gap by developing a collision avoidance system that explicitly follows the regulations set forth by COLREGs.

In the study by Tsou et al. [2], the authors proposed a collision avoidance route planning model based on the Ant Colony Algorithm (ACA) within an e-navigation framework. The Ant Colony Algorithm is inspired by the foraging behavior of ants, which find the shortest paths between their colony and food sources by laying down pheromones. In the context of ship navigation, the algorithm simulates a number of virtual 'ants' that explore possible routes, favoring paths that minimize risk and distance. Under this concept, onboard navigation systems utilize the integration of the ship's own sensors, navigational practices, maritime regulations (COLREGs), and real-time data from the Automatic Identification System (AIS) to optimize collision avoidance paths.

The main problem addressed at the paper, is the complexity of decision-making for ship collision avoidance due to human error and information overload. The experimental results showed that the ACA-based method effectively minimizes collision risks and optimizes routes by ensuring safe and economical navigation, when compared to traditional methods, the proposed solution achieved improved efficiency and reliability.

However, while the method supports single-to-many and many-to-many collision avoidance scenarios, it does not fully comply with COLREGs. As stated in the article, the constrained conditions for the objective function are minimizing total distance and collision risk, those are optimization goals rather than ensuring the encounter rules outlined in COLREGs are followed.

Cho et al. [4] focused on improving automatic ship collision avoidance by addressing the prediction of other ships' maneuvering intentions. The proposed method provides a decision-making procedure for safe navigation by predicting the likely actions of nearby ships. To achieve this, they construct a graphical model using a Dynamic Bayesian Network (DBN), which combines intentional behavior patterns with probabilistic reasoning.

The Dynamic Bayesian Network allows the system to update its beliefs about other ships' intentions over time, based on observed data such as speed, heading, and position. By calculating the maneuvering intent probability, the system can anticipate potential collision scenarios and take appropriate action in advance.

Simulation experiments were conducted to verify the feasibility of this approach. The results indicated that the model accurately predicted the maneuvering intent of other ships, allowing for timely and safe navigation decisions. Fig 5. shows the simulations that were taken to verify the approach.

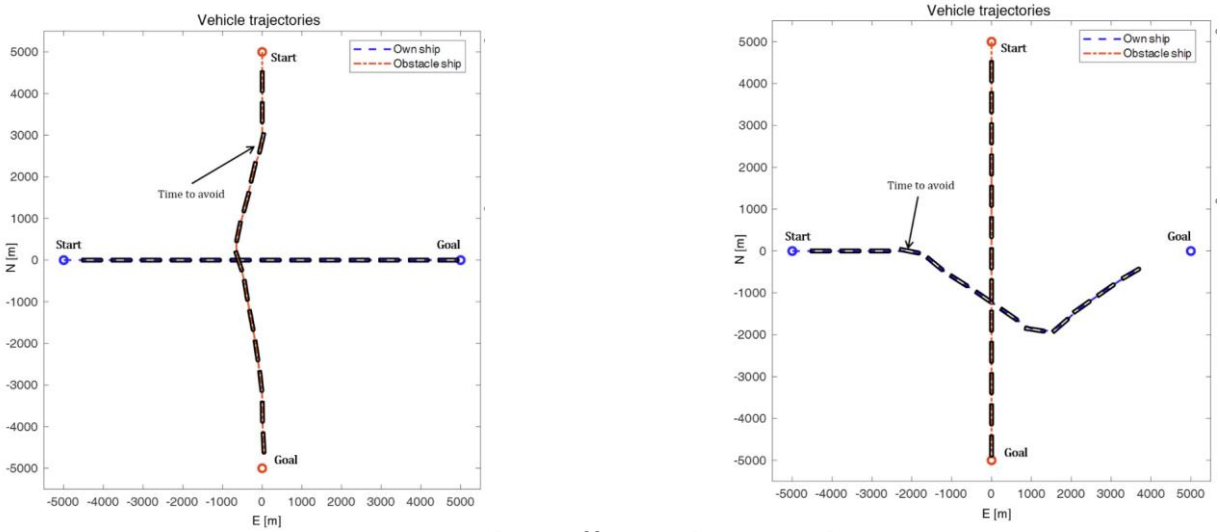


Fig 5. ship traffic simulation results

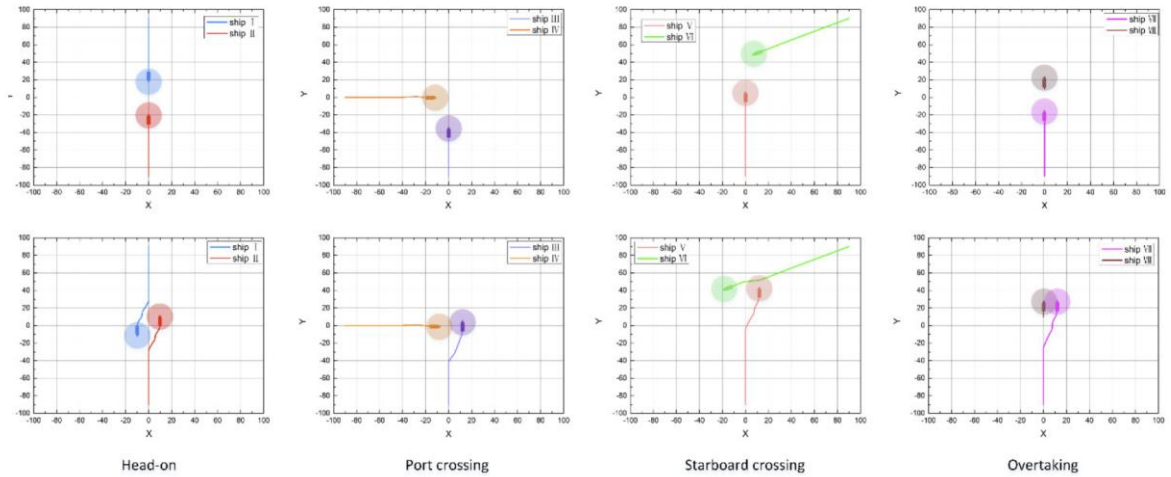
While the article presents a non COLREG-compliant solution, it primarily focuses on scenarios where one ship does not comply with the rules. We assume in our project that all ships are autonomous, controlled by the same algorithm, and therefore predicting intent is not an issue in our scenario.

Michalewicz et al. [3] proposed an Evolutionary Planner/Navigator (EP/N++) system to optimize ship trajectories in collision situations. The EP/N++ system uses evolutionary algorithms, which are computational methods inspired by natural selection and genetics, to search for near-optimal solutions in complex problem spaces. The method incorporates both dynamic constraints (such as the ship's current speed and heading) and static constraints (such as fixed obstacles) to generate safe, optimal paths for ships.

The simulation results showed that the EP/N++ system can effectively manage complex navigational scenarios and improve safety and efficiency compared to traditional methods. Anyway, the paper does suggest a route planning method, but doesn't comply with any COLREG rule.

The study presented by Wei and Kuo [7] provides a COLREG-compliant method to solve multi-ship collision avoidance problems based on the Multi-Agent Reinforcement Learning (MARL) algorithm.
The method proposed is based on 3 stages: Firstly, ship maneuverability and COLREG are taken into account for achieving multi-ship collision avoidance, after that, collision detection is done using Optimal Reciprocal Collision Avoidance (ORCA) method, which provides each ship a set of safe velocities that avoid collision, and calculate velocity obstacles of nearby ships, the study uses CTDE(Centralized Training Decentralized Execution) framework and DEC-POMDP(Decentralized partially observable Markov decision process) model to train the agents to avoid collision, then, multi-ship model was trained on situations based on CA-QMIX algorithm.

Simulation were performed to test the solution in a two-vessel, three vessel and four vessel scenarios, however, simulations indicate that while the algorithm avoid collisions, it is not truly COLREG compliant, even in two-vessel scenario the give-way is done by the port side crossing which should in fact keep its course and speed (Fig 6.). Also, the maneuvers seem hesitative, with multiple small corrections to course angle and speed.



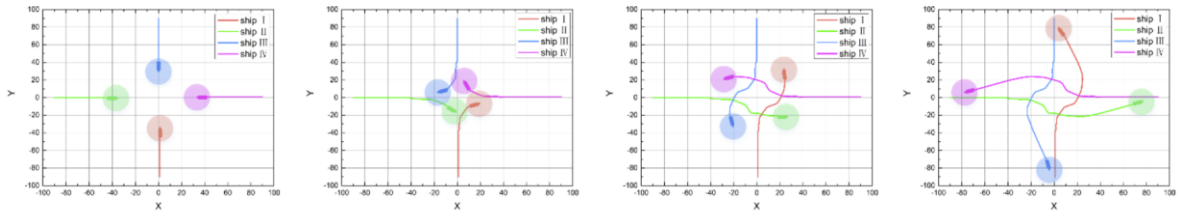


Fig 6. Two-vessel and four-vessel scenarios solution

Lyu and Yin [8] proposed a real-time path-planning method for autonomous ships using a modified Artificial Potential Field (APF) to address collision avoidance in dynamic maritime environments. The Artificial Potential Field method treats the ship as a particle moving under the influence of virtual forces: an attractive force pulling it toward the destination, and repulsive forces pushing it away from obstacles and other ships.
To comply with COLREGs and handle both moving targets and static obstacles, the authors enhanced the traditional APF approach by introducing subdivided zones around the ship to categorize potential collision risks. These zones allow the ship to assess the level of threat posed by nearby objects and adjust its behavior accordingly.

Their simulation results (Fig .7) demonstrated that the modified APF method successfully navigates multi-vessel encounters in real-time, considering unpredictable maneuvers of other vessels. Unlike other methods, this approach explicitly incorporates COLREG-compliant behaviors into the decision-making process.

While the modified APF method effectively addresses real-time path planning and collision avoidance with COLREG compliance, it has limitations. Notably, the algorithm is not a distributed solution, it is designed solely from the perspective of a single autonomous ship (Own Ship, OS). This means that the other Target Ships (TS) do not apply the algorithm or coordinate their maneuvers with the OS.

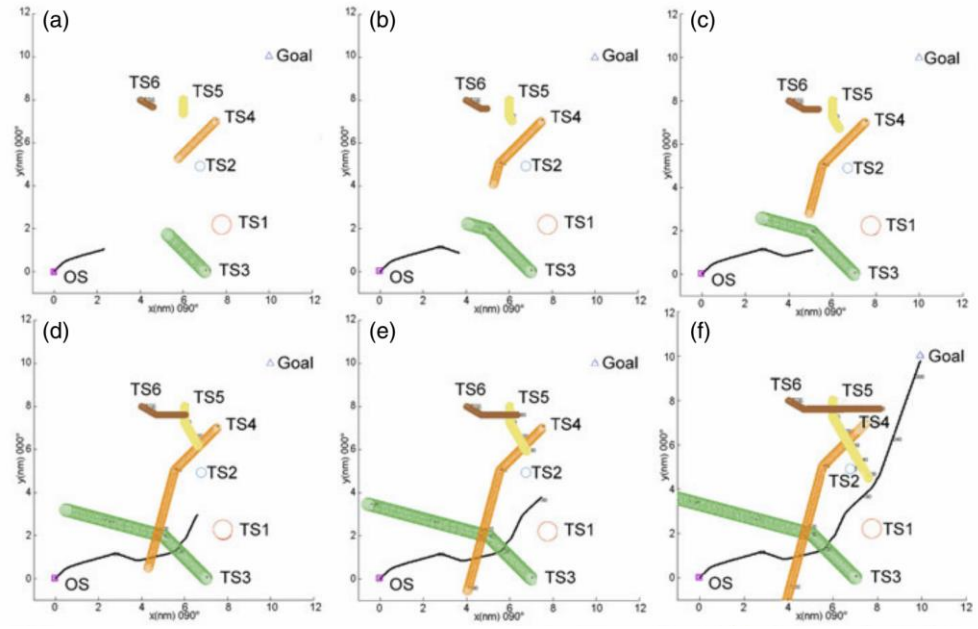


Fig 7. Simulation results using APF

Despite the extensive research and the variety of solutions proposed for autonomous ship collision avoidance, there remains a significant gap in achieving full compliance with COLREGs in multi-vessel scenarios.
Many existing methods, as discussed, either focus on two-vessel situations or do not entirely integrate COLREG rules into their algorithms. Some solutions optimize for collision avoidance and efficiency but overlook the explicit assignment of roles, which is a crucial aspect of COLREG. Others assume that not all ships will follow the regulations, leading to approaches that may not be suitable when all vessels are autonomous and compliant.

Given these limitations, there is a clear need for a new solution that ensures a full compliance with COLREG, operates effectively in multi-vessel environments, and allows all ships to coordinate their actions through shared compliance to the rules. Our project aims to address this need by developing a collision avoidance system that fills this critical gap, providing a fully COLREG-compliant algorithm suitable for autonomous maritime navigation in complex scenarios.

Proposed Solution

Problem Definition

Before addressing potential solutions, we define our problem as a search problem [10]. A search problem in the field of artificial intelligence involves finding a sequence of actions that transitions an agent(an entity that perceives its environment and acts upon it) from an initial state to a desired goal state within a defined state space. This framework is widely utilized for solving various decision-making and optimization tasks. The key components of a search problem include:

- State: A representation of the environment at a specific point in time.
- State Space: The set of all possible states that can be reached from the initial state through a sequence of actions.
- Initial State: The starting configuration of the environment.
- Actions: The possible maneuvers or decisions that can alter the state.
- Goal State: The desired configuration of the environment that satisfies the objective.
- Cost Function: A function that assigns a numerical cost to each state or action, guiding the search towards optimal solutions.

Our environment is modeled as an N×N open sea map with no obstacles except for the ships themselves. Each ship is defined by the following properties:

- Source and destination points: The starting and ending locations of the ship.
- Current position: The ship's position on the map at a specific time.
- Heading: The current direction the ship is moving.
- Maximum speed: The upper limit of the ship’s velocity.
- Current speed: The ship's speed at a given time.

- Role: The ship's COLREG-defined role (e.g., "give-way," "stand-on," or "maintain course").
- Physical dimensions: Configurable dimensions (e.g., width and length) of the ship.
- Status:
    - Green - no risk of collision
    - Orange - found collision in the future
    - Red - collision
    - ReachedDestination - whether reached destination or not.

Additionally, we have key configurable parameters:

- Safety zone: a circular area around the ships with configurable radius e.g., 50m.
- Horizon - the lookahead distance considered by the vessels when searching for potential collision and for collision avoidance maneuvering.
- Time step - An interval in the simulation during which the system updates the state of all ships.

At each time step, ships can take the following actions:

1. Adjust heading -10,0,+10 degrees.
2. Modify speed by -10,0,10 knots.

After setting up the environment, we can define the search problem with appliance to our problem:

1. State:
   A state represents a snapshot of the environment at a given time, including the positions, courses, speeds, and COLREG roles of all ships.
2. State Space:
   The state space encompasses all possible configurations of ship positions, courses, speeds, and roles over time.
3. Initial State:
   The initial state is defined by ships starting at their source points, with initial courses calculated to head towards their destinations.
4. Actions:
   At each time step, each ship can execute one of the following:
    - Change course (-10,0,10).
    - Adjust speed (-10,0,10).
5. Goal State:
   The goal state is achieved when all ships either:
    - Reach their destinations
    - Exit the horizon radius without collisions, all while adhering to COLREG.

Solution Approach: Tree Search Framework

Tree search is a fundamental method for exploring possible solutions in a structured problem space. It represents the problem as a tree, where nodes correspond to states and edges represent actions leading to new states.To solve this search problem, we propose modeling the decision-making process as a tree search:

1. Nodes in the Search Tree: Each node represents a state.
2. Edges: Each edge corresponds to an action taken by all ships.
3. Root Node: The initial state.
4. Goal Nodes: States where all ships have reached their goal state safely or reached their horizon limit with no collision and adhering to COLREG.

Collision detection with tree search

From the initial state, states are generated where each ship is headed to their destinations directly, we call this path the “optimal path” because that’s the shortest way to reach the destination and we want to keep ships at this path. States are generated until a limit - which we called earlier “horizon”.
The horizon is defined as a Nautical miles distance, to convert it to a number which will limit the search, we will calculate it by:

$$\frac{Horizon}{Distance\ that\ each\ ship\ passes\ at\ every\ time\ step.}$$

In our problem, collision is defined as a case where two or more ships enter each other’s safety zone, at each time step, the algorithm will look until the end of the horizon to determine whether ships have entered each other's safety zone, or in a simpler way - whether there’s a safety zone overlap.

In case of a safety zone overlap is found in the horizon, the current timestep statuses of the ships involved will be changed to orange, and the algorithm should start looking for collision avoidance, if not found, the status of the ship will remain green. Fig 8. shows the process of collision detection.

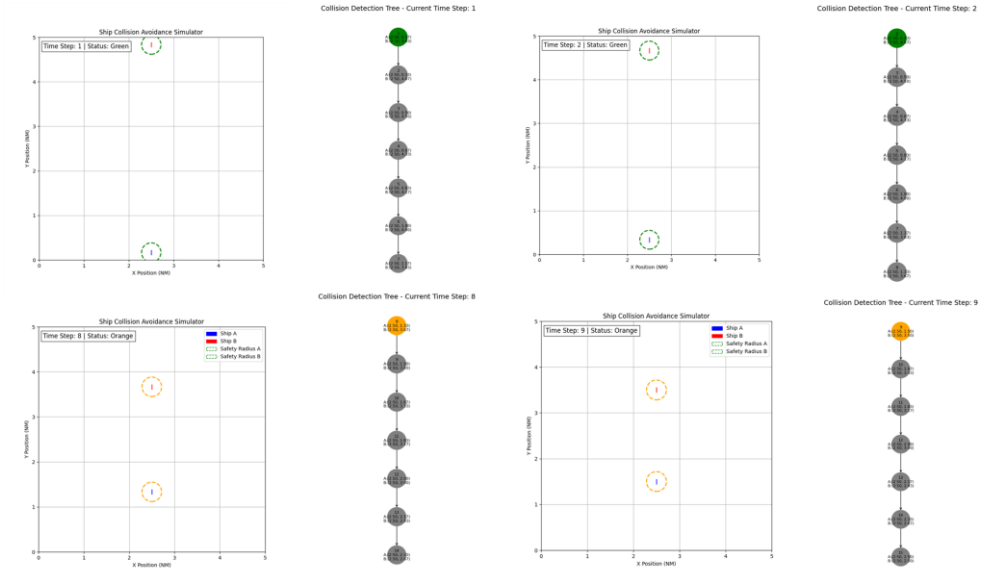


Fig 8. Collision detection process

Classifying Collision

After identifying a collision, we need to classify it in accordance to COLREG scenarios we expanded on in the introduction section. We can do that by calculating relative bearings of the ships involved in the collision. Relative bearings refer to the angle between the direction a ship is facing and the position of another object or ship.
We use the TAG-CSC (Temporal and Geometric COLREG Scenario Classification) system[11], where relative bearings are calculated to determine the positional relationship between vessels and to classify collision scenarios according to COLREGs. The process involves computing the angle between a ship's heading direction and the line-of-sight vector to another vessel. Specifically, for each ship, the system calculates the relative bearing by measuring the angle from the ship's current heading to the direction of the other ship's position. Fig 9. shows the visual representation of those angles.

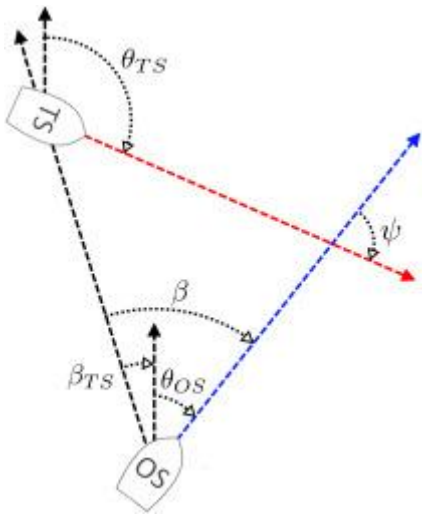


Fig 9. Visual representation of relative course ψ and relative bearing β between Own Ship (OS) and a Target Ship (TS)

After calculating those, we can classify the scenario according to Fig 10. and 11.

Scenario set	β_G boundaries (°)	ψ_G boundaries (°)
G_{HO}	[−45, 45)	[135, 180)
	[−45, 45)	[−180, −135)
G_{CGW}	[45, 180)	[−180, −64)
	[67.5, 112.5)	[−64, 0)
	[−30, 45)	[−135, −64)
G_{CSO}	[−180, −45)	[66, 180)
	[−112.5, −67.5)	[0, 66)
	[−45, 30)	[66, 135)
G_{OGW}	[−67.5, 67.5)	[−65, 66)
G_{OSO}	[−180, −112.5)	[−65, 66)
	[112.5, 180)	[−65, 66)

Fig 10. Classification of scenario based on β and ψ

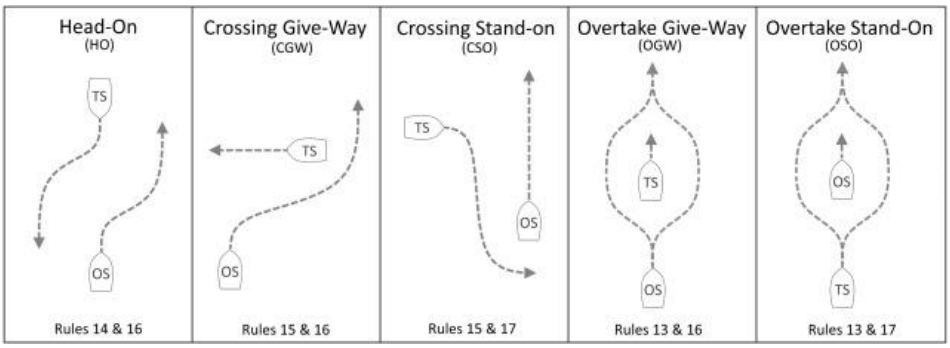


Fig 11. Scenarios and acronyms

Assigning Roles

At COLREG scenario, each ship is assigned with one of the following roles:

- Stand On
- Give Way

So after classifying the scenario, we would like to assign the correct role for each ship that's involved in the collision. TAG-CSC calculates relative bearings and analyzes the ships' headings to determine their positions relative to each other. Using this information, it identifies which vessel is on the starboard or port side (Fig 2.) of the other and assigns the roles accordingly.

Collision avoidance with tree search

So now we know the scenario type and the ships roles at the collision, we can start calculating what maneuver should be taken in order to avoid this collision with compliance to

COLREG, Fig 12. shows a visualization of a Head-on scenario and how it’s shown on the simulator.

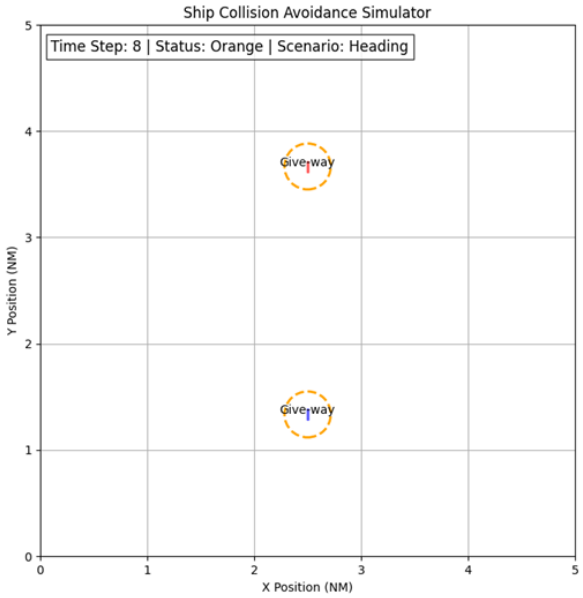


Fig 12. Visualization of collision detection for head-on scenario

As described earlier, “give-way” ships should maneuver to their starboard side, and “stand-on” ships should maintain course and speed.
So in order to avoid the collision, we start generating nodes that comply with the ship roles. Those nodes are generated from the state where the future collision was found.
So for example, in Fig 11. The algorithm found that in the horizon collision would happen, hence the status change to orange, the algorithm classified it as a head-on scenario and applied the ship's roles correctly. Because each ship is applied with a “give-way” role, the child nodes that will be generated will represent the next state where the ships turn starboard - as they should be as a “give-way” ships. Fig 13. provides visualization for this process

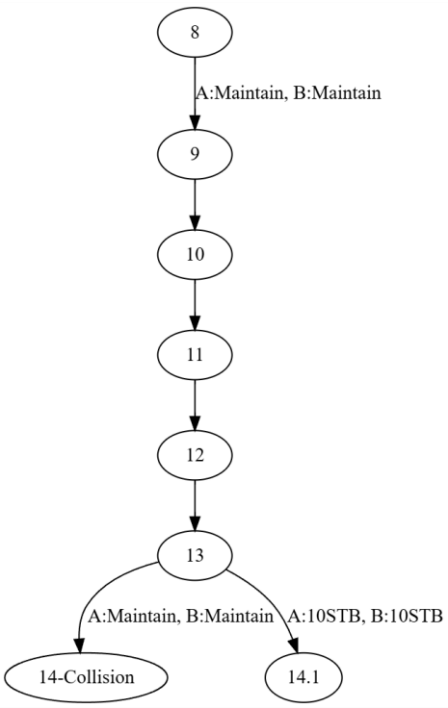


Fig 13. Process of generating child nodes

We aim to maneuver as late as possible, that means, that instead of generating child nodes from the collision detection node (node 8) we start generating from one state before the collision occurs.
So in our example we would start by generating state 14.1 - a state where both ships turn 10 degrees starboard.
This approach resembles a depth-first search (DFS), as it prioritizes delving deeper into the search space to test whether a maneuver resolves the collision before backtracking to explore other options.

By restricting the search to the horizon distance and generating nodes based on COLREG rules and collision-avoiding paths, we significantly reduce the size of the search space. In contrast, a brute-force approach would require generating the entire tree up until the horizon, with all actions being considered. Practically, from the initial state 81 child nodes will be generated, and from each child node another 81 child nodes will be generated, these amount of nodes will make the search infeasible, hence why we reduce the generated nodes only to the child nodes that will achieve our goals.

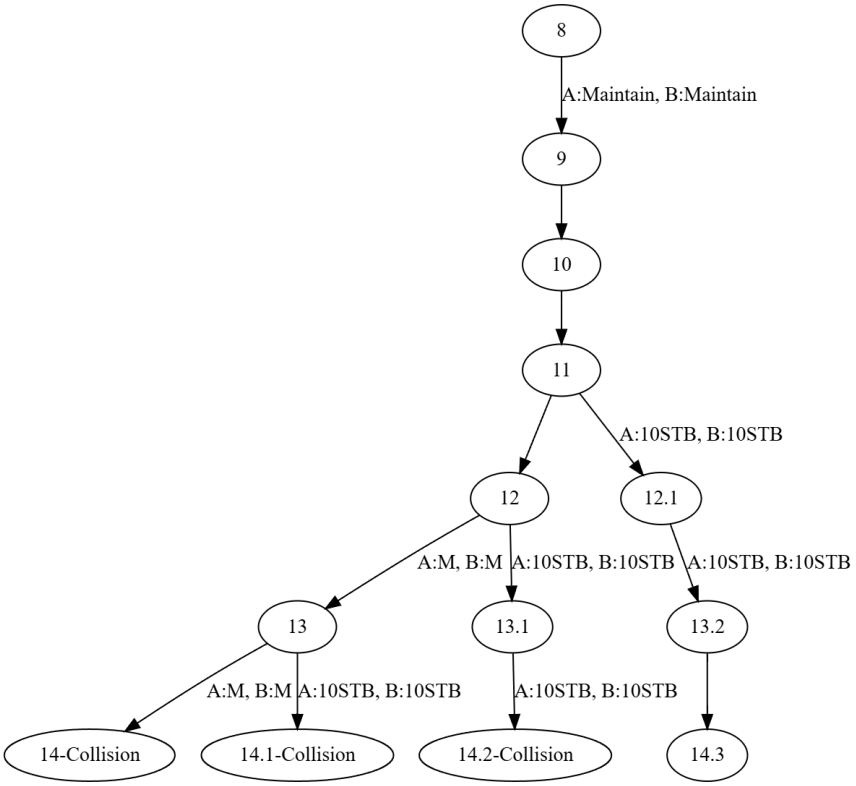


Fig 14. Final Search tree for collision avoidance

Going back to route

After maneuvering, the ships should head back to their destinations. So after node 14.3 - the algorithm simulates an horizon distance where the ships take the opposite of the maneuvers they took in order to avoid collision. The way it works is pretty similar to the collision avoidance generating nodes process.

Collision avoidance for multi-vessel scenario

Despite our extensive efforts and the development of various strategies, we were unable to fully resolve the collision avoidance problem for multi-vessel scenarios yet. The complexity of these interactions remains a significant challenge.
Because COLREG is well defined for 2-ship scenarios - a dynamic solution is needed for multi-vessel scenarios.
Our idea for now is that each ship should turn to its starboard in order to avoid the collision, acting like a roundabout. Fig 15. visualizes this behavior.

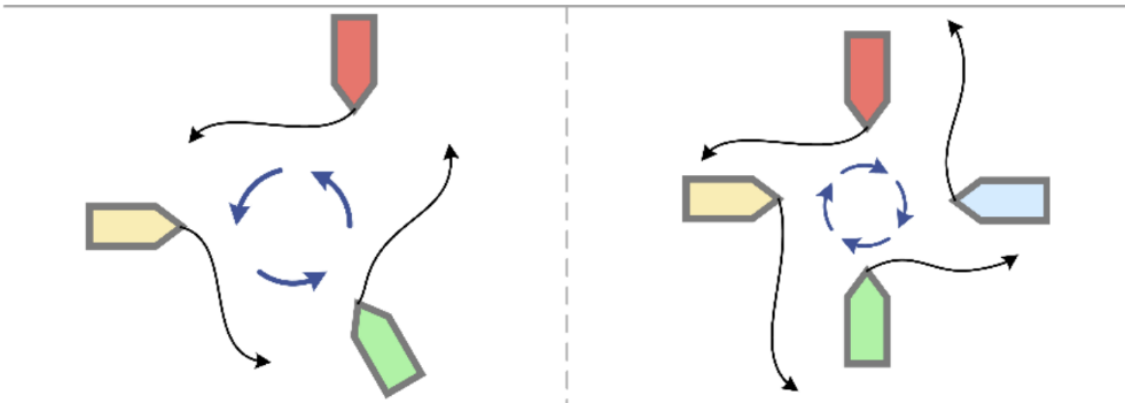


Fig 15. Multi-vessel roundabout behavior [7]

Our main focus on the next part is to develop the simulator and focusing on solving the problem for multi-vessel scenarios.

Expected Achievements

We aim to develop a collision avoidance algorithm for autonomous ships that fully complies with COLREGs, focusing on overtaking, head-on, and crossing situations for power-driven vessels. The algorithm is designed to handle complex multi-vessel scenarios, including nested situations where new collisions may arise from previous maneuvers.

Practically, we expect to create a functional simulator integrated with our algorithm, allowing users to test various scenarios by adjusting variables like the number of vessels and their source and destination points. This simulator will enable us to evaluate the algorithm's performance and scalability, particularly how completion time is affected by increasing numbers of vessels.

Theoretically, we anticipate that our algorithm will not only prevent collisions but also ensure full COLREG compliance in complex situations. We aim to identify any limitations and propose solutions, contributing to the advancement of autonomous maritime navigation.

In summary, the expected outcomes of our project are:

1. A COLREG-compliant collision avoidance algorithm capable of handling complex, multi-vessel scenarios.
2. A simulation system for testing and visualizing the algorithm's performance under various conditions.

By achieving these objectives, we hope to demonstrate the feasibility of fully autonomous maritime navigation systems that operate safely and efficiently, addressing current and future demands of global maritime transportation.

Requirements & Evaluation Criterias

This section outlines the requirements from the project / solution. It is divided into functional, non-functional and general requirements. The requirements are outlined in a manner that is self-contained and easy to grasp.

Each requirement includes an evaluation / success criteria column – this is the test method that will be applied to validate if the requirement has been met.

Definitions

1. Horizon – the lookahead distance considered by the vessels when searching for potential collision and for collision avoidance maneuvering.
2. Safety zone – a circular area around the ships with configurable radius e.g., 50m.
3. Collision – a case where two or more ships enter each other's Safety zone.
4. Source / Destination – coordinates defining the start position and end position of the ships. Different ships have different sources and destinations.

1. Functional requirements – pertain to functionality of the algorithm

#	title	Description	Evaluation / success criteria	Comment
1.1	Collision detection	The solution will incorporate collision detection capability. Detection should be done ahead of time and in a pre-defined and configurable horizon distance, e.g., 2NM.	1. Demonstration of the collision detection capability using the simulator. 2. 30 different scenarios. Each scenario will include 2 to 8 ships with different source and destination and speeds. 3. Ships with no risk of collision in the checked horizon will be marked with green circles. Ships who are at risk of collision – orange circles.	Demonstrating this capability does not mandate demonstration of collision avoidance maneuvering. It means that for this requirement it is OK for the ships to enter each other's safety zone in their path to destination.

			Ships who enter each other's safety zone – red circles.	
1.2	Collision avoidance: – 2 vessel scenarios	The support for 2-Vessel collision avoidance maneuvering will be according to COLREG rules 13, 14, and 15. Ships assigned the "Give-way" role will do it in full compliance with COLREG Rule 16. Ships assigned "Stand-on" role will do it in full compliance with COLREG Rule 17.	1. Demonstration of the collision avoidance capability in a 2-vessel scenario using the simulator. 2. For each rule (13, 14, 15), 10-different scenarios will be demonstrated, each with different source and destination speeds of the ships. 3. The scenarios should be crafted in such a manner that the collision course between the ships will be obvious and clear. 4. The roles of the ships for the avoidance maneuver will be clearly marked in the Simulator UI. Ships assigned the "Stand on" role will be labeled "Stand on".	No scenario should end with a collision. All scenarios should strictly follow the rules as outlined in the description. Special attention will be made for compliance with Rules 16 and 17.

			Ships assigned “Give way” will be labeled “Give way”.	
1.3	Collision avoidance: > 2 vessel scenarios	The support for > 2 Vessel collision avoidance maneuvering will be done according to COLREG Rule 8 (Action to avoid a collision)	1. Demonstration of the collision avoidance capability in 3-4-and 5 vessel scenarios using the simulator. 2. For each case (3, 4, 5 vessels) 5 different scenarios will be demonstrated, having different source and destination speeds of the ships involved.	No scenario should end with a collision. In some cases, it may be reasonable to completely stop a ship’s movement to avoid collision. At most 2 ships are allowed to stop or significantly slow down.

2. Non-functional requirements – pertain to the solution / product

#	**title**	**Description**	**Evaluation / success criteria**	**Comment**
2.1	Scenario definition	How the scenarios are defined for simulation.	The simulator will support types of scenario definitions. 1. Automatic – the user supplies the # of vessels in the scenario, and the simulator will do the rest (select source and destination, max speed, etc.) 2. Manual – the scenario definition (# of vessels, source,	In manual mode allows the exact same scenario to be tested again and again. In automatic mode this is not possible.

			destination, max speeds, etc.) will be entirely provided by the user in a "scenario definition" file.	
2.2	Scenario demonstration	Quality of the UI and ability to demonstrate the solution in a clear, smooth, and informative manner.	1. The scenarios should be demonstrated (visualized) in a clear and smooth manner. 2. The speed in which the scenario progresses on the screen, and accompanying information e.g., "Give way" / Stand-on" flags should align with the human eye and ability to grasp and process the information.	"Simulator" in this context pertains to the SW framework that runs the algorithm, including the run of the algorithm itself.
2.3	Scenario evaluation	Goodness metric for the algorithm	1. The simulator will record the overall time it took to complete each scenario. 2. This is defined as the sailing time (not simulation execution time). 3. The simulator will calculate the ratio between the "ideal" sailing time (time to complete the	For example, if the scenario takes 30 minutes of sailing time to complete (assuming no collisions – each ship sails directly to destination), but with collision avoidance maneuvering it would take 45 minutes to complete, the ratio is 30/45 = 0.66.

			scenario assuming no collision) and the sailing time under collision avoidance maneuvering.	
2.4	SW architecture / Coding quality	Ability to understand the code, project structure, and functionally. Division into SW modules based on functionality and scope.	It should be clear from referring to the code base of the project on the general structure of the project, functionally and purpose of the different components.	
2.4	Simulator / Algorithm interface	Code separation generality and scalability	1. The simulator should include a clear and simple API to call the algorithm and retrieve its output. 2. Likewise, the algorithm should implement that interface for easy integration with the Simulator. 3. The API should be generic and must not be tied to a specific algorithm implementation e.g., search trees. 4. Switching to a different algorithm implementation should be easy	Support the ability to switch to a different algorithm implementation.

			as modifying “include” statements in the simulator code.	

3. General requirements

	Title	**Description**	**Evaluation / Testing / success criteria**	**comment**
3.1	Logging	The simulator will record a detailed trace (log file) for each run.	The log will include, at minimum: 1. Scenario setup information: information that allows reproduce the scenario, including: # of vessels, source and destination, height, max speeds, and any additional information. 2. Detailed trace of the run: For each timestamp during simulation run, a record for every vessel in the scenario, that includes its current position, speed, state (“clear”, “give way”, “stand on”, etc. 3. The simulator will include a capability to load a trace file and replay it	The trace / log could be divided into multiple files.

3.2	Units	The units used by the algorithm/simulator	1. Nautical Miles (NM) for distance (board coordinates, distance between ships, lookahead distance, etc.) 2. Knots (NM/H) for speeds. 3. Meters – for Safety zone around the ships.	

Collecting Requirements

The requirements for our project were collected through a combination of in-depth literature review and ongoing discussions with our project supervisor. We started by reading academic papers on maritime navigation and collision avoidance, particularly those focusing on algorithms that comply with COLREGs. This helped us understand the current state of research and identify essential features and constraints for our system.
Regular meetings with our supervisor allowed us to refine these requirements, ensuring they are practical and aligned with both theoretical insights and real-world applications. We collected the requirements by studying various academic approaches to the problem and through meetings with our supervisors.

Engineering Process

This section details the research and development process we undertook to accomplish the project's objectives. It outlines the work carried out during the semester and is divided into two parts: the process and the product of the project.

Process

In this project, we tackle the problem of collision avoidance and path planning for autonomous ships using search tree algorithms with pruning techniques. Our primary goal is to enable autonomous vessels to navigate safely in congested waters while fully complying with the International Regulations for Preventing Collisions at Sea (COLREGs), specifically Rules 8 and 13–17.

We began our engineering process by focusing on the theoretical background. To thoroughly understand the core challenges, we expanded our knowledge on several key areas:

1. Maritime Navigation and Collision Avoidance: We studied the fundamentals of maritime navigation, focusing on the factors that contribute to collisions and the strategies employed to prevent them.
2. COLREGs Compliance: We delved into the COLREGs to comprehend the legal requirements and guidelines that govern ship movements. Emphasizing Rules 8 and 13–17, we analyzed how these regulations dictate vessel behavior in overtaking, head-on, and crossing situations.
3. Path Planning and Collision Avoidance Methods: We explored various approaches to path planning and collision avoidance, this involved studying different algorithmic strategies used in dynamic environments for autonomous navigation. By examining these methods, we gained insights into how they can be applied to our project, enabling us to develop a solution that efficiently manages navigation while ensuring compliance with maritime regulations.

To bridge theory and practice, we reviewed related work and existing solutions, as detailed in the literature review.
A significant part of our process was dedicated to integrating COLREGs into the algorithmic framework. We investigated methods to formally represent the regulations within the algorithm, making sure that all generated paths are legally compliant. This required us to consider how to encode rules as constraints and incorporate them into the algorithm.

We recognized the need to develop a simulator to test our algorithm. Designing this simulation presented new challenges, such as:

1. Environment Representation: Creating a virtual sea with appropriate dimensions and boundaries.
2. Multi-Vessel Interaction: Enabling multiple autonomous ships to operate within the simulation.

The simulation should allow the user to test a multi-vessel scenario

Product - The Simulator

The product is a simulation system that allows the user to configure and run multi-vessel maritime scenarios to test the collision avoidance algorithm. The system provides an interface where the user can select various variables, such as the number of vessels, locations of source points (starting positions), destination points, timestamp and other settings.

After the user finishes entering the simulation parameters and presses the start button, the system begins by initializing the vessels. It creates the specified number of vessels and assigns each one a source point and a destination point based on the user's input. The vessels are initialized with default attributes or any user-specified characteristics, such as speed. Once the vessels are initialized, the system applies the collision avoidance algorithm designed to comply with COLREG. The algorithm plans the vessel's path from its source to its destination while considering the presence and predicted movements of other vessels. By

accounting for potential interactions, the algorithm enables each vessel to navigate safely and efficiently within the simulated environment.

The simulation runs and updates the positions of all vessels over time. As the vessels move through the environment, they interact with one another, making real-time decisions based on the collision avoidance algorithm and the behaviors of other vessels.

Throughout the simulation, the system provides a visual representation that allows the user to observe the movements of the vessels in real-time. Users can monitor how each vessel navigates, avoid collisions, and comply with COLREG, gaining insights into the effectiveness of the algorithm and the vessels' behaviors.

The simulator continuously checks for potential collisions and monitors compliance with COLREGs. The system logs the scenario between two vessels - that means, if an overtaking scenario takes place, the system will log the time, id of vessels involved and the scenario. This ongoing monitoring ensures that any issues are recorded, which will enable users to identify areas where the algorithm may need improvement.

At the end of the simulation, the system displays the outcomes to the user. The results include the time that took the scenario to run, whether all vessels reached their destinations successfully, instances of near-collision, and efficiency metrics such as total travel time. This comprehensive feedback provides valuable information on the performance of the vessels and the collision avoidance algorithm.

After reviewing the results, the user can decide to adjust the simulation parameters. The system allows for re-running the simulation with modified inputs, enabling the user to test different scenarios or improve performance.

System Architecture and Modular Design

The simulator is designed with a modular architecture that separates the simulation framework from the collision avoidance algorithm. This separation ensures scalability and flexibility, allowing for easy integration of different algorithm implementations. The key components are:

1. Simulation Engine: Responsible for initializing vessels, updating positions over time, handling vessel interactions, and managing the overall simulation flow.
2. Collision Avoidance Algorithm Module: Encapsulates the logic for detecting potential collisions and determining COLREG-compliant maneuvers.

The simulator includes a clear and simple API that facilitates communication between the simulation engine and the collision avoidance algorithm. This API is generic and not tied to any specific algorithm implementation, making it easy to switch to different algorithms if needed. This design promotes code reusability and simplifies future enhancements.

Key Features of the API:

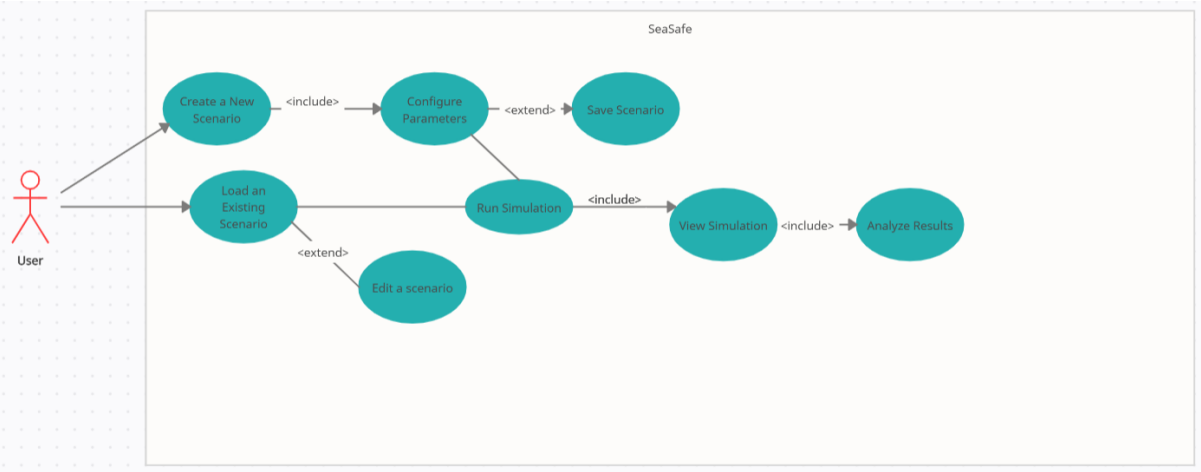
1. Input Parameters: The simulation engine provides the algorithm with necessary data such as vessel states, positions, velocities, and any relevant environmental information.
2. Output Data: The algorithm returns maneuvering decisions for each vessel, including speed adjustments and heading changes, ensuring compliance with COLREG rules.
3. Scalability: The API supports scenarios with varying numbers of vessels, enabling the algorithm to handle multi-vessel interactions efficiently.

Code Separation and Scalability

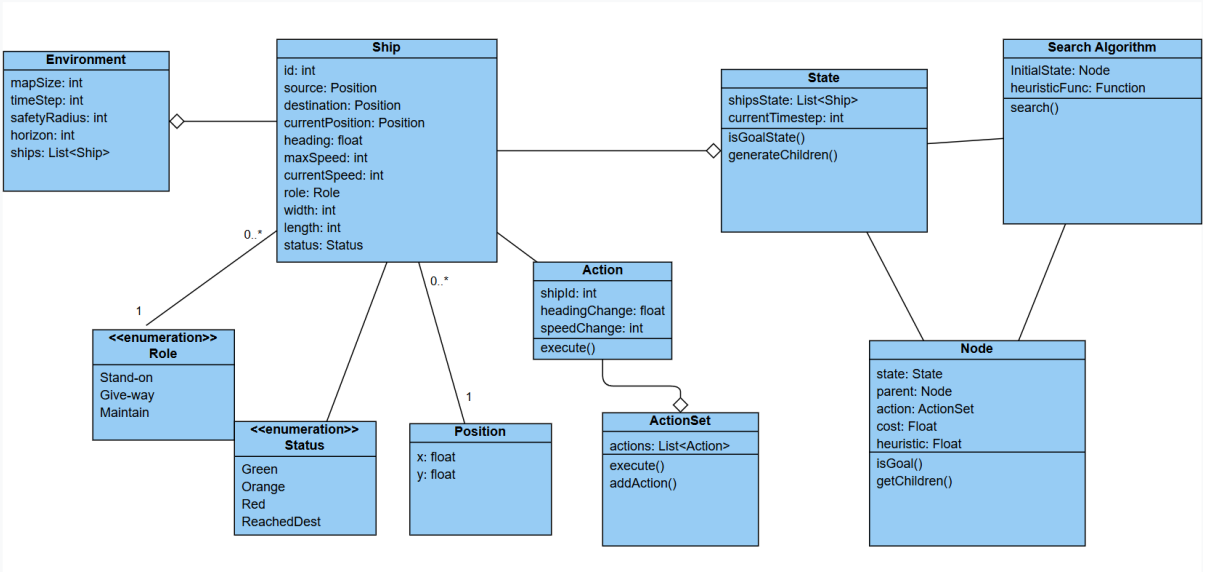
1. The algorithm and simulator are separate modules, allowing independent development and testing.
2. Switching to a different algorithm implementation is straightforward, requiring minimal changes (e.g., modifying import statements).
3. The modularity enhances the simulator's scalability and adaptability to future advancements.

UML

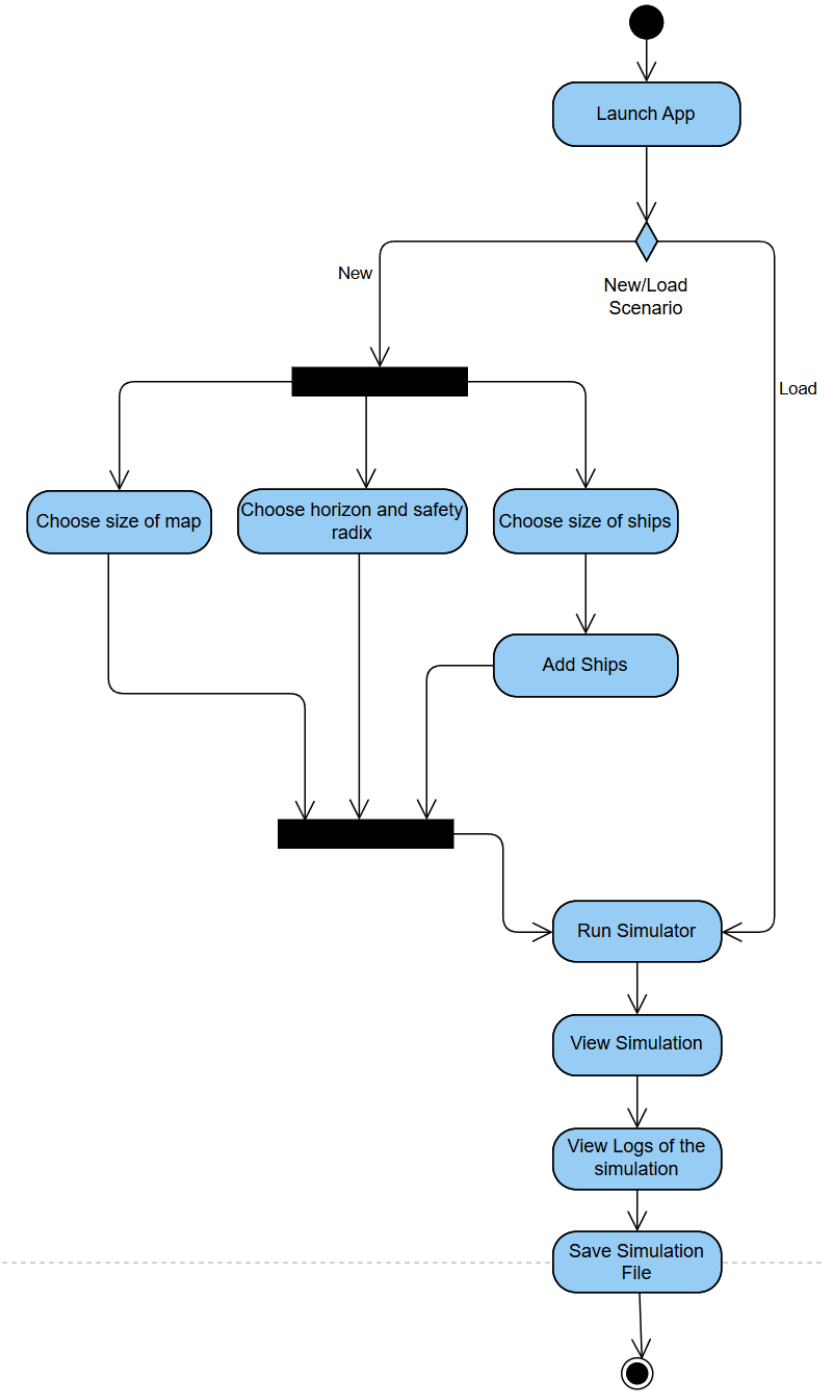
Use Case:



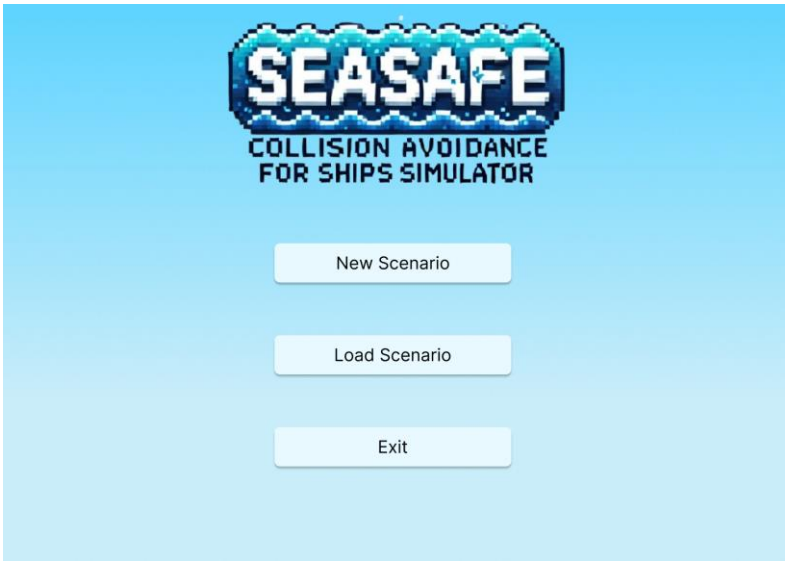
Class Diagram:

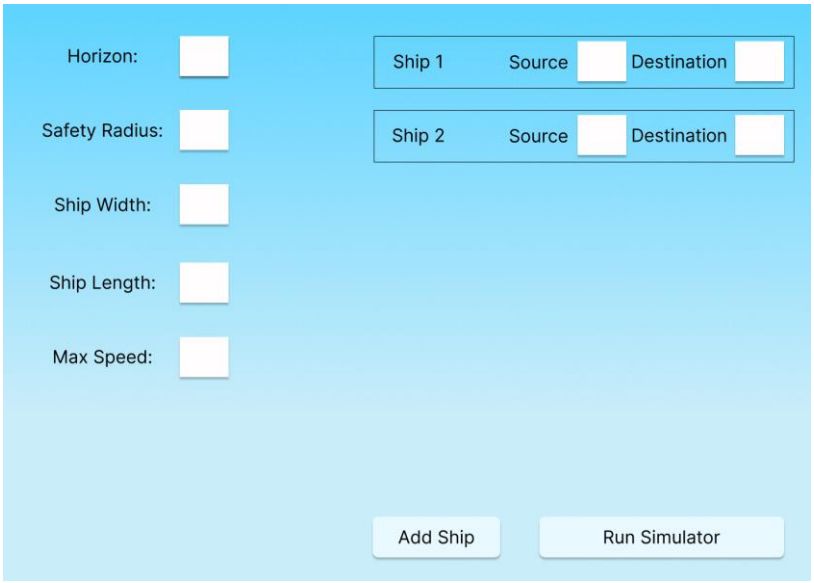


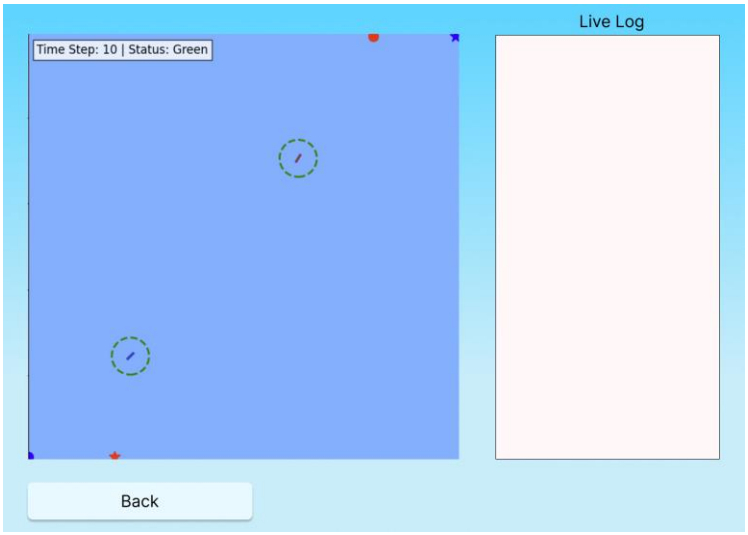
Activity Diagram:


Launch App
New
New/Load Scenario
Load
Choose size of map
Choose horizon and safety radix
Choose size of ships
Add Ships
Run Simulator
View Simulation
View Logs of the simulation
Save Simulation File

UI Design


SEASAFE
COLLISION AVOIDANCE
FOR SHIPS SIMULATOR
New Scenario
Load Scenario
Exit


Horizon:
Safety Radius:
Ship Width:
Ship Length:
Max Speed:
Ship 1
Source
Destination
Ship 2
Source
Destination
Add Ship
Run Simulator



Challenges

Developing a COLREG-compliant collision avoidance algorithm for autonomous ships poses several challenges:

1. Algorithm Scalability - As the number of vessels in the simulation increases, the computational complexity of predicting potential collisions and determining avoidance maneuvers grows significantly. The algorithm must be capable of processing information from multiple vessels and making real-time decisions without compromising performance. Ensuring scalability and computational efficiency is essential to handle complex scenarios involving many vessels simultaneously.
2. Verifying the algorithm - Verifying and validating the collision avoidance algorithm presents a significant challenge because it depends on the development of the simulator, which will happen in the second phase of the Capstone project.

Tools

1. **Python**: Python will be the primary language used for the development of the simulation and the implementation of the collision avoidance algorithm. Python is chosen due to its versatility, ease of development, and strong ecosystem of libraries for both computational tasks and visualization. Additionally, Python's ability to handle mathematical operations and integrate with various libraries makes it ideal for simulating real-time maritime scenarios.
2. **Python libraries such as NumPy, Matplotlib and Networkx -** We will utilize these Python libraries to enhance our simulation. NumPy provides efficient numerical computation for handling arrays and mathematical operations, essential for simulating vessel movements and collision detection. Matplotlib enables us to create visualizations of simulation data, such as vessel trajectories and collision points, aiding in analysis and presentation. NetworkX allows modeling the maritime environment as a graph, representing vessels and their interactions, which supports advanced algorithms for pathfinding and collision avoidance. Together, these libraries enhance the computational efficiency, visualization capabilities, and algorithmic functionality of our project.
3. **Pygame**: We plan to use Pygame, a Python library designed for writing games and graphical simulations. Pygame will be used to create a visual interface that allows us to display the vessels' movements in real time. Through this interface, we can observe the interactions between vessels, detect potential collisions, and track the paths generated by the algorithm. Pygame will also allow us to build a graphical user interface (GUI) where users can define parameters such as the number of ships, starting positions, destinations, and speed.
4. **Git for Version Control**: We will use Git for version control to manage the codebase. Git will allow for effective collaboration between team members, ensuring that all changes to the code are tracked and that multiple contributors can work on the project simultaneously. Additionally, Git will provide a mechanism for rolling back to previous versions if necessary. We will also leverage GitHub to host the repository, track issues, and collaborate on different branches of the project.
5. **Pytest:** Pytest will be the tool we use for writing and executing automated tests. Unit testing and integration testing will be vital to ensure that all components of the algorithm are functioning correctly and that any changes to the code do not introduce errors. Pytest's simplicity and ease of use make it a good choice for managing the test suite.

Testing Plan

In order to test our program, we constructed several test cases

Number	Test Case	Expected Result
1	Inputting non-numerical value for the number of ships	Error message: "Please enter a valid number of ships."
2	Press "Run Simulator" before inputting the number of ships	Error message: "Please enter the number of ships."
3	Number of ships entered exceeds maximum allowed (e.g., entering 9 when maximum is 8)	Error message: "Ship number should be 8 or fewer."
4	Inputting negative number for the number of ships	Error message: "Number of ships cannot be negative."
5	Inputting zero as the number of ships	Error message: "Please enter at least one ship."
6	Filling non-coordinate value for ship source point	Error message: "Please enter a valid coordinate for source point."
7	Filling non-coordinate value for ship destination point	Error message: "Please enter a valid coordinate for destination point."
8	Press "Run Simulator" before inputting source points	Error message: "Please enter ship source points."
9	Press "Run Simulator" before inputting destination points	Error message: "Please enter ship destination points."
10	Inputting invalid coordinate ranges (e.g., latitude beyond -90 to 90, longitude beyond -180 to 180)	Error message: "Coordinates are out of valid range."
11	Inputting non-numerical values for ship speeds	Error message: "Please enter valid numerical values for ship speeds."
12	In manual scenario mode, inputting inconsistent number of source and destination points	Error message: "Number of source and destination points must match the number of ships."

13	Running a simulation with two ships on non-collision courses	Ships proceed to destination without collision warnings; no maneuvers needed.
14	Running a simulation with two ships on a collision course	Collision detected; ships marked appropriately; collision avoidance maneuvers executed; no collision occurs.
15	Running a simulation with multiple ships (3-5) on collision courses	Collision detection and avoidance executed; ships maneuver safely; no collision occurs.
16	Exceeding the maximum allowed horizon distance (e.g., setting horizon greater than system limit)	Error message: "Horizon distance exceeds maximum allowed value."
17	Setting safety zone radius to an invalid value (e.g., negative number)	Error message: "Please enter a valid safety zone radius."
18	Loading a malformed or corrupted scenario definition file	Error message: "Scenario file is invalid or corrupted."
19	Running a simulation and verifying that logs are generated	Log files are created, containing scenario setup and detailed trace of the run.
20	Verifying units used in the simulation (distances in Nautical Miles, speeds in Knots, safety zones in meters)	All units are consistent and correctly displayed throughout the simulator.
21	Swapping the collision avoidance algorithm with an alternative implementation	New algorithm integrates smoothly; simulation runs correctly; minimal code changes needed.
22	Running a scenario to completion and recording sailing time	Sailing time is calculated; ratio between ideal and actual time is displayed accurately.
23	User cancels the simulation during execution	Simulation stops gracefully; resources are freed; user can restart or exit.

Since the simulator has not yet been developed, the following section outlines the planned approach to testing and validation that will be employed once the simulation environment is functional.

1. **Unit Testing:** We will develop unit tests for the core functions of the algorithm, such as collision detection, path planning, and status updates for vessels. Each function will be tested in isolation to verify its correctness. For example, the collision detection module will be tested with predefined scenarios to ensure that it can accurately detect when vessels are on a collision course.
2. **Integration Testing:** Once individual modules are verified, integration testing will be conducted to ensure that the different parts of the system (e.g., collision detection, COLREG rule enforcement, and navigation adjustment) work together seamlessly. This will ensure that the system is cohesive and can handle complex multi-vessel interactions without errors.
3. **Simulation Testing:** Once the simulator is developed, it will be used to run various test scenarios. These will range from simple two-vessel encounters (crossing, overtaking, and head-on) to complex, multi-vessel scenarios. During each simulation, the system will log all decisions made by the algorithm and compare them against the expected outcomes according to COLREG. The accuracy of the collision avoidance decisions will be validated against known maritime rules.
4. **Performance Testing:** The performance of the algorithm will be tested by running simulations with an increasing number of vessels. We will measure the response time of the system in calculating paths, detecting collisions, and adjusting courses. Performance metrics such as computation time for each simulation, and how well the system handles scenarios involving more vessels, will be key indicators of success. We aim to ensure that the system can handle real-time scenarios efficiently.
5. **Manual Review of Simulations:** After running simulations, we will manually review the simulation logs and outcomes to verify that the system's decisions are consistent with the expected behaviors. Special focus will be placed on ensuring that all COLREG rules are followed during each maneuver.

By following these methods and leveraging the outlined tools, we will ensure that the algorithm is rigorously tested and validated as the simulator is developed. This approach will ensure that the final system is accurate, reliable, and efficient in multi-vessel collision avoidance.

References:

1. Li, L.-N., Yang, S.-H., Cao, B.-G., and Li, Z.-F., “A summary of studies on the automation of ship collision avoidance intelligence,” Journal of Jimei University, China, Vol. 11, No. 2, pp. 188-192 (2006)

2. Ming-Cheng and Hsueh, Chao-Kuang (2010) "THE STUDY OF SHIP COLLISION AVOIDANCE ROUTE PLANNING BY ANT COLONY ALGORITHM," Journal of Marine Science and Technology: Vol. 18: Iss. 5, Article 16.

3. Modeling of Ship Trajectory in Collision Situations by an Evolutionary Algorithm, Roman S´mierzchalski, and Zbigniew Michalewicz, Senior Member, IEEE

4. Intent inference of ship maneuvering for automatic ship collision avoidance,Yonghoon Cho, Jungwook Han, Jinwhan Kim

5. Li, S., Liu, J., & Negenborn, R. R. (2019). Distributed coordination for collision avoidance of multiple ships considering ship maneuverability. Ocean Engineering, 181, 212-226. https://doi.org/10.1016/j.oceaneng.2019.03.054

6.Andreas Nygard Madsen, Magne Vollan Aarset, Ole Andreas Alsos, Safe and efficient maneuvering of a Maritime Autonomous Surface Ship (MASS) during encounters at sea: A novel approach, Maritime Transport Research, Volume 3, 2022.

7. Guan Wei and Wang Kuo, COLREGs-Compliant Multi-Ship Collision Avoidance Based on Multi-Agent Reinforcement Learning Technique, 2022

8. Lyu, Hongguang. (2018). COLREGS-Constrained Real-time Path Planning for Autonomous Ships Using Modified Artificial Potential Fields.

9. COLREG-Compliant Optimal Path Planning for Real-Time Guidance and Control of Autonomous Ships, Raphael Zaccone 2021

10. Russell, S., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach (4th Edition). Pearson.

11.Jordan Gleeson, Matthew Dunbabin, Jason J. Ford, COLREG Scenario classification and Compliance Evaluation with temporal and multi-vessel awareness for collision avoidance systems,Ocean Engineering,Volume 313, Part 3,2024,119552,ISSN 0029-8018,

12. Review of Maritime Transportation, United Nations trade & development

13. Hasanspahić, N.; Vujičić, S.; Frančić, V.; Čampara, L. The Role of the Human Factor in Marine Accidents. *J. Mar. Sci. Eng.* 2021, *9*, 261. https://doi.org/10.3390/jmse9030261