An Improved Real-time Collision-Avoidance Algorithm Based on Hybrid A* in a Multi-Object-Encountering Scenario for Autonomous Surface Vessels

Tianlei Miao[a,b,1], Ehab El Amam[b,2], Peter Slaets[c,3] and Davy Pissoort[a,4]

[a]Department of Electrical Engineering (ESAT), Waves: Core Research and Engineering (WaveCore), Bruges Campus, KU Leuven, Bruges, Belgium
[b]Autonomy department, RH Marine Netherlands B.V. , Schiedam, The Netherlands
[c]Group T, Robotics, Automation and Mechatronics(RAM), Leuven Campus, KU Leuven, Belgium

ARTICLE INFO

Keywords:
real-time Collision Avoidance
Hybrid A*
Autonomous Surface Vessels
multiple objects encountering

ABSTRACT

Collision-avoidance algorithms for maritime autonomous surface vessels are increasingly being analyzed for challenging scenarios. However, the trade-off between the computation's effectiveness and efficiency is always an intractable problem, especially in complex, multi-object-encountering cases. This paper proposes an improved Hybrid A* algorithm that searches a node map with discrete control behaviors as 'movement options'. The real-time performance in maritime, multi-object-encountering scenarios is improved by narrowing the search space with the proposed pre-processing method, i.e., the collision velocity check (CVC). The following is incorporated within the mentioned approach: the compliance of the relevant rules in *International Regulations for Preventing Collisions at Sea* and the dynamics of the ship. A comparative study with various multi-target scenarios is conducted with a baseline method. Simulation results show the effectiveness of the proposed method. The use of the CVC significantly reduces the time of the computation with little reduction in the solution's accuracy. Additionally, the proposed method is applied within a real-time simulation environment, which is fed by logged data collected during an actual sea trial.

1. INTRODUCTION

An autonomous surface vehicle (ASV) can reduce the number of sailing accidents caused by human error. In addition, onboard crew expenses are greatly reduced. However, a collision-avoidance (CA) algorithm is always an intractable problem in terms of real time, robustness, and rule compliance. Even though a substantial amount of research has been conducted in this area, the main challenge faced by many researchers when dealing with the massive uncertainties and complexity of the dynamic environment for local CA still exists. Furthermore, compliance with the *International Regulations for Preventing Collisions at Sea* (COLREGs) has accentuated the calculation complexity of CA algorithms. In addition, uncertainty and calculation costs increase further with the growth of the number of targets in multi-target scenarios. This paper proposes a hybrid method based on the improved Hybrid A* using discrete control behaviors as the input with a node map. A collision pre-check method, i.e. the collision velocity check (CVC), is used to narrow the search space, resulting in high computation speed. This method aims to provide ASV systems with an effective solution in real time, with an appreciable frequency update, so that collisions can be avoided. The COLREGs rules 6, 8, 13, 14, 15, and 16 (seen in Appendix) are considered. Complying with these rules is designed as a part of the cost functions. In addition, the proposed algorithm can offer suggested routes or alternatively new control commands (e.g., a new course and speed setpoint) to assist manned surface vehicles (MSVs) with an Integrated Navigation System (INS).

Benefiting from the development and applications of artificial intelligence in, e.g., the automotive and airplane industries, an increasing number of researchers are trying to apply deep learning (DL) to the maritime CA context. Cheng and Zhang (2018) used deep reinforcement learning (DRL) with a deep Q-networks architecture to achieve obstacle-collision avoidance. Meyer et al. (2020) applied a DRL agent trained in a simulated environment to the real and complex shorelines of the Trondheim fjord. Although most articles claim the effectiveness of the proposed method, the lack of explanation makes these methods challenging to be accepted by the industry. Besides, the processing of sensors and uncertainties is included in the end-to-end DL, which is similar but still varies to the other methods. The lack of a standardized description of the perceived maritime environment also leads to differences between the data sets,

*Corresponding author: Tianlei Miao
tianlei.miao@rhmarine.com (T. Miao)

[1]Tianlei Miao is a Ph.D. researcher at the KU Leuven and RH Marine on ensuring autonomous sailing from A to B. He obtained a joint Masters in Science from both Norwegian University of Science and Technology and Royal Institute of Technology in 2019. His current research interests include data fusion, multi-target tracking, and collision avoidance in autonomous sailing.

[2]Ehab El Amam is an engineering consultant at RH Marine Netherlands B.V., the Netherlands. His expertise includes dynamic position system, autopilot system, and energy management system.

[3]Peter Slaets is an associate professor in Robotics, Automation, and Mechatronics(RAM) at the Faculty of Engineering Technology, Catholic University of Leuven. His research interests include autonomous localization, navigation, embedded hardware, and sensor fusion.

[4]Davy Pissoort is a professor in the M-Group (WaveCore), Faculty of Engineering, KU Leuven. His research interests include autonomous systems, system safety, electromagnetic compatibility, and electromagnetic interference.

which reduces the reliability of the data.

Fuzzy logic is a popular method that combines expert knowledge with international rules like COLREGs. Hu et al. (2020) proposed a real-time collision-avoidance method based on fuzzy logic. The solutions are retrieved from a knowledge base to find a new heading command for collision avoidance. Brcko et al. (2021) proposed a multi-parametric decision model based on fuzzy logic. The model calculates the necessary course alteration in a collision-avoidance situation. However, the dependence on prior knowledge is always a critical issue for this method.

Another option is the field method, such as the artificial potential field (APF)(Khatib, 1986) or the limited cycle method (LCM)(Soltan et al., 2009). A potential field is created to describe the collision risk with objects and then converts the path-finding problem into a gradient-descent problem or a dynamic-model-solving problem. For example, Beser and Yildirim (2018) proposed a COLREGs-compliant APF method for the obstacle avoidance of ASVs. Lyu and Yin (2019) used this modified APF for ASVs in real-time path planning. However, the main drawback of the field method is that the algorithm might be trapped in a local minimum, especially in terms of the dynamic factors. Though many studies have been conducted to solve the local minima problem, finding a generalized method still remains unsolved.

Optimization-problem solving can also be a class of solutions for collision avoidance. These methods try to find the corresponding variables to optimize the cost function with all kinds of constraints. Typically, such optimization problems can be solved with computational algorithms, such as the evolution algorithm (EA)(Lin et al., 1994) or the ant-colony optimization (ACO) algorithm (Dorigo et al., 2006). Szłapczyński and Ghaemi (2019) proposed a method of planning safe trajectories that is able to simultaneously make decisions in real time and without any interaction with a human operator, handle basic types of encounters and guarantee compliance with the COLREGs. A multi-criteria ACO-based algorithm for ships in the environment with static and dynamic obstacles was proposed by Lazarowska (2017). More variants (Huang et al., 2019) are proposed to cope with a generalized situation and the collision-avoidance behaviors of other ships. However, these methods usually require a relatively long computation time (Vagale et al., 2021), especially when the ships are in complex scenarios. Apart from that, the optimal solution is usually given by solvers, which is not very transparent to users. EA might not converge within the desired time in complex scenarios, and a discrete algorithm's computational time can be slow when the sample numbers or solution space are large. To speed up the solving process, a method, velocity obstacle (VO) (Zhuang et al., 2016) is proposed for optimization problems by excluding dangerous velocity areas. Besides, a method based on model predictive control (MPC) with discrete control behaviors was proposed by Johansen et al. (2016). It is computationally simple and yet quite versatile as it accounts for the dynamics of the OS, the dynamics of the steering and propulsion system, and the forces due to wind and ocean current. However, this method is not globally optimal and over-depends on the risk model for the future. Despite this, the proposed concept of discrete control behaviors provides a new idea to balance the computational effectiveness and efficiency in practice.

Graph search methods, a kind of computational algorithm for the optimization problem, are classical methods for pathfinding and collision avoidance. The well-known algorithms, like Dijkstra (Dijkstra et al., 1959), A* (Hart et al., 1968) and their numerous variants, have been validated in various fields. Instead of searching the optimization in a continuous or discrete space, these methods search a graph representation of the space. One variant, Repairing A* (R-RA*) was used for maritime CA by Campbell and Naeem (2012) within a decision-making framework. Another variant Theta* (Daniel et al., 2010) uses some key points (such as corners) and a line of sight pointing directly back to an ancestor during retrace. Kim et al. (2014) proposed an approach based on the Theta* algorithm to create paths in real-time, considering both angular rate (yaw rate) and heading angle of unmanned surface vehicles. Singh et al. (2018) extended the implementation of this Theta* in an environment cluttered with static and moving obstacles and different current intensities. The variant, Hybrid A* (HA*), unlike conventional variants that only allow visiting centers, corners, or edges of grid cells, associates with each cell a continuous state of vehicles (Dolgov et al., 2008). Bitar et al. (2020) detailed a two-stage trajectory planner, where the initial step uses a discrete polygonal representation of the configuration space, such that a Hybrid A* algorithm can compute an initial dynamically feasible trajectory. Ship dynamics can be considered during the graph search. However, there is still a common problem with these methods: the computational load depends on the resolution of the map or the sample number, even though there has been a lot of work to overcome it (Kurzer, 2016; Chen et al., 2016).

Considering the limitations of each method, this paper proposes a hybrid method that is applicable to maritime CA scenarios. The proposed method uses discrete control behavior as an input and uses one of the newest variants of A*, Hybrid A*, with a dynamic node map to find the best route. The dynamics of the own ship (OS) are considered during each movement. A pre-processing step, the CVC, is proposed to narrow the search space. It uses the same idea as VO and has an easier implementation with discrete inputs. COLREGs rules 6,8,13,14,15,and 16 are considered as a part of the cost function. This method is designed for the application in different scenarios and scopes, from an intermediate sailing scope (e.g., 5 nautical miles) to a local scenario (within 1 nautical mile). More specific

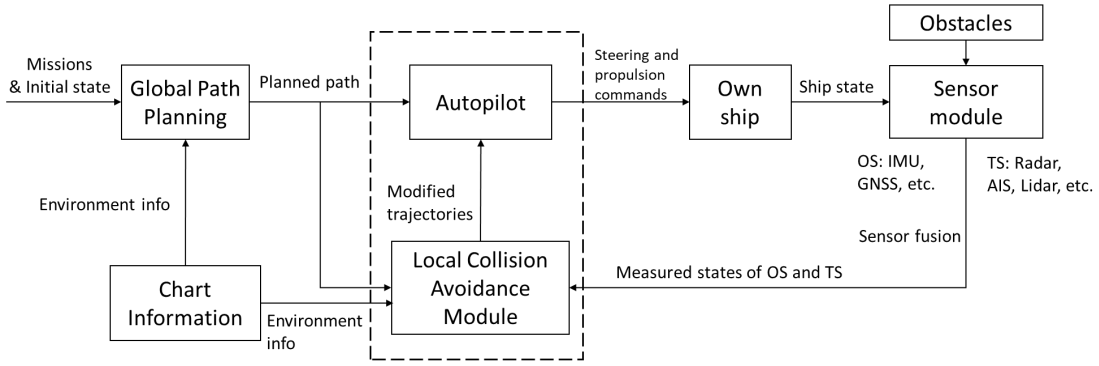


Figure 1: Overview diagram of information flow of Autonomous Sailing System.

contributions are listed as follows:

- An improved Hybrid A* is proposed using a node map associated with discrete control behaviors instead of a grid map. In this way, each movement during the search process can be directly associated with specific control commands. This algorithm can modify the local route and indicate the specific control commands to follow the route.
- The real-time performance of the proposed method is improved using the proposed pre-processing method, i.e. collision velocity check. It uses the same idea as VO, i.e., the selected velocity should fall outside the collision velocity cone. With the given discrete control commands, each control combination (e.g., course and speed) can be directly checked if it is risky or relatively safe. The search space is narrowed by removing potentially risky options.
- The algorithm takes many aspects into consideration, including the sailing time, distance, safety, the ship's own motion constraints, dynamic obstacles, and compliance with a subset of the COLREGs. Two new factors, relative posture and trajectory smoothness, are proposed and quantitatively calculated in the cost function. The calculation of the heuristic cost is improved to adapt to maritime scenarios.

The remaining parts of this paper are organized as an autonomous sailing system overview, a CA algorithm, a simulation and a comparative study, the discussion and the conclusions.

2. AUTONOMOUS SAILING SYSTEM

2.1. Overview

Figure 1 is an overview of an autonomous ship's system architecture. According to the given missions and the chart information, an initial path is first generated by the global path planning (GPP) module. The autopilot outputs specific steering and propulsion commands to the propulsion and steering of the OS in order to follow the path. The sensor module, including both endogenous and exogenous sensors, aims to collect the current state of the own ship (OS) and the states of the obstacles. Typically, data from different sensors need to be fused to reduce the data uncertainty and deal with data conflict, e.g., fusing the global navigation satellite systems (GNSS) and the inertial measurement unit (IMU) for the OS, fusing the radar and automatic identification system (AIS) for obstacles. A detailed description and surveys of such data-fusion modules are beyond the scope of this paper, and we refer to our previous work (Miao et al., 2020). The local CA module is the focus of this study. Compared to the GPP module, the local CA module is designed to modify the local route according to a dynamic and more detailed surrounding environment. After that, the autopilot will command the ship to follow the new route or commands.

2.2. Collision Avoidance Module

The specific workflow of the proposed CA module is illustrated in Figure 2. Firstly, discrete control options like course offsets and speeds are set in advance as one of the inputs for the CA module. The range and interval of options are based on the sailing scale and associated requirements. With the current states, the motion model of the OS, and the environment disturbances, a list of optional trajectories of the OS is obtained. Static local environment information, such as the shore geometry, water depth, restricted areas, is inputted as a factor for the hazard evaluation. For dynamic obstacles, according to the current and historically observed states, a prediction model is used to obtain the obstacles' trajectory for a certain period in the future. The waypoints provided by the GPP module will be set as the initial nominal destination in the local CA. Cost/loss functions are built and normalized for all these factors, and different weight factors are given according to specific design requirements. Then we select the trajectory combination with the minimum total cost and its corresponding control behaviors.

In this paper, in order to support the local CA module, we assume the following information and capacities are available:

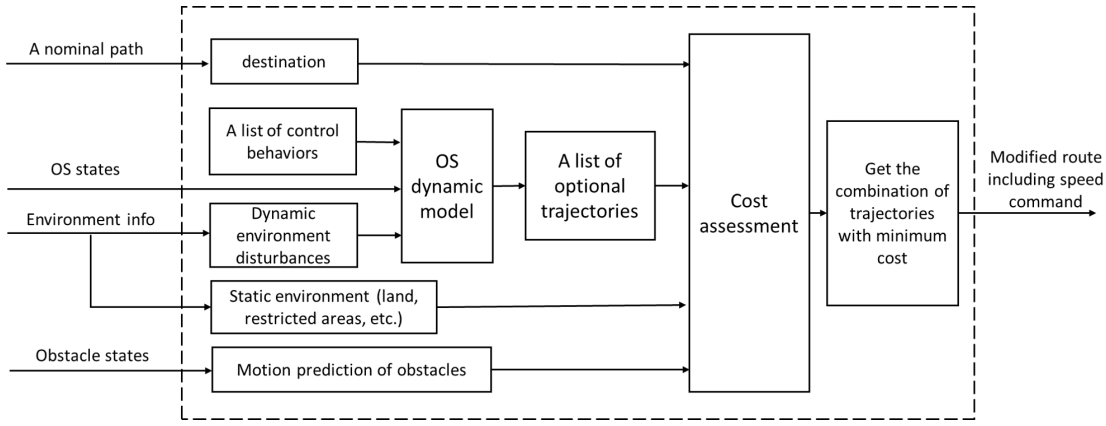


Figure 2: Architecture of Collision Avoidance module.

- Real-time updated states (position, velocity, heading) of the OS, measured by endogenous sensors such as IMU, GNSS.
- A list of obstacle states (positions and velocities), measured by exogenous sensors such as Radar, AIS, LiDAR, camera. Data from different sensors is assumed to be already fused and taken as the truth in this research.
- A desired destination of the OS.
- Kinematic and dynamic models of the OS.
- A prediction model of the obstacles' future trajectory.
- Static environment information (shore geometry, water depth, etc.) from electronic charts.
- Measurements or estimates of wind and ocean current relative speed of the sailing area.

2.3. Dynamic Model of the Ship

A 3-degrees-of-freedom (DOF) dynamic model of the ship is used in this paper based on Fossen's simplification (Fossen, 2011):

$$\begin{aligned} \dot{\eta} &= R(\psi)(\upsilon + \upsilon_0) \\ M\dot{\upsilon} + C(\upsilon)\upsilon + D(\upsilon)\upsilon &= \tau + R(\psi)^T w \end{aligned} \tag{1}$$

where η is the system state that contains the position and heading angle (x, y, ψ), the vector $\boldsymbol{\nu}$ consisting of the linear velocities and the angular velocity (u, υ, r), the vector υ_0 is the current velocity. The input τ represents the commanded thrust and the moments from a PD controller and w is the wind force on the ship. M, $C(\boldsymbol{\nu})$, $D(\boldsymbol{\nu})$ and the $R(\psi)$ are inertia matrix, the Coriolis–centripetal matrix, the damping matrix, and the rotation matrix, respectively.

2.4. Algorithm Implementation

The proposed algorithm functionalizes the local CA module. Within a finite prediction-time horizon, the initial route will be locally modified according to dynamic environments. The local collision-free route will be re-planned within a certain time interval. The selection of the time interval depends on the requirements of the sailing scenarios and is limited by the computational time of obtaining each solution. In order to reduce unnecessary repetition of the calculations, the re-planned route is only accepted when it is significantly better than the old one or there are apparent differences detected between the measurements and the previous predictions. Once a new local route is accepted, it will become the new nominal route for future steps.

The original Hybrid A* has been used successfully in the automotive industry (Dolgov et al., 2008). Though it searches for a solution in a discrete grid map, the states of vehicle can be kept continuous. Both the environment constraints and ship motion constraints can be considered.

Based on HA*, we proposed an improved Hybrid A* (IHA*) using a set of discrete control behaviors as input. The movements during the extension process of the HA* take place in a node map rather than a typical grid map. The nodes of the alternative trajectories are generated dynamically, such that no pre-calculation of the cell cost is needed. The following set of optional control behaviors is a simple example:

- Optional course offsets: -15, 0, 15 (in degrees)
- Optional speeds: full speed (100% nominal propulsion), half speed (50% nominal propulsion)

The course offsets here are relative to the current course of the OS, and the speed options are the percentage of the full propelling speed of the OS. The example gives only three options for the course and one option for speed. In a practical system, the optional control behaviors should be as extensive as the computation time allows to ensure the

solution's accuracy.

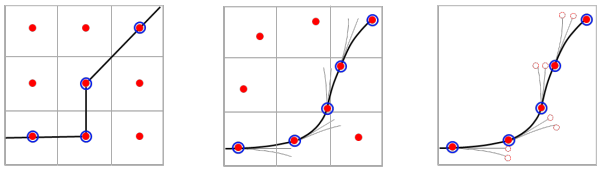


Figure 3: Graphical comparison of search algorithms. Left: A* only visits grid center. Center: Hybrid A* associates a continuous state with each cell, so it can visit all the positions within a cell (Dolgov et al., 2008). Right: improved Hybrid A* searches in a node map.

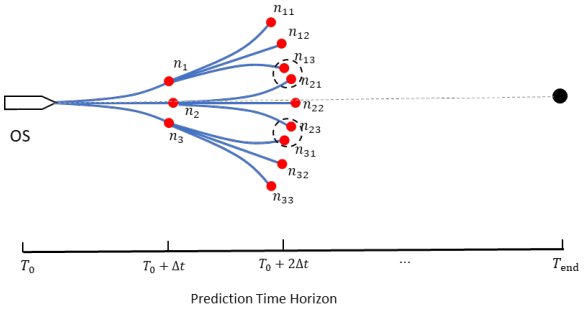


Figure 4: Generated searching nodes and corresponding trajectories tree.

As shown in Figure 4, T_0 is the current time, T_{end} is the end time of the prediction time horizon, $T_{end} - T_0$ is the prediction time horizon, ΔT is the time interval between two prediction points within the time horizon, expressed as

$$\Delta T = \frac{T_{end} - T_0}{N} \tag{2}$$

where N is the desired number of predicted time points. Similar to the number of discrete control behaviors, a more significant number of time points always helps to improve the prediction fineness when the computation time allows. A certain time point in the future within the prediction time horizon is described as

$$T = T_0 + n \cdot \Delta T, \ , n = 1, 2, \dots, N \tag{3}$$

The same set of control behaviors associated with propulsion force and steering angle is given at each prediction step. Endogenous sensors give the measured states of the OS. With the dynamic model of the OS, predicted trajectories are generated correspondingly within the time interval ΔT. The ends of the trajectories are exactly the nodes dynamically created for the solution searching list in traditional A*. The historical cost and the heuristic cost are calculated for each node, and the one with the smallest cost will be added into the Close List (Hart et al., 1968). This list stores all the nodes reached at least once during the expansion searching of A*.

Concerning the dynamic environment, the sensor module with multi-target tracking functionality provides the estimated real-time states of the obstacles. In order to estimate the collision risk in the future, it is necessary to predict the motions of the obstacles. Optimal prediction is used for the motion prediction, based on the observed historical states using a simple kinematic model within the time horizon. Due to the lack of updated measurements, it gives the prediction by iterating the prediction step of the Kalman filter (Särkkä, 2013) and adding the noise of the motion model with a Gaussian distribution. It is noted that the obstacles or other ships will also try to avoid the collision, which leads to a significant offset between the prediction and the actual states. We can use better models considering the dynamics, maneuverability, and intention of the obstacles to improve the motion prediction, which is currently beyond the scope of this paper. Nevertheless, with timely updating of the states of obstacles from the sensor module and re-predicting their motions, a new solution with corresponding control behavior can be found, which guides the OS to avoid the collision. After the prediction, each node's historical cost and heuristic cost can be calculated in terms of different aspects, which is elaborated in Section III.C.

Pre-processing: narrowing search space

Although the improved HA* can generate both a modified route and associated control behaviors using a node map, the size of the search space using a node map is not reduced. On the contrary, the exploration of adjacent nodes on the map increases the search space to some extent. To improve its real-time performance, a collision pre-check method is proposed to narrow the search space. Like the VO, this method excludes nodes with apparently high collision risk in advance. Figure 5 illustrates its essential idea. A and B are two vessels with velocity $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$, respectively. The circles denote the safety domain of each vessel with radius d_A^{safe} and d_B^{safe} respectively, where other obstacles are not allowed to enter. According to the specific sailing scenarios, the safety domain can be represented as other types, such as ellipse and polygon. When the velocity of one vessel is fixed (e.g., $\boldsymbol{v}_B$ is fixed), one can find a velocity range of another vessel (e.g., $\boldsymbol{v}_A$), within which there is going to be a collision in the future. As long as we choose the velocity outside this range, the collision can be avoided. An easy way to obtain the velocity range of vessel A is demonstrated in Fig 5.(b). Firstly, the vessel A is regarded as a point target. Velocity $\boldsymbol{v}_A$, $\boldsymbol{v}_B$ are vectors induced from points A and B, respectively. Secondly, $\boldsymbol{v}'_B$ is generated by translating $\boldsymbol{v}_B$ to the point A, then the relative velocity $\boldsymbol{v}_{AB}$ can be drawn that is induced from $\boldsymbol{v}'_B$. The movement relation can be regarded as the situation that vessel B is relatively static and vessel A is moving with velocity $\boldsymbol{v}'_{AB}$ that obtained by translating $\boldsymbol{v}_{AB}$ to A. A new safety circle with radius d_{AB}^{safe}, the sum of d_A and d_B, is drawn around vessel B. Finally, two tangents are added to the circle from point A and the cone area sandwiched by the tangents is the collision cone of the velocity. If and only if the relative velocity vector $\boldsymbol{v}'_{AB}$ induced from point A falls

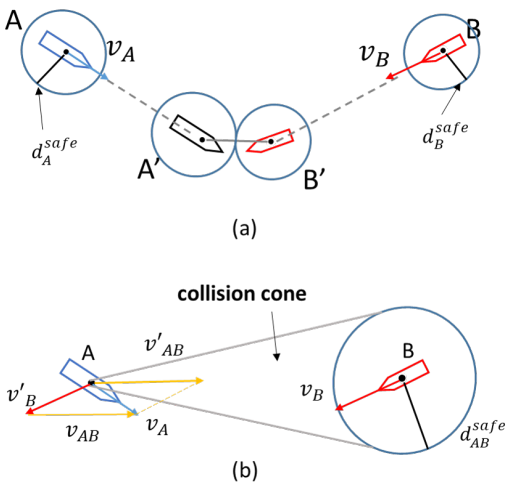


Figure 5: The concept of collision velocity region.

into the cone, it can be regarded that there is a collision in the future.

The set of all the velocity vectors that fall in the cone area is the collision cone set, which is denoted as the set $\boldsymbol{CC}$. Then, the control behaviors that have no collision risk should fulfill:

$$\boldsymbol{V}(\alpha, v) \notin \boldsymbol{CC} \tag{4}$$

Where $\boldsymbol{V}$ is the velocity vector caused by the alternative control behaviors, α is the course and v is the velocity of OS. The way to use this relation to check the potential collision possibility is named as the CVC, and an index of CVC with a certain target ship (TS) is defined as

$$index_{TS}^{CVC} = \begin{cases} 1, \ \theta(\boldsymbol{V}_{AB}, \boldsymbol{AB}) < arcsin\left(\dfrac{d_{AB}^{safe}}{|\boldsymbol{AB}|}\right) \\ 0, \ otherwise \end{cases} \tag{5}$$

where θ is the angle between two vectors and $\boldsymbol{AB}$ is a vector pointing from A to B.

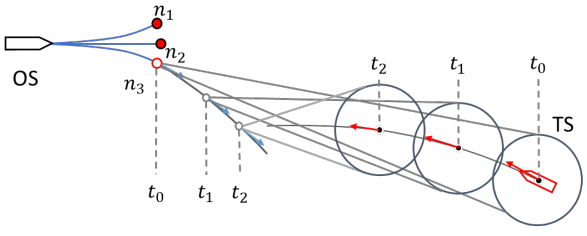


Figure 6: Operating Collision Velocity Check for the discrete time points in the future to exclude the nodes with collision risk

The above check is based on the fixed and known velocities of the OS and the target. To cope with non-linear trajectories according to the prediction model, we sample the predicted states of OS and TS in the discrete future time points and operate CVC for selected future time points. An example is illustrated by the node n_3 in Figure 6: The trajectories and states of the OS and TS can be predicted with the motion prediction model, at discretized prediction time points $t_0, t_1, \ldots, t_m$, (t_0 is the initial time) with time interval Δt in the future. A CVC is done with predicted positions and velocities of the OS and the obstacles for each prediction point. Considering the uncertainty of the prediction over time, we add up the effect of time:

$$index_{TS}^{CVC}(t) = \begin{cases} \dfrac{\Delta t}{t - t_0 + \Delta t} & , \theta(\boldsymbol{V}_{OT}, \boldsymbol{OT}) < arcsin\left(\dfrac{d_{OT}^{safe}}{|\boldsymbol{OT}|}\right) \\ 0 & , otherwise \end{cases} \tag{6}$$

where the d_{OT}^{safe} is safe distance between the OS and the TS. $\boldsymbol{OT}$ is a vector pointing from the OS to the TS. The CVC index with a certain target over the whole predicted time is

$$index_{TS,\,tot}^{CVC} = \sum \left(index_{TS}^{CVC}(t)\right), \qquad t = t_0, t_1, \ldots \tag{7}$$

which should not exceed the threshold in order to keep a control behavior sufficiently safe. Mathematically, CVC can be described as a boolean function:

$$CVC_{TS} = \begin{cases} True & , index_{TS,\,tot}^{CVC} \leqslant \lambda^{CVC} \\ False & , otherwise \end{cases} \tag{8}$$

where λ^{CVC} is the threshold to pass the CVC in terms of each target. The λ^{CVC} is determined by several properties, such as the number of obstacles in the detection range and the number of the total predicted time points. The state 'True' means the current control behavior passes the check and can be left in the search space. It is not necessary to count all the moments. With the above expression, the CVC index becomes small after the first few time points. Thus we can only calculate the time points with the most influence on the result.

For multiple targets, TS_1, TS_2,…,TS_K, a good alternative control behavior should keep OS collision-free or sufficiently safe with all objects. It can be described as

$$CVC_{TS_1} \wedge CVC_{TS_2} \wedge \ldots \wedge CVC_{TS_K} = True \tag{9}$$

For the control behaviors that not fulfill the CVC, we remove them from the search space.

Post-processing

In addition to pre-processing, we can also apply post-processing steps to improve the execution process of the algorithm further. The idea of pruning used here is cutting down similar branches, which contributes only very little to the solution, but vastly increases the size of the search space, after cost calculation. The similarity of all kinds of states (position, velocity, acceleration, etc.) needs to be checked in this node-searching HA* method. The branch will be pruned when similarities are found in all aspects.

The pseudo-code of implementation of this improved HA* is given in Algorithm 1.

Algorithm 1 Improved HA* (n_{start},n_{des})

$List_{open} \leftarrow n_{start}$
$List_{close} \leftarrow \emptyset$
while $List_{open} != \emptyset$ **do**
  $n_p = PopTop(List_{open})$
  $List_{cloce} \leftarrow List_{close} \cup n_p$
  if $GoalReach = true$ **then**
    break
  else
    for $courses = c_1, c_2, \dots, c_i, \dots, c_I$ **do**
      for $speeds = s_1, s_2, \dots, s_j, \dots, s_J$ **do**
        run CVC
        if $CVC = false$ **then**
          continue
        else
          compute total cost $f^{i,j}$
          if $Pruning = true$ **then**
            continue
          else
            $List_{open} \leftarrow List_{open} \cup n_T^{i,j}$
          end if
        end if
      end for
    end for
  end if
end while

2.5. Cost Function

The design of cost functions over paths that yields the desired sailing behavior is usually a major challenge in a practical system. The construction of the cost function in this paper mainly refers to existing literature (Johansen et al., 2016) and (Vagale et al., 2021). Several improvements are made according to the scope of our research and the specific implementation method HA*. New factors and concepts such as relative posture and smoothness are introduced and quantitatively calculated for a better cost estimation. Overall, the cost function $f(n)$ of HA* consists of two parts: the historical cost $g(n)$ and heuristic cost $h(n)$:

$$f(n) = g(n) + h(n) \tag{10}$$

where n is the current node. The cost functions of all aspects are normalized to make a fair comparison.

2.5.1. Historical cost

The historical cost of one node is the minimal accumulated cost from the start point to this node. In this paper, the total historical cost is the sum of the costs for distance, time, safety, COLREGs, and smoothness

$$\begin{aligned} g =& W_{dist} \cdot C_{dist} + W_{time} \cdot C_{time} + W_{safe} \cdot C_{safe} \\ &+ W_{colr} \cdot C_{colr} + W_{smth} \cdot C_{smth} \end{aligned} \tag{11}$$

where W is the weight factor in terms of 5 aspects. These weight factors are set to a value of 1 by default. We can adjust according to the specific scenarios and requirements.

Distance

The distance cost consists of the minimum accumulated distances of the calculated trajectories.

$$C_{dist} = \frac{\sum \left(D_{traj} \right)}{D_{nomi}} \tag{12}$$

where D_{nomi} is the length of the nominal path. The nominal path is initialized as the path from one global waypoint to the next global waypoint output from the GPP module and it will be updated when a new route is accepted.

Time

The time cost is the accumulated sailing time starting from the initial time T_0.

$$C_{time} = \frac{T - T_0}{T_{nomi}} \tag{13}$$

where T is the current time at a certain node and T_{nomi} is nominal sailing time. We use the estimated time sailing through the nominal path with the associated speed.

Safety

The safety cost is defined as a sum of risks of collision and grounding. In terms of collision, the collision risk is the product of collision likelihood factor and collision consequence cost with a certain target, which is expressed as

$$C_{safe,coll}^{TS} = L \times C_{coll}^{TS} \tag{14}$$

where L, C_{coll}^{TS} are collision likelihood factor and collision consequence cost respectively. The collision likelihood factor L takes the time effect, distance as well as the relative posture RP between TS and OS into consideration.

$$L = \begin{cases} inf & , d_{TO}^i \leqslant d_{coll}^{min} \\ (\frac{1}{|T - T_0|})^p \cdot (\frac{d_{safe}^i}{d_{TO}^i})^q \cdot RP & , d_{coll}^{min} < d_{TO}^i \leqslant d_{safe}^i \\ 0 & , otherwise \end{cases} \tag{15}$$

where d_{coll}^{min} is minimum allowable distance between OS and TS, which is set as a hard constraint. The exponent p and q describe the time effect and distance-associated risk, respectively. The distance d_{safe} should be chosen large enough to be compliant with COLREGs rule 8, 16, i.e. to make the substantial action to keep well clear. The collision consequence cost mainly depends on the relative velocity and some properties of OS and TS such as type and size, and OS's right to stay on or responsibility to keep out of the way.

The newly introduced factor RP aims to correct the impact caused by the relative posture when two ships encounter

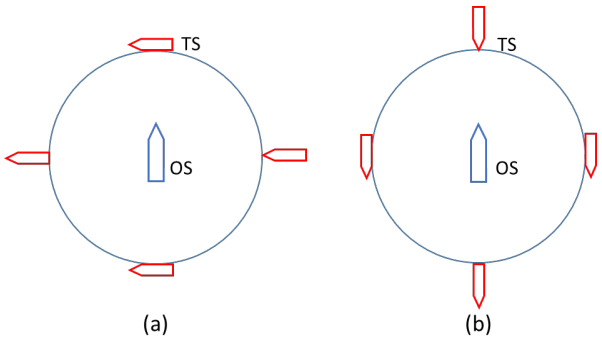


Figure 7: The different postures when two ship encounter.

each other. As illustrated in Figure 7, different relative postures bring the OS different collision threats and costs, even when the distance, time, and relative velocity are the same. In particular, in the field of last-minute collision avoidance, a parallel collision posture usually minimizes the collision damage. However, this factor is ignored by most researchers. It is denoted as

$$RP = 2 \cdot \left(1 - \left|sin\left(\frac{\theta}{2}\right)\right|\right) \tag{16}$$

where θ is the angle between relative velocity vector and the vector pointing from the OS to the TS.

For multiple targets, the collision safety cost is the sum over all targets.

$$C_{safe,coll} = \sum \left(C_{safe,coll}^{TS_k}\right), \ k = 1, 2, \ldots, K \tag{17}$$

For the grounding, the safety cost is defined as

$$C_{safe,grd} = \begin{cases} inf & , d_{grd} \leqslant d_{grd}^{min} \\ \dfrac{d_{grd}^{max} - d_{grd}}{d_{grd} - d_{grd}^{min}} & , d_{grd}^{min} < d_{grd} \leqslant d_{grd}^{max} \\ 0 & , otherwise \end{cases} \tag{18}$$

where d_{grd}^{min} is the minimum allowable distance to the shore, d_{grd}^{max} is the maximum distance that has grounding risk, d_{grd} is the minimum distance from the current position to the shore. The d_{grd}^{min} is set as a hard limitation, and its determination requires the information from the chart, such as the water depth. The set of d_{grd}^{max} introduces an inflation area where the cost gradually reduces to zero. The total safety cost is the sum of costs of preventing the OS from all the hazards:

$$C_{safe} = W_{safe,coll} \cdot C_{safe,coll} + W_{safe,grd} \cdot C_{safe,grd} \tag{19}$$

Similarly, without specific requirements, we set both weight factors to 1.

COLREGs-constraints

The Rules 6, 8, 13, 14, 15, and 16 (seen in Appendix) of COLREGs are taken into consideration in this study. Rules 8 and 16 are considered by adding associated constraints during the design of the function and selection of parameters. To be compliant with Rule 8, the prediction horizon $T_{end} - T_0$ is set to be significantly larger than the time to make a substantial change of course and speed. The safe distance d_{safe} is set in the safety cost calculation. The reduce-speed and stop options are included in the input discrete options. The design of the safety cost makes it extremely high when there is a risk of collision, which means the action to avoid collision will always be taken even if rules are violated. In the weighted cost function, the speed-change cost has a higher weight than the course-change cost, such that the OS is prioritized to change course, in compliance with both Rules 8 and 16.

The compliance with Rules 6,13,14,15 is designed as a part of the cost function, which is an indicator function as follows:

$$C_{COLR}^{i} = \begin{cases} 0 & , COLREGs \ compliant \ with \ target \ i \\ 1 & , otherwise \end{cases} \tag{20}$$

The penalty cost will arise when any rules are violated. The violation conditions of each rule are listed as follows:

- Rule 6 - Safe speed: It is violated when the current selected speed is larger than the safe speed restriction.
- Rule 13 - Overtaking: It is violated when the TS and OS is too close during overtaking.
- Rule 14 - Head-on: It is violated when the TS and OS stands on each starboard during a head-on encounter.
- Rule 15 - Crossing: It is violated when the TS is on the OS's starboard side during a crossing encounter.

The specific judgment conditions for each encounter situation are based on the Johansen et al. (2016). Besides, an additional condition - Approaching, is added for the violation of Rules 14 and 15. A TS is said to be approaching the OS if

$$\theta(\boldsymbol{V}_{OT}, \boldsymbol{OT}) < 90° \tag{21}$$

where 90° could be replaced by a more suitable angle depending on the velocity and type of target ships. The violation of Rules 14 and 15 holds only when the TS and OS are approaching each other.

Smoothness

The concept, smoothness of trajectory, is introduced and quantitatively calculated in this session. For the CA system on manned ships, the smoothness of the local collision-free path influences the comfortableness of crews and passengers. Frequent and sharp acceleration and turning also increase the fuel cost. People on the other surrounding

vessels are anticipating the movements of the OS. Frequent heading/course alternations make it hard to anticipate on, as such this could lead to unwanted scenarios. With such concern, we introduce the smoothness cost to describe the cost caused by such changes. It consists of two parts: course change cost and speed change cost expressed as

$$C_{cc} = \left| \frac{max\left(\left|\Delta\alpha - \lambda_{cc}\right|, 0\right)}{K_{cc} \cdot \lambda_{cc}} \right| \tag{21}$$

$$C_{sc} = \left| \frac{max\left(\left|\Delta v - \lambda_{sc}\right|, 0\right)}{K_{sc} \cdot \lambda_{sc}} \right| \tag{22}$$

where $\Delta\alpha$ and Δv are the maximal change of course and speed on each generated trajectory segmentation caused by the control behaviors, K_{cc} and K_{sc} are scale factors to adjust the effect brought by the changes. These two factors may depend on several properties, such as the type of the OS and its size. For small ships, they are usually more agile and have better maneuverability, the same course-change rate or accelerations lead to less impact of the total cost compared to large vessels. Some slight changes are normal during the sailing. Thus, when the change is sufficiently small, no cost is generated. The entire smoothness cost is

$$C_{smth} = W_{cc} \cdot C_{cc} + W_{sc} \cdot C_{sc} \tag{23}$$

Weight factors are added to adjust the priority of each factor, which is set to 1 by default.

2.5.2. Heuristic cost

The heuristic cost is essentially a prediction of cost to the goal in the rest prediction time. Good construction of heuristic cost can guide the expansion direction, which shortens the searching process. But, due to the relative scarcity of information, accurate estimation is rather difficult. We estimate the heuristic cost considering the aspects of distance, time, and safety.

Firstly, we re-use the CVC index to indicate the heuristic safety cost. Even though the options with high collision risk have been abandoned during the CVC, the remaining options are not 100% safe. The CVC index provides us with an estimation of potential collision risk. That no extra computing time required required is also an important reason. Thus, the heuristic safety cost is simply using a normalized CVC.

$$C_{heur,safe} = \frac{index^{CVC}_{TS,\,tot}}{\lambda^{CVC}} \tag{24}$$

The distance is estimated based on current state of OS the environment given by

$$C_{heur,dist} = \frac{K_{heur,dist} \cdot D_{OD}}{D_{nomi}} \tag{25}$$

where $K_{heur,dist}$ is the scale factor determined by the current state (position, heading, speed) of the OS, D_{OD} is the distance between the OS's current position and the destination.

The heuristic time cost is estimated based on the distance and current speed of the OS, which is expressed as

$$C_{heur,time} = \frac{K_{heur,\,d} \cdot D_{OD}}{\left(K_{heur,\,t} \cdot \left|v_c\right| + \left(1 - K_{heur,\,t}\right) \cdot \left|v_{nomi}\right|\right) \cdot T_{nomi}} \tag{26}$$

where $K_{heur,\,t}$ is the function of time and v_c is the current speed.

$$K_{heur,\,t} = \frac{T - T_0}{T_{end} - T_0} \tag{27}$$

The entire heuristic cost is

$$h = C_{heur,\,safe} + C_{heur,\,dist} + C_{heur,\,time} \tag{28}$$

3. SIMULATION & COMPARISON

This section illustrates the performance of the proposed method by simulation and a real case study. At the same time, to illustrate the impact of the real-time performance of using the CVC, both results using improved the Hybrid A* only and using the improved Hybrid A* with the CVC are given. The same control behaviors are given as follows:

- Optional courses: -45, -30, -15, 0, 15, 30, 45 (in degrees)
- Optional speeds: full speed (100% nominal propulsion), half speed (50% nominal propulsion), stop speed (0% nominal propulsion)

A wide range of simulated multi-target cases is considered, from simple encountering scenarios to complex and mixed encountering scenarios, from single target-moving-mode to multiple target-moving-mode combinations. The proposed algorithm operates once for each case because these cases are static which are like frames extracted from a dynamic sailing process. After that, the method is also applied to a real multi-target case using logged Radar and AIS data (the data here is pre-fused referring to our previous work (Miao et al., 2020)) during a sea trial. The logged data, including Radar AIS and GPS, is constantly sent by a virtual user datagram protocol (UDP) sender according to the logged timestamp with a frequency of more than 1 Hz. A 20-minute simulation runs in Simulink, which keeps receiving and updating the data with a constant sampling time of 1 second. The proposed algorithm is called once every second. To ensure there is no latency, the algorithm should finish the calculation and provide a modified solution within the sample time interval.

3.1. Simulated Scenarios

We set the prediction time horizon as 800 seconds, the time of straight-line sailing through the setting scenarios at full-speed 19.4 knots (10 m/s), with a 40-second time interval for all simulated cases. The parameter assumptions of the OS used during the simulation are set in Table 1. Three methods: Johansen et al. (2016) (named as Johansen's method below), the proposed improved HA* without CVC

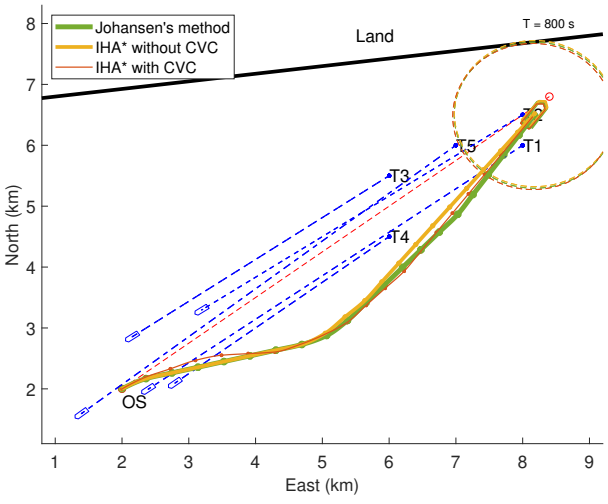
Table 1
Own ship's parameters in simulations

Parameters	Value	Unit
length	120	m
breadth	24	m
draught	7	m
full speed	10	m/s
min turning radius	400	m

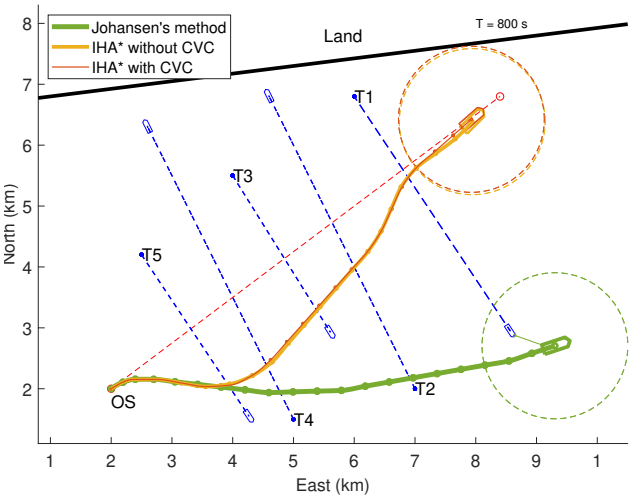
and with CVC, are implemented separately and illustrated together in each scenario to make a comparison. Johansen's method is selected as it also uses discrete control behaviors as the inputs, and the proposed method is also partly inspired by it. Besides, it has a good real-time performance. One thing we have to be aware is that the control behavior is fixed after every single running of basic Johansen's method, which is caused by its mechanism. To plan a complete route to the destination, one can either add the possibility of altering control behavior during planning or use replanning. The former leads to an exponential growth of the computation time. We use replanning with the same time interval for Johansen's method to make it a complete route plan. It means $7 \cdot 3 = 21$ scenarios for each step and $21 \cdot 20 = 420$ in total are calculated during each complete computation. The number of scenarios in Johansen's method impacts the computation efficiency, in a similar way to the concept of the node number using IHA*. We also use the IHA* without the CVC as another comparison to show the influence of the CVC. All the parameters are the same for all the methods and scenarios. The scenarios and results are illustrated in Figures 8-10 representing the snapshots of the Scenarios 1-7 at the end of the time horizon. Figures 12 and 13 illustrate the progression of route planning in time. For every scenario, 50 simulations are done, and the average computing time for operating CA calculation and node numbers using IHA* are listed in Table 2. The size of the search space in Scenario 7 is visualized using the searched nodes or calculated scenarios, illustrated in Figure 14. The statistic results of the computation time in all scenarios are illustrated in Figure 15. The whole path length and the estimated subsequent sailing time to the destination are listed in Table 3. The minimal distance between the OS and TSs during the whole planned route in Table 4. The following symbols and color codes are applied:

- The red solid point at the bottom left corner is the current position of OS, which is also the starting point of the whole local collision avoidance process. The destination (nominal goal) locates at the top right corner, illustrated as a red hollow point. The nominal path of OS is illustrated as a red dashed line connecting the start point and destination.

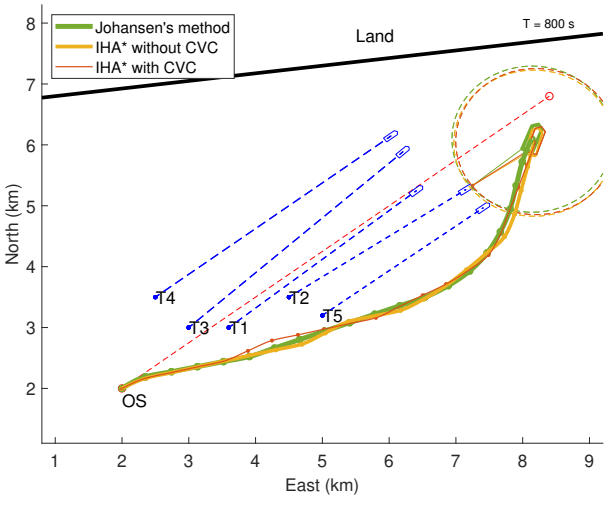
- According to the three methods (the method of Johansen et al. (2016), the IHA* without and with CVC),



(a) Scenarios 1: All head-on encounter

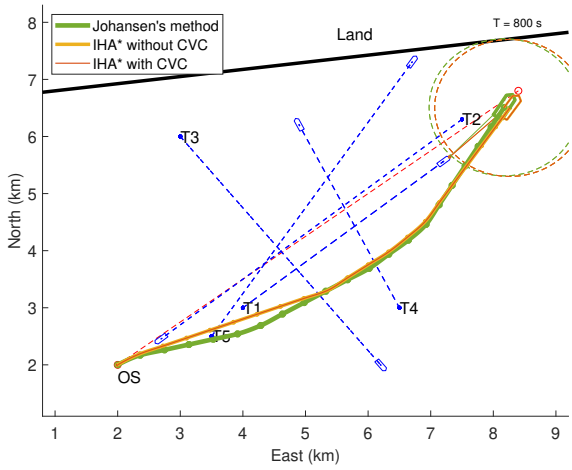


(b) Scenarios 2: All crossing encounter

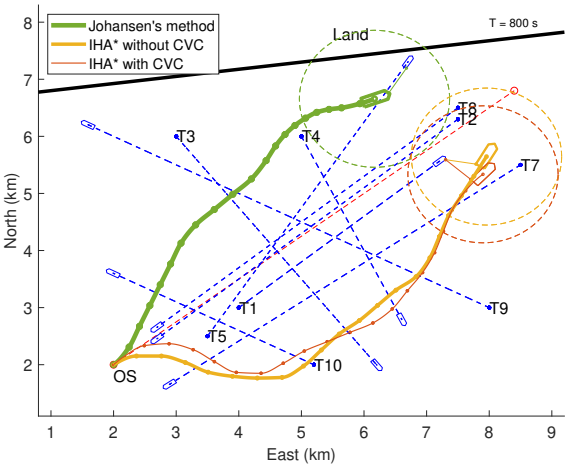


(c) Scenarios 3: All overtaking encounter

Figure 8: Simulation scenarios with single encounter scenarios.

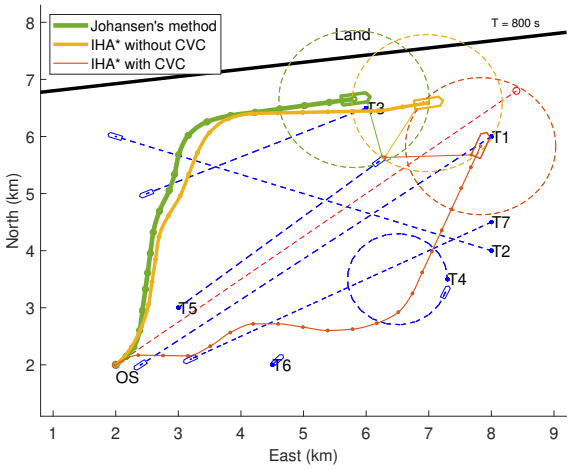


(a) Scenarios 4: Mixed encounter scenarios with 5 targets

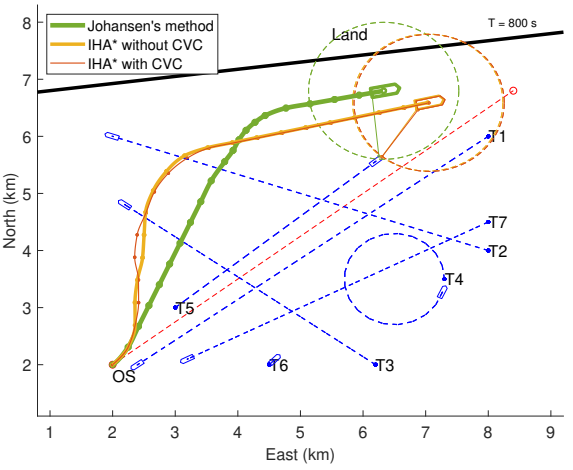


(b) Scenarios 5: Mixed encounter scenarios with 10 targets

Figure 9: Simulation scenarios with mixed encounter scenarios.



(a) Scenarios 6: More target moving modes



(b) Scenarios 7: More target moving modes

Figure 10: Simulation scenarios with mixed encounter scenarios and more target moving modes.

the OS, OS's trajectory, OS safety range, and the current minimum distance between OS and TSs are uniformly denoted in green, orange, and red, respectively. Furthermore, to tell the difference when results are overlapped, the trajectories are also given with 4, 2, and 1 line widths, respectively. A ship-shape expresses the final state of the OS. The dashed circles around OS are the safety range.

- All targets, target trajectories are in blue. Trajectories are illustrated as dashed lines, starting from solid blue points.
- The bold black line is the land-sea boundary, and the land side is pointed out.

Since many studies have provided a validation of A* in the maritime CA context (Kim et al., 2014; Singh et al., 2018), we no longer focus a single-target scenarios. Instead, our research focuses on complex multi-target scenarios. However, the complexity of the multi-target scenarios also differs a lot. We start simulations with relatively simple cases. Figure 8(a)(b)(c) illustrates the CA solution in a five-target scenario with a single encounter situation: only head-on, only crossing, and only overtaking, respectively. Though the encountering modes are the same, in each case, five targets are set with different starting points and velocities. The same factor is that they all have a linear moving mode, which is relatively predictable. The results first show that a collision-free solution with continuous states of the OS can be successfully obtained with the improved Hybrid A*.

In the head-on and overtaking scenarios, three methods all generate solutions sailing at the starboard side with similar path lengths, which are 8385 m, 8353 m, and 8404 m in the head-on scenario, and 8801 m, 8798 m, and 8763 m in the overtaking scenario, respectively. In the crossing case, though Johansen's method and IHA* both suggest going starboard, the solution given by Johansen's method

Table 2
Mean computation time of all scenarios using three methods, and node number using the IHA* with or without the CVC.

Scenarios	Johansen	IHA*		IHA* with CVC	
	time(s)	time(s)	nodes	time (s)	nodes
1-Fig8(a)	0.4650	0.4825	638	0.2212	282
2-Fig8(b)	0.7926	1.1773	774	0.2048	276
3-Fig8(c)	0.9061	1.8385	781	0.5903	364
4-Fig9(a)	0.9344	0.9021	384	0.3949	222
5-Fig9(b)	0.8158	6.9560	2332	0.1921	371
6-Fig10(a)	0.9377	5.8816	2614	0.3421	253
7-Fig10(b)	0.6805	2.3437	1622	0.5380	754

Table 3
Path length and left sailing time using three methods. 1: Johansen's method, 2: IHA* without CVC, 3: IHA* with CVC.

Scenarios	Path length (m)			left sailing time (s)		
	1	2	3	1	2	3
1-Fig8(a)	8385	8353	8404	40	37	42
2-Fig8(b)	12297	8633	8597	288	65	61
3-Fig8(c)	8801	8798	8763	81	82	79
4-Fig9(a)	8349	8334	8329	36	35	35
5-Fig9(b)	9281	10076	10400	352	210	242
6-Fig10(a)	9700	9472	9144	267	149	117
7-Fig10(b)	9187	9392	9361	220	141	139

reduces the speed at the beginning to avoid T5 while the solutions from IHA* choose to sail with full speed. Besides, Johansen's method keeps going starboard instead of altering the course back to pass the targets 3 and 5. It follows the COLREGs rule with a sacrifice of the path distance and subsequent sailing time. Since a full speed is kept and the course is altered back to the goal, the position of the OS by the end of the routes using IHA* is closer to the destination rather than using Johansen's method. Two solutions generated by IHA* with and without the CVC are similar. In terms of the computational time, Johansen's method can obtain results in three scenarios with 0.4650 s, 0.7926 s, and 0.9061 s, and IHA* with CVC took 0.2212 s, 0.2048 s, and 0.5903 s, respectively. The computational time of IHA* without the CVC is relatively slower, which takes 0.4825 s, 1.1773 s, and 1.8385 s, respectively. The computational time of IHA* using the CVC, compared to the one not using, is improved by an average of 68%. The reduction of computational time benefits from the reduction of the node number, which is from 638 to 282, from 774 to 276, and from 781 to 364, respectively. The minimum distances between the OS and TSs using the three methods are also similar: 665 m, 636 m, and 648 m in the head-on case, 618 m, 610 m, and 603 m in the crossing case, and 636 m, 607 m, and 620 m in the overtaking case, respectively.

The simulations with mixed encounter scenarios are shown in Figure 9. We put multiple targets with different encountering scenarios (head-on, crossing, overtaking) in one scenario. In such a mixed scenario, all three methods generate a collision-free route. For the five-target case illustrated in Figure 9(a), the three routes are all similar with slight difference in the beginning. The minimum distance is a bit greater with Johansen's method than using IHA*, which is 636 m, 614 m, and 614 m, respectively. However, the time using IHA* with and without the CVC (0.9021 s and 0.3949 s) is shorter than using Johansen's method (0.9344 s). In the ten-target scenarios, there is a bigger difference of the routes using Johansen's method and IHA*. The Johansen's method generates a route sailing from the port side that is close to the shore, while both the IHA* with and without the CVC

Furthermore, we extend the motion models of the introduced targets by adding a target (T4) turning a circle and a target (T6) keeping still. The purpose is to test the robustness of the proposed method by making the motion of targets less predictable. Compared to the previous scenarios, it takes a longer time to obtain the solution using IHA* with the CVC 5.8816 s in Scenario 6 - Figure 10(a), and 2.3437 s in Scenario 7 - Figure 10(b), respectively). There is no significant increase of the computation time using IHA* with the CVC, which is 0.3421 s in Figure 10(a) and 0.5380 s in Figure 10(b). In Figure 10(a), although the routes given by Johansen's method and the IHA* without CVC are similar, the Johansen's method reduced the speed at the beginning, which causes a longer sailing time than using the IHA*, while the IHA* with CVC keeps full speed. The solution using IHA* with CVC goes to starboard. This selection leads to a slightly smaller minimal distance (600 m, compared to 646m using Johansen and 610 m using IHA* without CVC), while achieves a shorter path length and left sailing time. For targets 4 and 6 (T4 and T6), a collision-avoidance action from the starboard side is given by the IHA* with CVC. The setting of targets in Scenario 7 is slightly different from the one in (a) (the trajectory of T3 changes), while the route is very different. All three methods select to sail from the port side in this scenario. Due to the path change of T3, the route given by the IHA* with the CVC also changes accordingly to the port side to avoid it and then is similar to the route using the one without CVC. After the altering course at the beginning, the solution given by Johansen's method keeps the course until T=400s and then turns to starboard with reduced speed. By contrast, the solutions of IHA* give a larger course change at the beginning and turns back to the destination earlier. The total path lengths do not vary much using three methods (9187 m, 9392 m, and 9361 m), and the computation time is 0.6805 s, 2.3437 s, and 0.5389 s, respectively.

3.2. Real Case Study

The improved Hybrid A* with CVC is also applied to a simulation with logged data during a real sea trial illustrated in Figure 11(a). The sea trial was carried out on the coastal

Table 4
Path length and minimum distance during sailing using different methods. 1: Johansen's method, 2: IHA* without CVC, 3: IHA* with CVC.

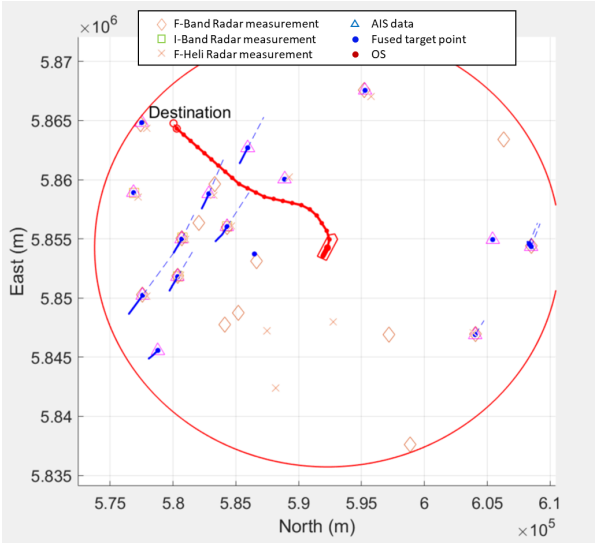
Scenarios	minimum distance (m)		
	1	2	3
1-Fig8(a)	665	636	648
2-Fig8(b)	618	610	603
3-Fig8(c)	636	607	620
4-Fig9(a)	636	614	614
5-Fig9(b)	700	610	852
6-Fig10(a)	646	610	600
7-Fig10(b)	668	602	603

water near the Netherlands, which is around 20 km away from the shore. The logged data is associated with the sailing period after 30-min departing from 'Den Helder' harbor. It is seen in Figure 11(a), the area is busy within the selected range (8 nautical miles) and there is far away from the shore. So there are no shore restrictions and no other restrictions, the ship has sufficient maneuvering space.

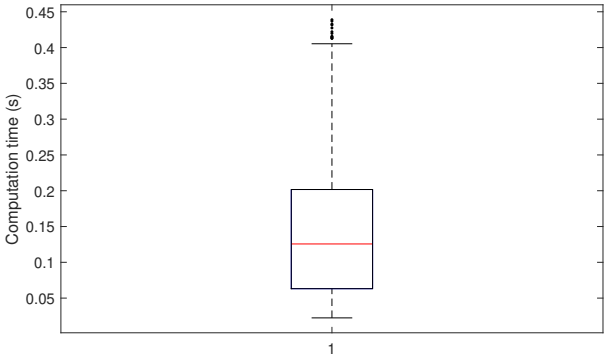
In the simulation, the logged data, including Radar AIS and GPS is replayed, by using a virtual UDP sender according to the timestamp. A UDP receiver keeps receiving and updating data with a constant sample time of 1 second. The CA algorithm is called once every second. The data from different sensors are filtered with a Kalman filter and fused. The corresponding preparing work like aligning the timestamp is done in advance.

The circle here denotes the sensor's detection range; at the center of the circle is the current position of OS. The symbols - diamond, square, cross, and triangle express the measurements from F-Band Radar, I-Band Radar, F-Heli Radar, and AIS, respectively. The blue points are the positions of targets after fusion. The dashed lines describe the velocity of the targets.

During this case, a collision-free path (red line) can be generated successfully for each frame using the proposed algorithm. According to the log period, a re-planning of the path is done with a constant frequency (every 1 second). Due to the sailing scope of this case, 1 HZ is sufficient for the system to act. To stabilize the entire route, the re-planned path is only accepted when it is significantly better than the old one or the new updated environment information is different from the prediction. The computation time of each time step is recorded, and the statistic information is shown by the boxplot in Figure 11(b). There is no latency with the CVC as the pre-check during the entire simulation.



(a) Real-time simulation with logged data in a sea trial

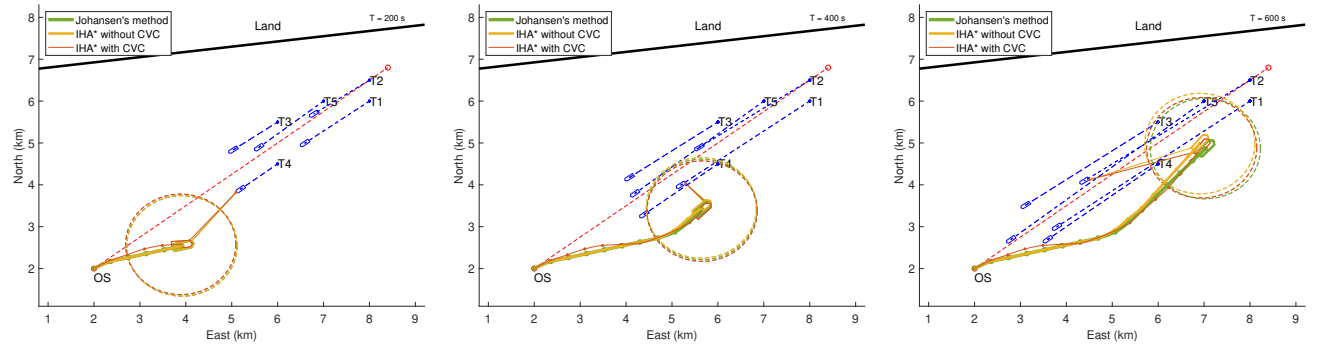


(b) Boxplot of the computation time of real case study

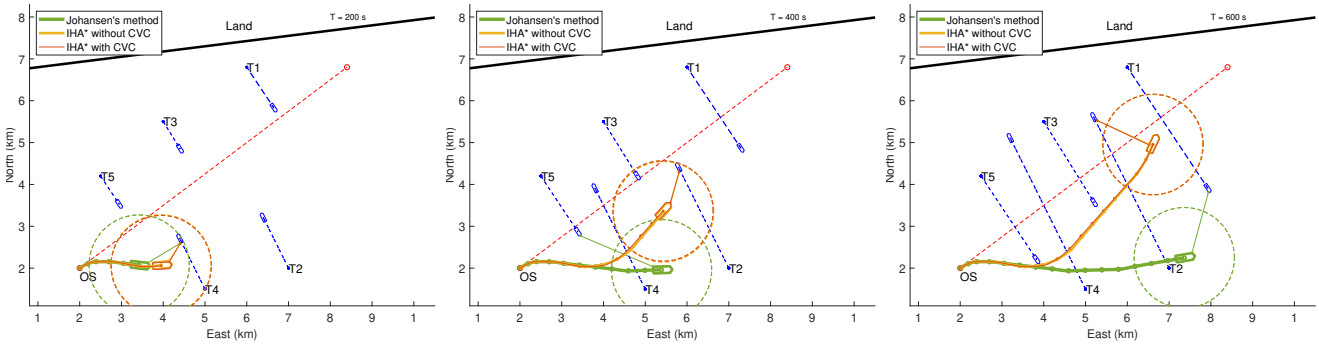
Figure 11: Real case study results

4. DISCUSSIONS

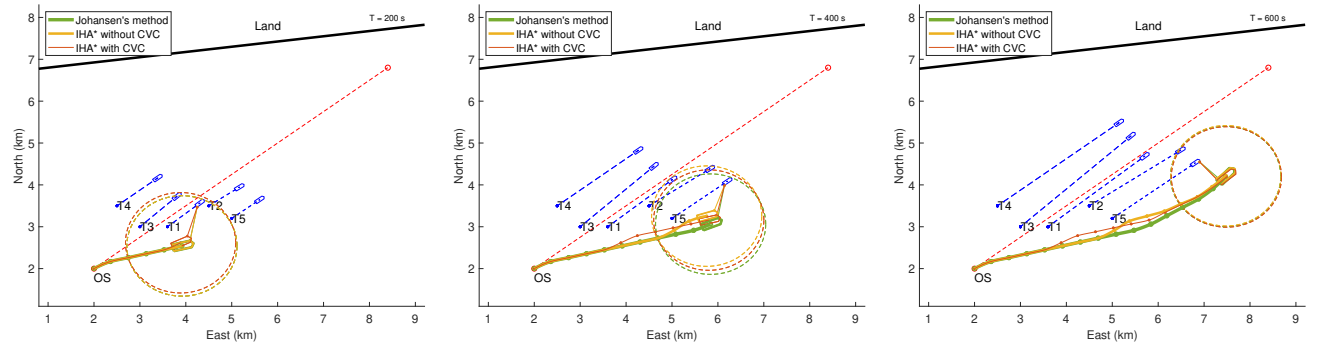
The proposed CA method can provide a collision-free solution in all the simulated cases. A smooth path with continuous state change can be generated, benefiting from searching in a node map. Furthermore, a pre-collision check method, the CVC, is proposed and applied to speed up the computation time of this improved Hybrid A*. The collision risk of each discrete control behavior can be estimated in advance, such that the options with an obviously high collision risk can be excluded from the search space in advance. As seen from the results, the method, IHA* without CVC, requires a longer computation time, especially in complex scenarios. For example, the average computation time can be up to 6.9569s in Scenario 6 and 5.8816s in Scenario 7. This is precisely the common problem we mentioned before, limiting the application of A* algorithms in dynamic collision avoidance. Using the CVC, all scenarios' computation time is shortened considerably (average 76%) with only slight changes (the most significant difference occurs in Scenario 6, where a safe and effective solution is still found) in the final solution. In the best case, the



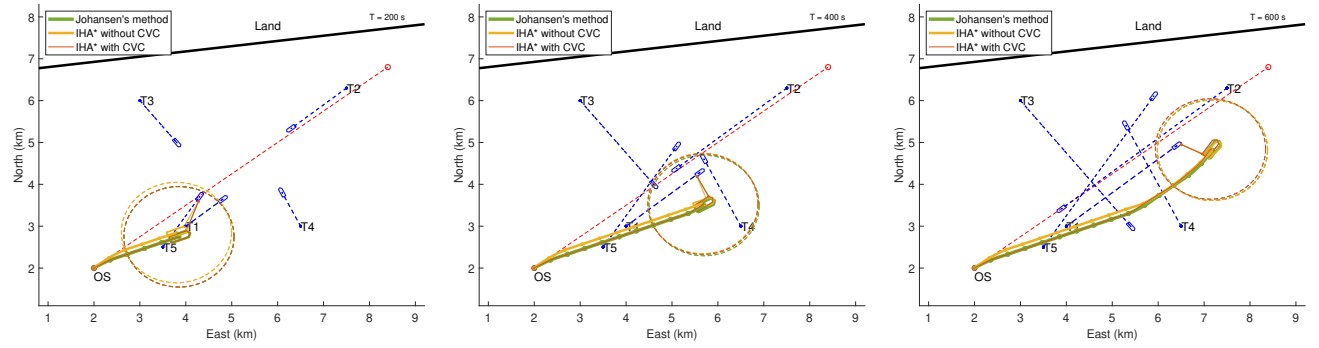
(a) Scenario 1: All head-on encounter



(b) Scenario 2: All crossing encounter

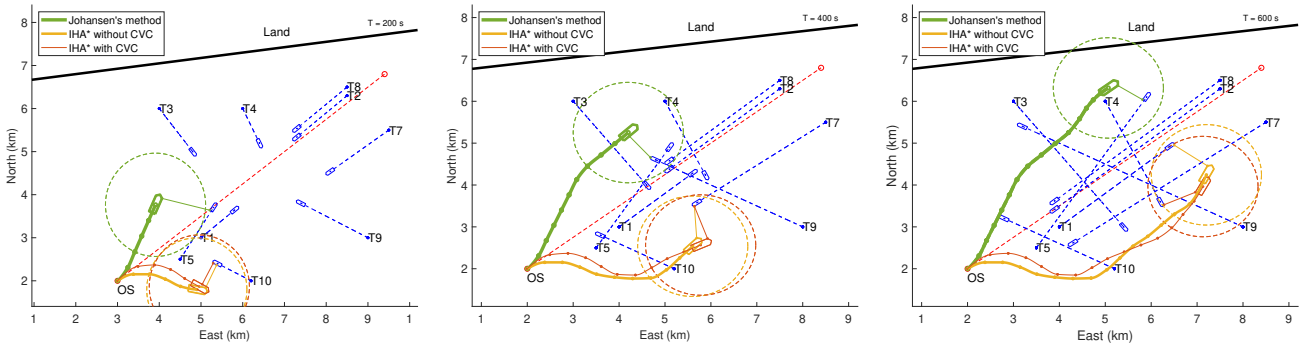


(c) Scenario 3: All over-taking encounter

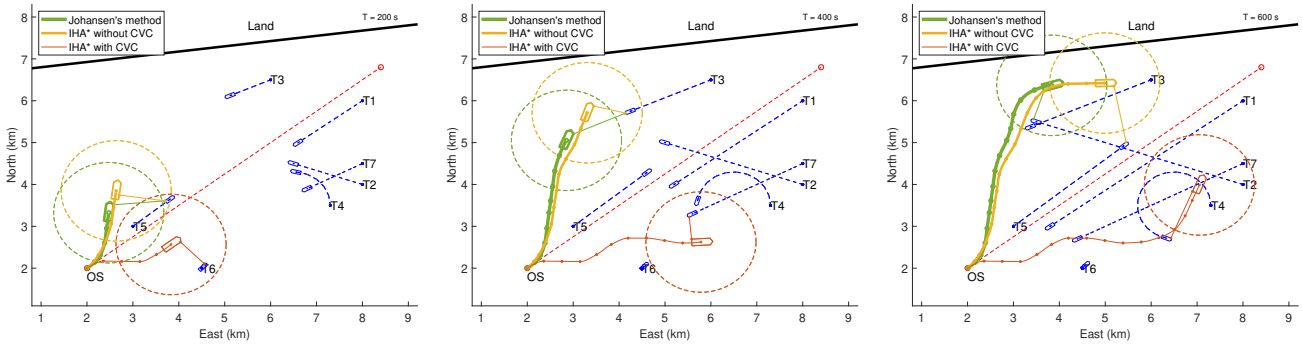


(d) Scenario 4: Mixed encounters with 5 targets

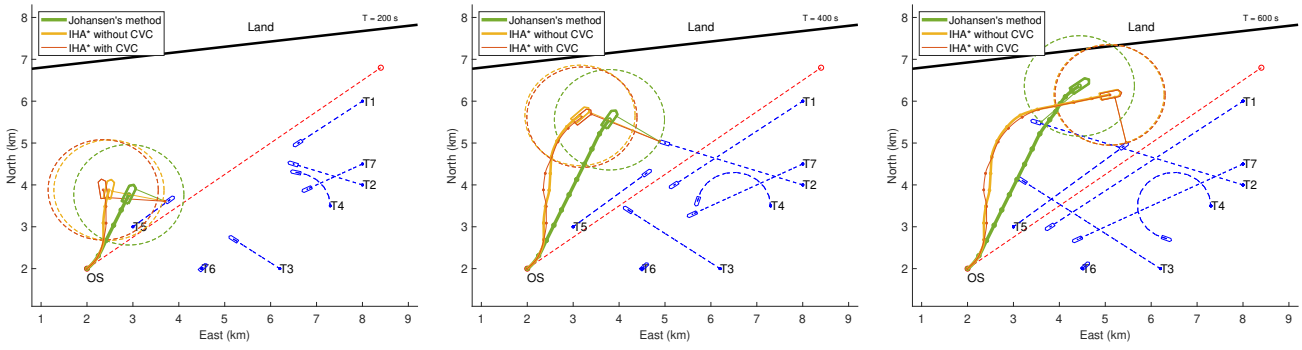
Figure 12: Collision avoidance progression in time of Scenarios 1-4. T = 200s, 400s, 600s, respectively



(a) Scenario 5: Mixed encounters with 10 targets

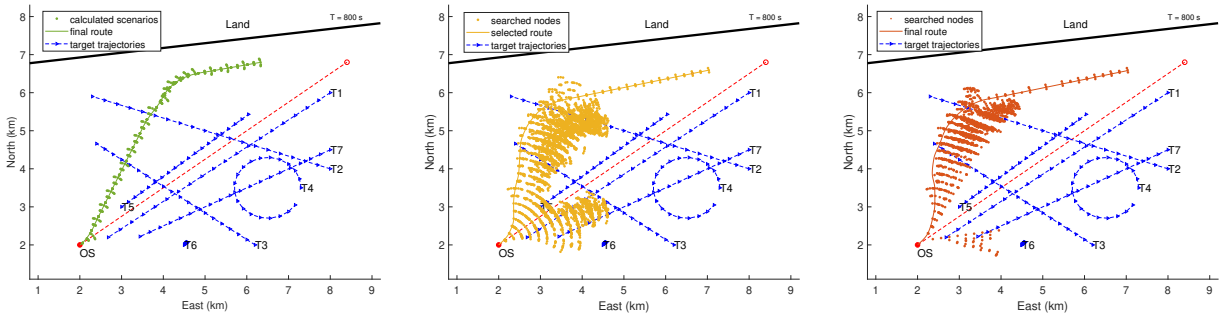


(b) Scenario 6: More target moving modes - 1



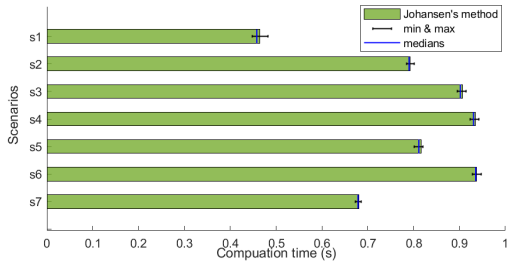
(c) Scenario 7: More target moving modes - 2

Figure 13: Collision avoidance progression in time of Scenarios 5-7, T = 200s, 400s, 600s, respectively

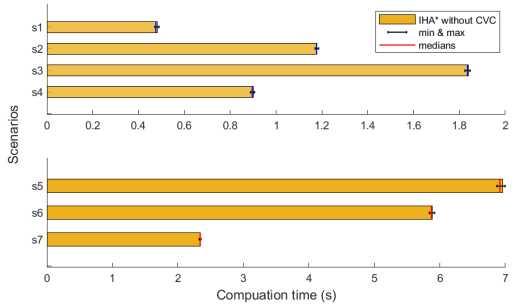


(a) Search space of Johansen's method (b) Search space of IHA* without CVC (c) Search space of IHA* with CVC

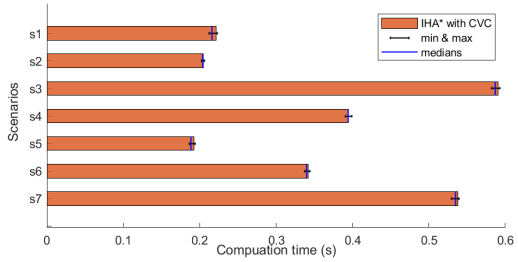
Figure 14: Search space size of three methods in Scenario 7



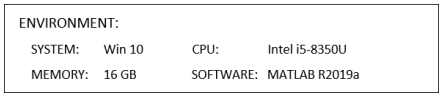
(a) Johansen



(b) IHA* without CVC



(c) IHA* with CVC



(d) Environment information

Figure 15: Computation time statistics of different scenarios

computation time is shortened by 97% (Scenario 5). With this pre-check method, the real-time performance of HA* is indeed greatly improved. Benefiting from increased computation efficiency, with the same computation power, a more accurate solution can be found with a richer set of control behaviors within the same sample time. Alternatively, a higher computation frequency can be realized with the same course and speed options.

As a comparison, Johansen's method can also give an effective solution with a relatively high real-time performance. In Scenarios 1 and 3, there is no significant difference between the solutions provided by IHA* and Johansen's method. In the crossing case (Scenario 2), the solution given by Johansen's method keeps the course to starboard until passing T5. It follows the COLREGs, but the price is the sacrifice of sailing distance and time. By contrast, IHA* gives a relatively reasonable way, which benefits from setting the heuristic cost of the A* algorithm. In complex scenarios (Scenarios 4-7), IHA* gives a similar but relatively better solution. E.g., the routes are identical in Scenario 4 and 7, but the total path length is shorter using IHA*. In Scenarios 6, though the IHA* with CVC goes to a different side and has more turning actions than Johansen's method, the minimal distance is still considered safe according to our setting and the sailing distance and time are even less. Furthermore, there is a speed reduction of the solution using Johansen in Scenario 5. By contrast, the IHA* keeps the full speed. A shorter computation time can be achieved using the IHA* with the CVC than Johansen's method in all scenarios. Meanwhile, the solution is similar or better in safety, sailing distance, and time. Theoretically, the solution given by the IHA* is more optimal than Johansen's method with the setting in this study. Because it can search backward and check all the possibilities during the solution searching process. However, we must be aware that the two methods have inherent differences. The cost functions are designed in different structures, and solution searching mechanisms also differ.

Besides, according to the number of options and replanning times we set, Johansen's calculated scenario number is fixed to 420 in all scenarios. It is indeed seen from the results that the computation time is relatively stable. Nevertheless, the specific time can be different due to the calling of different functions in different scenarios, e.g., when TSs and OSs are close, a polygon distance is calculated instead of point distance, which impacts the total computation time. Note that the fixed number is only used for our comparison, which can differ in other cases. The size of the search space, i.e., the number of the nodes, is not fixed using the IHA* method. It can be small, e.g., in Figure 9(a), when the design of the heuristic cost is accurate. However, it is larger than the one using Johansen's method in most cases. So, it usually takes longer to use the IHA* without the CVC than when using Johansen's method. Using the CVC, the number of nodes can be significantly reduced, leading to a shortened computation time. Furthermore, the computation time of Johansen's method is an accumulation of 20-time replanning according to our setting. If we only consider the safety and ignore the long-term impacts, it should be faster than the proposed method. In addition, one can increase the accuracy of the solution using Johansen's method by adding the possibility of changing control behaviors. However, it will lead to the exponential growth of the computation complexity, such that it is not selected for the comparison in this paper.

Although the computation time of the solutions using the CVC is shortened in all the scenarios, theoretically, an improper parameter setting might lead to a longer time using the CVC than when not using it. The reason is that

good options can be removed inappropriately. Thus follow-up studies on how to select a proper CVC threshold are required. Another reason for this potential problem could be the inappropriate safety domain. A circular safety domain is applied to all the targets based on our current research scope - coastal waters, which might not be suitable for other scenarios like inland shipping. A sufficiently large safety radius has to be selected to ensure safety in any situation. The ships are assumed to be polygons to calculate the minimum distance when they are close enough. To further describe this domain, we can use more specific shapes such as an ellipse, a polygon, or even an actual shape. Making these parameters self-adjustable to different scenarios is one of our future work.

Some COLREGs rules like 8, 16 are considered during the parameter tuning. Some are considered as a part of the cost function, which means COLREGs compliance is not designed as a hard constraint. COLREGs rules can be violated when the total cost can be much smaller as a result. By giving more weight to the COLREGs cost, the violation of the COLREGs will not be favorable. We made such a design because the current version of COLREGs is not designed for autonomous ships, and thus it is difficult for the current autonomous systems to cover all the rules. E.g., Rule 8 indicates if there is a risk of collision, a ship is required to act even if it has the right-of-way. However, the 'risk of collision' is not clearly defined and can be subjective. In busy areas, the communication between crews and signal lights might determine the actions of two encountering ships. Rules of the interaction between autonomous ships and manned ships can be slightly different, while the corresponding rules are not fully developed yet. Moreover, with shared information between ships, there is a possibility that a better decision beyond the COLREGs might be made. With all these concerns, we make the weight of COLREGs-compliance adjustable. Such that we can adjust the extent of the compliance according to different cases.

An extensive set of tuning parameters is involved in this method, such as the course change $\Delta\alpha$ threshold and speed change Δv threshold. The selection of these parameters depends on several properties such as the type and size of OS. The cost calculation of all aspects is normalized, so we can adjust weight factors to exhibit a range of different priorities and behaviors. For different cases, it can also be time-consuming as the tuning parameters are not completely independent.

This method can be further refined by using more detailed representations of the uncertainty of the models, e.g., a more realistic ship model instead of the simplified 3-DOF model, a better motion-prediction model for obstacles. The Kalman filter is used in this paper for motion prediction, which is a predictor combining the historical measurements and the kinematic model and its noise. We can choose a better predictor, considering more factors, such as more specific motion models of obstacles, a shared destination, and a sailing intention, to improve the prediction performance. Another point worth noting is that all the calculation times mentioned in this paper are based on a PC with the computational power of an Intel i5 core. The details of the computation environment is given in Figure 15(d). The actual calculation time depends on the specific computational environment.

For the real case study, it is indicated by the upper quartile in Figure 11(b) that 75% computation time is less than 0.2 s. The maximum is around 0.41 s, and the minimum is 0.02 s. All the times are less than 1 second, including the outliers. The result proves the effectiveness of the proposed IHA*. Furthermore, it is observed that the average time is less than the previous simulations. It means, in this case, the upcoming encountering scenarios are not as complex as we designed in the simulations. That is because the actual density of obstacles is smaller due to the larger map scale. The own ship does not encounter all the targets simultaneously. Instead, the encounters happen sequentially.

5. CONCLUSIONS

This paper proposes a hybrid method based on Hybrid A* with discrete control behaviors for maritime real-time collision avoidance in multi-target scenarios. A collision pre-check method, i.e., the collision velocity check (CVC), is proposed and applied to speed up the solution search process. Simulations show that the method is effective and can safely manage complex scenarios with multiple dynamic obstacles and uncertainty associated with the sensors and predictions. The computational efficiency is improved massively by using CVC, as shown by the comparison study, while the selected trajectory is not changing much when a proper threshold is selected. For the determination of the CVC threshold in a generalized scenario, further studies are required. Furthermore, the method is also successfully applied to a real-time simulation of a multi-target case using logged radar and AIS data from a real sea trial.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No 812.788 (MSCA-ETN SAS). This publication reflects only the authors' view, exempting the European Union from any liability. Project website: http://etn-sas.eu/.

APPENDIX

This section briefly lists the relevant rules for our research from COLREGs, COLREG (2003):

- **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 reciprocal 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.

References

Beser, F., Yildirim, T., 2018. Colregs based path planning and bearing only obstacle avoidance for autonomous unmanned surface vehicles. Procedia computer science 131, 633–640.

Bitar, G., Martinsen, A.B., Lekkas, A.M., Breivik, M., 2020. Two-stage optimized trajectory planning for asvs under polygonal obstacle constraints: Theory and experiments. IEEE Access 8, 199953–199969.

Brcko, T., Androjna, A., Srše, J., Boć, R., 2021. Vessel multi-parametric collision avoidance decision model: Fuzzy approach. Journal of Marine Science and Engineering 9, 49.

Campbell, S., Naeem, W., 2012. A rule-based heuristic method for colregs-compliant collision avoidance for an unmanned surface vehicle. IFAC proceedings volumes 45, 386–391.

Chen, L., Negenborn, R.R., Lodewijks, G., 2016. Path planning for autonomous inland vessels using a* bg, in: International Conference on Computational Logistics, Springer. pp. 65–79.

Cheng, Y., Zhang, W., 2018. Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. Neurocomputing 272, 63–73.

COLREG, I., 2003. Convention on the international regulations for preventing collisions at sea, 1972. IMO: London, UK .

Daniel, K., Nash, A., Koenig, S., Felner, A., 2010. Theta*: Any-angle path planning on grids. Journal of Artificial Intelligence Research 39, 533–579.

Dijkstra, E.W., et al., 1959. A note on two problems in connexion with graphs. Numerische mathematik 1, 269–271.

Dolgov, D., Thrun, S., Montemerlo, M., Diebel, J., 2008. Practical search techniques in path planning for autonomous driving. Ann Arbor 1001, 18–80.

Dorigo, M., Birattari, M., Stutzle, T., 2006. Ant colony optimization. IEEE computational intelligence magazine 1, 28–39.

Fossen, T.I., 2011. Handbook of marine craft hydrodynamics and motion control. John Wiley & Sons.

Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics 4, 100–107.

Hu, Y., Meng, X., Zhang, Q., Park, G.K., 2020. A real-time collision avoidance system for autonomous surface vessel using fuzzy logic. IEEE Access 8, 108835–108846.

Huang, Y., Chen, L., Van Gelder, P., 2019. Generalized velocity obstacle algorithm for preventing ship collisions at sea. Ocean Engineering 173, 142–156.

Johansen, T.A., Perez, T., Cristofaro, A., 2016. Ship collision avoidance and colregs compliance using simulation-based control behavior selection with predictive hazard assessment. IEEE transactions on intelligent transportation systems 17, 3407–3422.

Khatib, O., 1986. Real-time obstacle avoidance for manipulators and mobile robots, in: Autonomous robot vehicles. Springer, pp. 396–404.

Kim, H., Kim, D., Shin, J.U., Kim, H., Myung, H., 2014. Angular rate-constrained path planning algorithm for unmanned surface vehicles. Ocean Engineering 84, 37–44.

Kurzer, K., 2016. Path planning in unstructured environments: A real-time hybrid a* implementation for fast and deterministic path generation for the kth research concept vehicle.

Lazarowska, A., 2017. Multi-criteria aco-based algorithm for ship's trajectory planning. TransNav: International Journal on Marine Navigation and Safety of Sea Transportation 11.

Lin, H.S., Xiao, J., Michalewicz, Z., 1994. Evolutionary algorithm for path planning in mobile robot environment, in: Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, IEEE. pp. 211–216.

Lyu, H., Yin, Y., 2019. Colregs-constrained real-time path planning for autonomous ships using modified artificial potential fields. The Journal of navigation 72, 588–608.

Meyer, E., Heiberg, A., Rasheed, A., San, O., 2020. Colreg-compliant collision avoidance for unmanned surface vehicle using deep reinforcement learning. IEEE Access 8, 165344–165364.

Miao, T., El Amam, E., Slaets, P., Pissoort, D., 2020. Multi-target tracking and detection, fusing radar and ais signals using poisson multi-bernoulli mixture tracking, in support of autonomous sailing, in: Proceedings of the International Naval Engineering Conference & Exhibition (INEC), Institute of Marine Engineer, Science & Technology (IMarEST); Zenodo. pp. 1–13.

Särkkä, S., 2013. Bayesian filtering and smoothing. 3, Cambridge university press.

Singh, Y., Sharma, S., Sutton, R., Hatton, D., Khan, A., 2018. A constrained a* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents. Ocean Engineering 169, 187–201.

Soltan, R.A., Ashrafiuon, H., Muske, K.R., 2009. Trajectory real-time obstacle avoidance for underactuated unmanned surface vessels, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. 1059–1067.

Szłapczyński, R., Ghaemi, H., 2019. Framework of an evolutionary multi-objective optimisation method for planning a safe trajectory for a marine autonomous surface ship. Polish Maritime Research 26, 69–79.

Vagale, A., Oucheikh, R., Bye, R.T., Osen, O.L., Fossen, T.I., 2021. Path planning and collision avoidance for autonomous surface vehicles i: a review. Journal of Marine Science and Technology , 1–15.

Zhuang, J.y., Zhang, L., Zhao, S.q., Cao, J., Wang, B., Sun, H.b., 2016. Radar-based collision avoidance for unmanned surface vehicles. China Ocean Engineering 30, 867–883.