Real-time Collision Avoidance Algorithm for Inland Waterway Sailing Scenarios - Experiment Validation

Tianlei Miao [ab*], Ehab El Amam [b], Bas de Kruif [c], Peter Slaets[d], Davy Pissoort [a]

[a]*Department of Electrical Engineering (ESAT), Waves: Core Research and Engineering (WaveCore), KU Leuven Bruges Campus, Belgium*
[b]*Autonomy department, RH Marine Netherlands B.V., Schiedam, The Netherlands*
[c]*Maritime Research Institute Netherlands (MARIN), The Netherlands*
[c]*Group T, Robotics, Automation and Mechatronics (RAM), KU Leuven, Belgium*
*Corresponding author. Email: tianlei.miao@rhmarine.com

Synopsis

A Collision Avoidance (CA) system with collision avoidance algorithms is critical to Autonomous Surface Vehicles (ASV). This paper explores a hybrid collision avoidance method based on an improved hybrid A* algorithm for inland waterway scenarios. This method uses discrete control actions, such as heading and speed setpoints, as input and generates a modified local path and control setpoints. A pre-processing method, namely a collision velocity check, is used to speed up the computation progress. Multiple aspects, including safety, path length, sailing time, rule compliance, and smoothness, are considered in finding the optimal collision-free path. We implemented the method under the Robot Operation System (ROS) architecture and conducted real-ship experiments for validation. The results show the effectiveness of this method.

Keywords: Autonomous surface vehicles; Collision avoidance; Path planning; Hybrid A*

1 INTRODUCTION

There has been a growing interest in autonomous sailing in recent years. With the rapid development of autonomy today, it is a trend that Autonomous Surface Vessels (ASV) will reform the current sailing and shipping pattern due to their efficiency and performance in navigation. To achieve higher autonomy levels, many studies and projects have been carried out in the last decade. E.g., the Norwegian University of Technology and Science (NTNU) conducted a project, AUTOSEA (Brekke et al., 2019), aiming at sensor fusion and collision avoidance for ASVs. Another research project, Autonomous Barges for Smart Inland Shipping (AUTOBarge), was established in 2021 by multiple universities, research organizations and companies in Europe, e.g., KU Leuven, TU Delft, Kongsberg, RH marine and Maritime Research Institute Netherlands (MARIN), to explore the idea of smart inland shipping (Autobarge).

One critical part of ASVs is a collision avoidance (CA) system with an effective and efficient collision avoidance algorithm or path planning algorithm. Though the different terminologies are used in different literature, the goal is the same: to find a path that is safe, optimal with regard to optimization goals, and feasible for the own ship (OS) during the whole sailing time. Typically there are two flavours of path planning, *global path planning* and *local path planning* (Vagale et al., 2021). The former focuses on the static environment and known obstacles and optimizes factors like fuel cost and sailing time. This paper focuses on the latter one, the local path planning, which emphasizes avoiding the dynamic obstacles in direct vicinity to generate a feasible and safe path. However, designing a real-time algorithm that can simultaneously cover multiple aspects is challenging. First, the environmental dynamics and moving obstacles need to be tackled. Planning a route in restricted areas also requires more accurate knowledge on the own ship's dynamics. Besides, the compliance with the rules, such as *International Rules for Preventing Collisions at Sea* (COLREGs) for maritime scenarios and *Police regulations for the navigation of the Rhine* (RPR) for the Rhine river, further increases the difficulties of algorithm design. In addition, the collision avoidance algorithm needs to generate solutions online with sufficient computational speed to ensure safety. Typically a trade-off needs to be found among all these concerns.

Authors' Biographies

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 Master's Degree 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-object tracking, and collision avoidance in autonomous sailing.

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

Bas de Kruif is employed at the Maritime Research Institute Netherlands as a researcher in the field of control. His research activities focus on controlling ship motions for autonomous surface and underwater vessels.

Prof. dr. 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.

Prof. dr. 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.

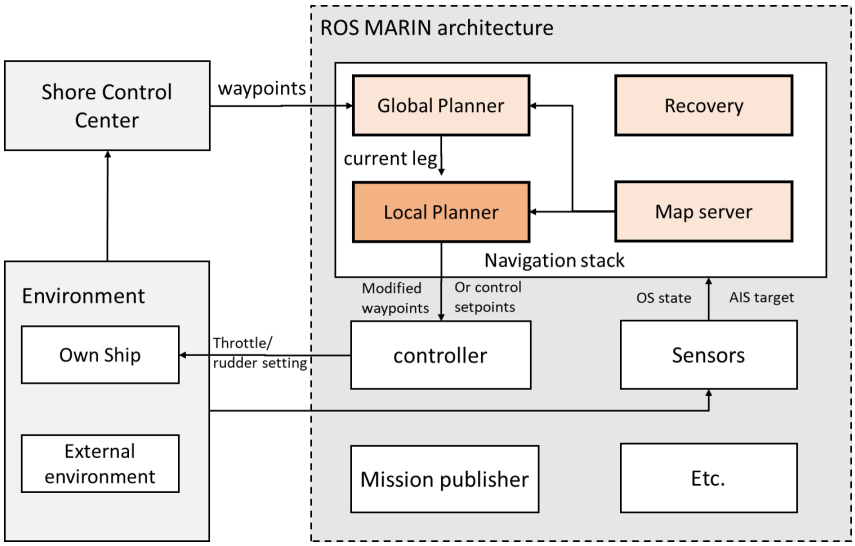

Figure 1: The architecture for autonomous sailing of the vessel

The graph search method is a classical algorithm for path planning and collision avoidance. The initial idea of the graph search method aims to find the shortest path between points a and b on a geometric map. Later it has also been applied to find the solution between two states in a system, e.g., find a sequence of moves to win in a chess game. One well-known method was proposed by Dijkstra (1959) and named after him. During the search progress, Dijkstra's algorithm favors lower-cost paths. However, it does not utilize any heuristic information, even if a general direction has already been obtained. A-star (A*) (Hart et al., 1968) was proposed based on Dijkstra's method. It uses the actual distance from the start and the estimated distance to the goal. Due to the success of applying A* in various fields, many variants have been further proposed. Weighted A* and dynamic weighted A* were proposed to adjust the impact of actual cost and heuristic (Pohl, 1970; Pearl, 1984). Dynamic A* (D*), Lifelong Planning A* (LPA*), and a light version, D* lite, were proposed for incomplete map information and changing maps costs (Stentz, 1997; Koenig et al., 2004; Koenig and Likhachev). In addition, Dolgov et al. (2008) proposed hybrid A* (HA*) considering the dynamics of the vehicles. It associates a continuous state with each cell based on the motion constraints, which builds a connection between the search progress of the A* and the vehicle dynamics. Another characteristic is to calculate the heuristic in a hybrid way: geometry-dependent and dynamic-dependent. Typically, an additional smoothing step is required to modify the path further, considering more details of dynamics or kinematics.

There are also many applications of the variants of A* in the maritime scenario. One variant, a heuristic Rule-based 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 key points (such as corners) during path searching and a line of sight pointing directly back to an ancestor during retrace. It allows the searching towards any angle in a grid map, which weakens the restrictions of grid size. Kim et al. (2014) proposed an approach based on the Theta* for real-time path planning, considering both angular rate (yaw rate) and heading angle of unmanned surface vehicles (USVs). Singh et al. (2018) extended the implementation of this Theta* in an environment cluttered with static and moving obstacles and different current intensities. These two methods were applied in more realistic environments. However, they still rely on the grid map, so the selection of grid size will influence its computational efficiency and collision avoidance decision. Besides, only simple encounter scenarios are tested, and the computational speed is not mentioned or ignored. In addition, rare literature tested their algorithms on a real ship. In order to produce a comprehensive collision avoidance solution in real-time concerning multiple optimization aspects, this paper has applied an improved hybrid A* algorithm that generates a modified local path and associated control setpoints for inland waterway scenarios. With the discrete control input, the ship dynamics or kinematics have already been considered during the search progress, such that no extra smoothing step is needed. A real-ship experiment was conducted as the validation.

2 AUTONOMOUS SYSTEM OVERVIEW

The autonomous sailing system designed in this research follows the architecture of Maritime Research Institute Netherlands (MARIN) (see in Fig1). It has basic modules for autonomous vessels, like sensing, navigation and

control. All the modules are built as packages based on Robot Operation System (ROS). The emphasis of this paper, collision avoidance, is implemented as a plugin in the local planner of the navigation module. A Shore Control Center (SCC) that MARIN uses for their maritime simulators has been adopted to control ASVs remotely. It can send and receive data and commands and has a graphic user interface (see in Fig 2) for displaying states of the OS and target ships (TS), editing global waypoints, publishing missions (e.g., start a path follower or heading keeper), and switching control modes.

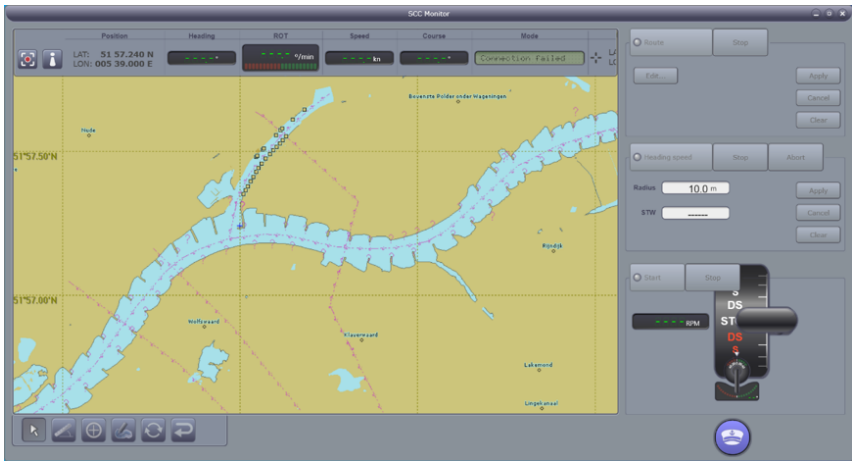

Figure 2: Shore Control Center

ROS is an open-source framework designed for general robot software development. It provides various packages for essential functions, like the sensor and controller modules. Based on this, MARIN applied it in a maritime context. The function of collision avoidance is implemented in the node "(move_base)" of the navigation stack. Its standard structure consists of five parts: global planner, local planner, global costmap, local costmap, and recovery behavior. The whole "move_base" takes the global waypoints from SCC as the input. The global planner is re-designed as a straight-through module that outputs the following few global waypoints. Besides, it will check if there are any input changes. Next, the local planner carries out the planning and decision stage of collision avoidance. Finally, the feasible path is calculated using the hybrid algorithm with the given global waypoints and the data from sensors (e.g., OS, TSs, shore). A detailed design of the local planner is shown in Fig 3, and pseudo-code is given in Algorithm 1.

Generally, the environmental information, such as wind, current, land, and water depth, is obtained from the chart and sensors. Obstacle information is obtained from AIS, Radar, Lidar, and camera sensors. Except for the AIS data, which already includes the target point date (sent in NMEA sentences), object detection and classification are required for other sensors. After that, the future motion of these objects will be further used for the cost estimation in the path planning step. Based upon environmental information, obstacle information, and together with the given discrete control action, a continuously varying trajectory segment can be calculated. We select the trajectory combination with the minimum total cost according to the cost models concerning all the aspects. Both modified waypoints and control setpoints are outputted to the controller.

3 COLLISION AVOIDANCE ALGORITHM

The collision avoidance algorithm used in this research is based on a hybrid A* method. An intermediate collision velocity check is included to speed up the computations. We further customized the implementation, and modified cost functions in this research for inland waterway scenarios. A set of discrete control actions is used as the input. According to the kinematic model used in this paper, heading and speed options are input.

*3.1 Improved hybrid A**

The improved hybrid A* (IHA*) was proposed by our previous work (Miao et al., 2022). Based on HA*, node map is used as the search space (see in 4), which is dynamically generated during the solution searching, instead of using a conventional grid map. Each movement is independent of the grid size and shape. This algorithm uses discrete control actions as input. As an improved version of HA*, it can also associate the continuous state of the own vessel with nodes, and associated trajectories can be generated. A pre-processing method, collision velocity check, filters out the set of unsafe control inputs to increase the computational speed. Cost functions and heuristics

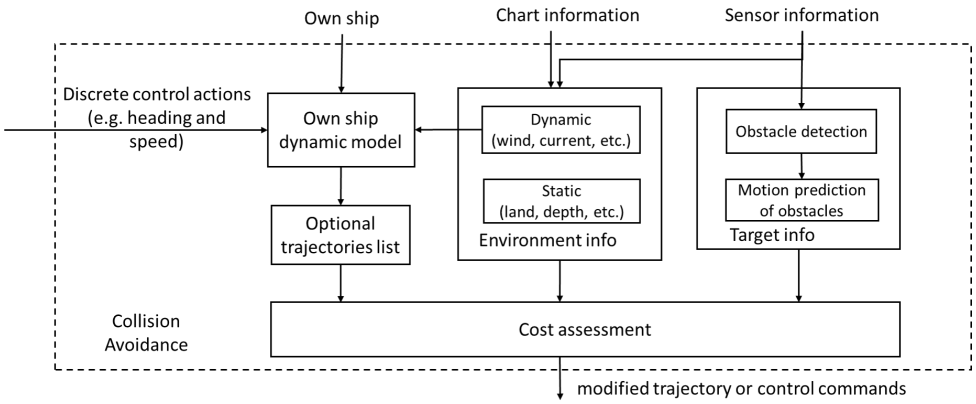

Figure 3: The structure of the designed local planner

are re-designed for the inland waterway context in terms of five aspects: safety, path length, sailing time, rule compliance, and path smoothness. Rule compliance is considered a part of the cost function. For the real-time implementation, online replanning is performed and executed during experiments.

A standard kinematic model of ASVs is used in this research shown as follows:

$$\begin{cases} \dot{x} = u \cdot cos\varphi \\ \dot{y} = u \cdot sin\varphi \\ \dot{u} = a_t \\ \dot{\varphi} = a_n / u \end{cases} \tag{1}$$

where (x, y) are the position of the ship, u is the speed, and φ is the heading angle. a_t and a_n are tangential and normal accelerations, respectively. This model comprises various kinematic models, such as "Dubins car" model (Dubins, 1957).

Fig 4 illustrates the search process. The timeline at the bottom represents the entire planning time horizon from T_0 to T_{end}. It is broken down into discrete time slots with a fixed time interval Δt. With the current state of the OS and given control actions (e.g., heading offset -15°, 0°, and 15°, and full speed), the associated trajectories (blue solid lines) for each time slot and state of the nodes (red solid dots) can be calculated using the motion model. The entire search space is like a 'tree'.

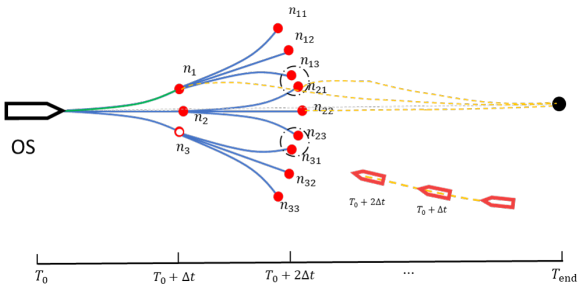

Figure 4: Search process of the IHA*

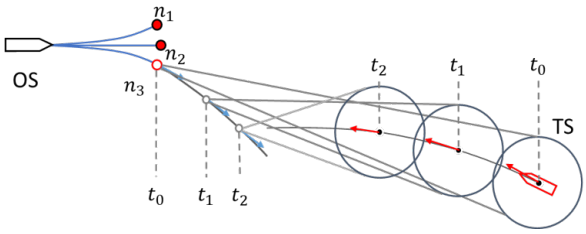

Figure 5: Collision Velocity Check

Instead of expanding in all directions equally, the hybrid A* always favors the path with a minimum total cost, which consists of the actual cost and the heuristic, defined as:

$$f(n) = g(n) + h(n) \tag{2}$$

where $g(n)$ is the actual accumulated cost (historical cost) of the node n represented by the green solid line and $h(n)$ is the heuristic cost of the node n represented by the orange dotted line.

To speed up the search process of the hybrid A*, a method, collision velocity check, is proposed and used in the pre-processing step. Its essential idea is to remove the potential dangerous velocity area from the search space.

With the prediction of the trajectory of both OS and obstacles, the minimum distance is checked at discrete future time points.

In addition to pre-processing, a post-processing step, pruning, is also used for further acceleration. It aims to cut down the branches that contribute only very little to the solution, but vastly increase the size of the search space 'tree'. Usually, such branches have a similar state to existing ones, but with relatively higher costs. As shown in Fig 4, the nodes in the dotted circle will be compared. 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 hybrid A* (n_{start}, n_{des})

$List_{open} \leftarrow n_{start}$
$List_{close} \leftarrow \emptyset$
while $List_{open} != \emptyset$ **do**
  $List_{open} = Sort(List_{open}, f, reverse)$
  $n_p = PopTop(List_{open})$
  $List_{cloce} \leftarrow List_{close} \cup n_p$
  if $GoalReach = true$ **then**
    break
  else
    for $heading = h_1, h_2, \ldots, h_i, \ldots, h_I$ **do**
      for $speed = s_1, s_2, \ldots, s_j, \ldots, s_J$ **do**
        compute next position $n_T^{i,j}$
        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

3.2 Assumptions

The complexity of ASV collision avoidance is massive and a number of simplifications have been made to reduce the intricacies of the problem. Here, the following assumptions have been made:

- The sailing area is considered to be in an inland waterway environment. Temporal and spatial variability in the chosen study area in terms of environmental effects and moving vessels is considered quasi-static for the period of the voyage.
- The OS is modeled as a rigid polygon under the assumption that an effective, robust controller quickly establishes the commanded velocity.
- AIS measurements are used to determine the obstacle position over time with a Kalman filter.
- The moving obstacles are modeled as a polygon that has a fixed shape with different scales.

3.3 Safety domain & rule-compliance domain

A proper safety domain needs to be used for the collision risk evaluation. Since we focus on the inland waterway scenario, a simple circle domain is unsuitable for this research. It is a typical situation when two vessels pass each other from the two sides of the canal. There should be no risk or low risk. Thus, an ellipse

is selected to represent the safety domain of our ship as seen in Fig 6. The rule-compliance domain defines the current encountering situation when obstacles approach, which is also represented as an ellipse, while the size can be different. The encountering cases are also defined in this domain.

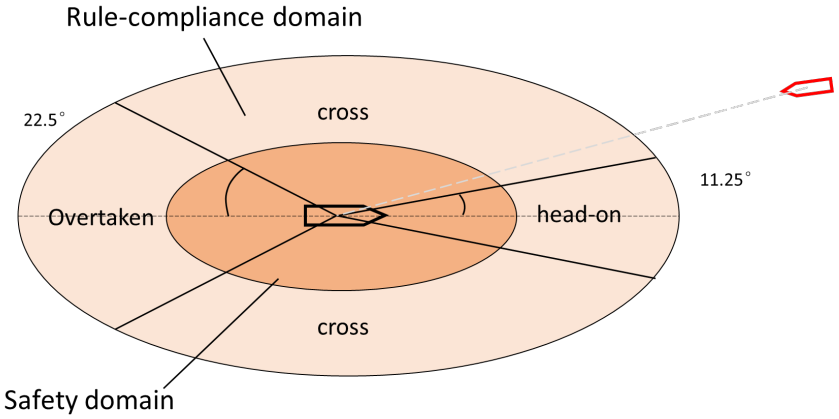

Figure 6: Safety domain and rule-compliance domain

3.4 Cost functions

The total cost is a summation of the normalized cost functions in multiple aspects. The basic components follow our previous work described in Miao et al. (2022). Weight factors are set for all aspects. Relatively higher weight is given to the safety aspect.

3.4.1 historical cost

The historical cost of a node is the minimal known accumulated cost from the starting node. It is determined in five aspects: path length, sailing time, safety, rule compliance, and smoothness.

The historical path length is the total length of all the trajectories selected. Its cost is expressed as:

$$C_{leng}^{hist} = \frac{\sum(L_i)}{L_{nomi}} \tag{3}$$

where L_i is the length of trajectory segment i and L_{nomi} is the nominal length.

The historical time is the accumulated sailing time starting from the initial T_0. Its cost is expressed as

$$C_{time}^{hist} = \frac{(T - T_0)}{T_{nomi}} \tag{4}$$

where T_{nomi} is the nominal sailing time.

The safety is expressed by the accumulated collision and grounding risks with certain risk assessment models. Its cost is expressed as the sum of the accumulated collision and grounding risk of all trajectories

$$C_{safe}^{hist} = \sum C_{coll,i}^{hist} + \sum C_{grd,i}^{hist} \tag{5}$$

The collision cost of each trajectory is designed as a piece-wise function. When the distance between the OS and obstacle is far, the cost is 0; when it is too close, it is infinite; when it is in between, it is modeled as a function of time and distance as

$$C_{coll,i}^{hist} = (\frac{1}{|T - T_0|})^p \cdot (\frac{d_{safe}}{d_i})^q \cdot f_{RP} \tag{6}$$

where p and q are the time and distance index, respectively, d_{safe} is the minimum safety distance, and d_i is the distance between the OS and obstacle. The relative posture factor, f_{RP}, is used to estimate the influence of different encountering postures. The grounding cost consists of the cost from the pre-calculated global costmap and the time effect

$$C_{grd,i}^{hist} = (\frac{1}{|T - T_0|})^p \cdot C_{cell}^{m,n} \tag{7}$$

where (m,n) associates the column and row position on a costmap.

The rule-compliance is designed as a penalty function when the rules are violated, expressed as

$$C_{rule,i}^{hist} = \begin{cases} 1, & \text{if rule violated} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Thus, the historical rule-compliance cost is a summation of each trajectory i

$$C_{rule}^{hist} = \sum C_{rule,i}^{hist} \tag{9}$$

The smoothness aspect is designed to avoid sharp turnings and accelerations. So an extra cost is introduced when the heading and speed change too fast.

$$C_{smth}^{hist} = \frac{max(\dot{\varphi} - \lambda_h, 0)}{\lambda_h} + \frac{\max(\dot{u} - \lambda_s, 0)}{\lambda_s} \tag{10}$$

where λ_h and λ_s are thresholds of heading and speed change.

3.4.2 Heuristic cost

The heuristic cost is a cost prediction for the rest route to the goal. Good construction of heuristic cost can guide the expansion direction, shortening the search process. We estimate the heuristic cost considering two aspects: path length and sailing time.

The heuristic path length cost is defined as:

$$C_{leng}^{heur} = \frac{L_{heur}}{L_{nomi}} \tag{11}$$

where the path length is estimated by straight distance to the goal, d_{goal}, and the angle difference between the current course and direction to goal α:

$$L_{heur} = d_{goal} \cdot (1 + |sin(\frac{\alpha}{2})|) \tag{12}$$

The heuristic sailing time cost is defined as:

$$C_{time}^{heur} = \frac{T_{heur}}{T_{nomi}} \tag{13}$$

$$T_{heur} = \frac{L_{heur}}{\frac{T-T_0}{T_{end}-T_0} \cdot |u_c| + \frac{T_{end}-T}{T_{end}-T_0} \cdot |u_{nomi}|} \tag{14}$$

where the heuristic sailing time is estimated based on the ship's current speed u_c, nominal speed u_{nomi} and left path length L_{heur}.

3.5 Online re-planning

As a graphic planning method, replanning is used for the real-time execution of collision avoidance. It means the algorithm will generate a new path based on the current updated situation with a specific frequency to cope with the changing environment and OS state. Due to limited computational power, the replanning frequency can not be infinite. A proper frequency needs to be selected according to the specific requirements, such as the vessel's maneuverability, and reaction speed according to the current situations. Besides, to avoid the motion oscillation due to the replanning, the new planned path is only accepted when it is much better than the old one or the global waypoints change. The replanning is implemented by modifying the " (nav_core)" package of the standard navigation stack in ROS. Algorithm 2 presents the entire process of the replanning progress.

Algorithm 2 Re-planning of IHA*

$P_{current} \leftarrow WP_{global}$
while $Clock_{system}$ mod 1 **do**
  $(P_{new}, cost_{new}) \leftarrow IHA*(n_{start}, n_{des})$
  if $cost_{new} - cost_{current} < threshold$ or environment changes **then**
    $P_{current} \leftarrow P_{new}$
  end if
  run $pathfollower(P_{current})$
end while

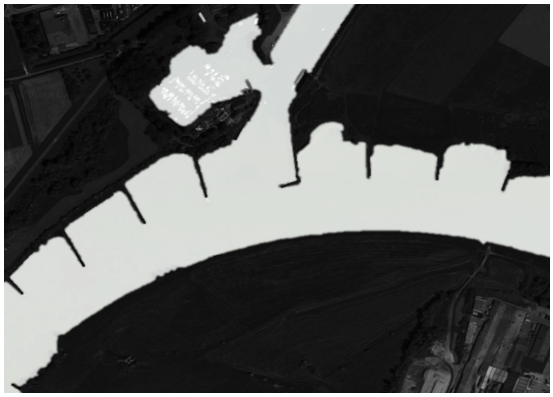

Figure 7: Port of Wageningen

Figure 8: Cost map with an inflation layer

4 Experiments & Results

The location for the experiments is selected at the Port of Wageningen (51.9565°, 5.6495°) on the Rhein river (see in Fig 7). The land area (dark color area) is not accessible, which is regarded as a hard constraint during sailing. In contrast, the water area (light color area) is the free-access sailing domain of this research. An inflation layer with 20 meters thickness is added to represent the influence of the water depth. An incremental cost function of this inflation layer is used to connect the land and water area smoothly (see in Fig 8). Virtual moving obstacles are used for the experiments in the dynamic environment. AIS data of these moving obstacles were received and used. The dynamic state of the OS is updated with the GPS.

The global waypoints are manually given in the shore control center. The current leg and next few waypoints will be sent to the local planner. The local planner will generate a modified intermediate path and associated control setpoints. We set the nominal speed of the OS to 10 knots and the planning time horizon to 5 minutes with a 10-second time interval, according to the scope of this research. The initial discrete control options of the CA algorithm are given as follows during experiments and the associated a_t and a_n are used:

- heading change with respect to previous steps: -45, -30, -15, 0, 15, 30, 45 degrees
- speed: 20, 50, 100 percentages of full speed

4.1 Experiment setting

A 7m long unmanned RHIB 'Ms. Auris' supported by MARIN was used as our OS. It is equipped with all essential relevant functions to actuate missions, such as GPS antennas and an onboard computer. During the experiment, four cases were tested: static map, stationary obstacle, head-on, and overtaken scenarios. For each case, several global waypoints were pre-set manually and the OS starts from the current position to reach the waypoints in sequence. Collisions or groundings would occur on purpose with these initial waypoints. The modified waypoints from the CA algorithm were received in real-time. Then, a line-of-sight (LOS) controller is used to follow the modified waypoints. The re-planning ensures real-time safety with 1 Hz for all the cases.

The algorithm is first tested in a static map without obstacles. In this case, six waypoints (wp1-wp6) are pre-set, and a potential grounding is designed between wp1 and wp2. The result is illustrated in Fig 11. In the second case (Fig 12), a static obstacle, which is a stationary vessel, is injected into a 3-waypoints route to create a potential collision. A moving obstacle (T-1) sailing from west to east, which has varying speed and heading, is injected to test the head-on case (Fig 13). For the overtaken case (Fig 14), a moving vessel sailing along the port side of the OS is set to create a potential collision risk.

4.2 Experiment results

In all cases, a set of modified intermediate waypoints is successfully generated between every two global waypoints. Own ship is displayed in orange, the obstacle is displayed in dark red, the initial global waypoints are in light blue, and the modified waypoints are in red color.

Figure 9: MARIN's RHIB, the 'Ms. Auris'

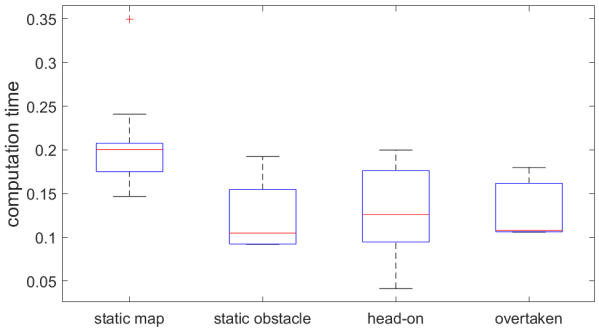

Figure 10: computation time box-plot for all cases

The modified waypoints for the first static map case avoid the potential grounding between global waypoints 1 and 2. To make the path shorter, these new waypoints go through the inflation area a bit. The consideration of multiple aspects causes it. For the other legs (segment between waypoints), the path is modified with intermediate waypoints based on the updated OS state. In the fixed obstacle case, the generated waypoints guide the OS to avoid collision with T-1 from the starboard side. The speed is slowed when OS is approaching the stationary ship due to a slow speed option selected by the algorithm. A sufficient distance to both T-1 and shore is kept to ensure safety.

For two cases with moving obstacles, the states of OS and the obstacle are displayed to show the encountering details. The color turns from dark to light over time. In the head-on case, the modified waypoints intend to avoid the port side. Firstly, the waypoint set in purple is associated with the slow-speed obstacle in the beginning. When OS detects the obstacle has increased speed, a new set of waypoints in tomato color is accepted. So the waypoints get updated. In the overtaken case, modified waypoints show the avoidance of the coming obstacle and the static map.

We logged the computation time of each case during the experiments and box-plotted the average time in Fig 10. It can be seen that all the computation time are less than 1 second, which means a solution can be generated within the desired time. There was no extra latency during experiments.

4.3 Discussions and future research

Overall, the improved hybrid A* with collision velocity check is successfully implemented under the ROS architecture of MARIN. By specifying the cost functions for the inland waterway context, the collision avoidance method can generate suitable solutions without collisions and groundings in real-time. The results of the experiments show that the RHIB avoided the groundings and collisions in all tested cases.

For static environments (static map and obstacles) and moving obstacles, optimal solutions are generated during each mission based on the designed cost functions. The modified route is the result of balancing multiple aspects: safety, path length, sailing time, rule compliance, and smoothness. This balance can be seen in some places. E.g., the modified waypoints go through the inflation layer a bit to shorten the path distance and sailing time. Moreover, in the head-on case, the typical starboard rule is not followed - the avoidance action is taken from the port side due to the limited space left on the starboard. Such design aims to achieve a more comprehensive solution. In the static obstacle case, the algorithm gives a solution avoiding the obstacle from the starboard, instead of the port side. One reason is that although a stationary ship was injected as the static obstacle, it was drifting away from the shore due to the current and wind. The obstacle motion predictor gave a trajectory leaving the shore, which resulted in the avoidance from starboard. Thus, a modification is needed for the obstacle motion prediction according to the external environment. Besides, if a port-side avoidance is required according to the rules, we can modify the solution by correcting the rule-compliance cost. Besides, a higher weight factor of rule compliance can be set, such that violating rules can have more penalty costs. Such that, rule-compliant solutions can be favored.

By re-planning, new sets of waypoints are constantly generated to ensure real-time collision avoidance. However, new waypoints will be accepted only when the change of environment is detected, e.g., the new next global waypoint is received, and obstacle state changes are detected. For cases like the static map and the static obstacle, only one set of modified waypoints is accepted for each leg. Such a design avoids the common drawback of re-planning, which causes unnecessary fluctuation of the vessel heading.

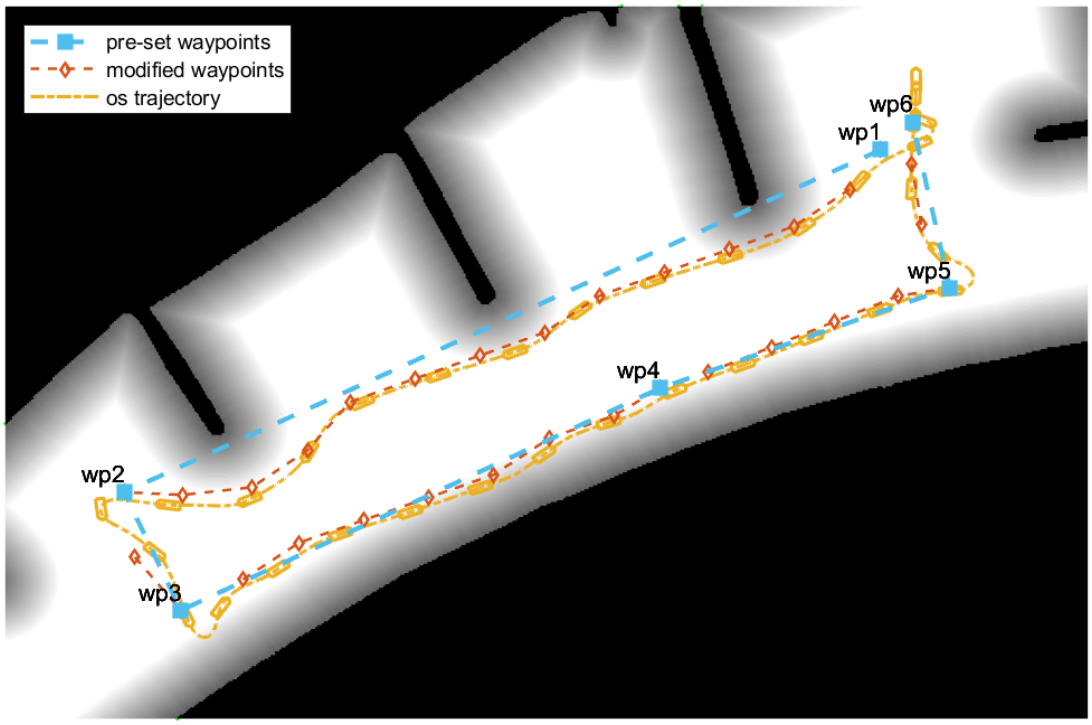

Figure 11: Experiment 1: static map

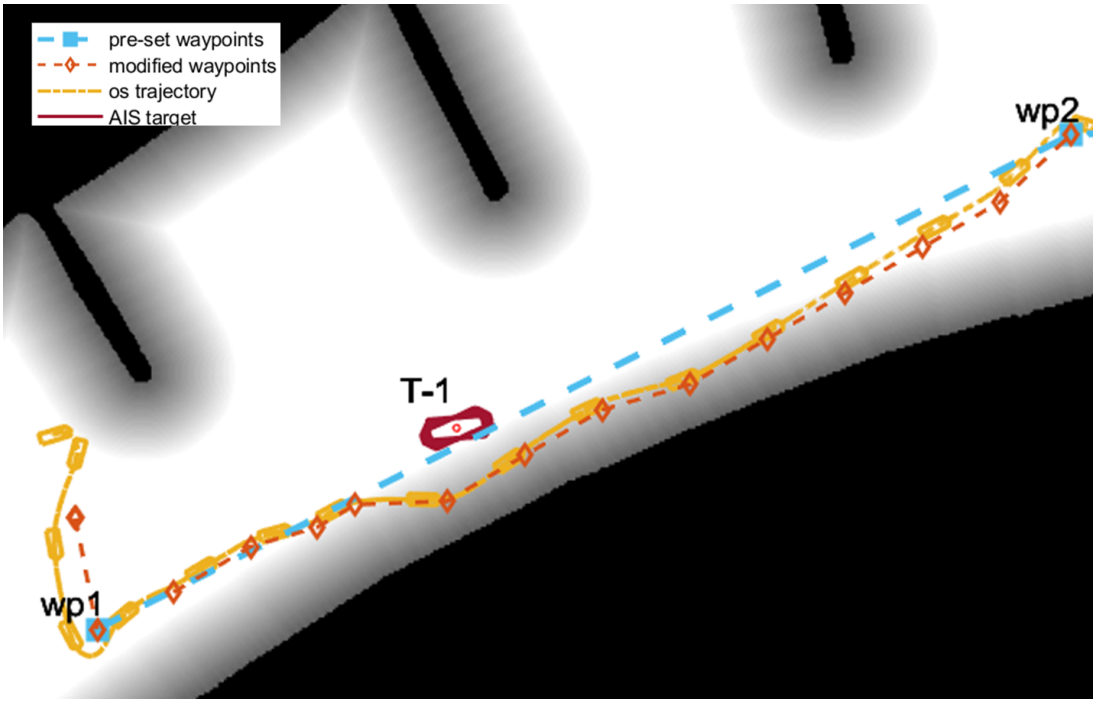

Figure 12: Experiment 2: static obstacle

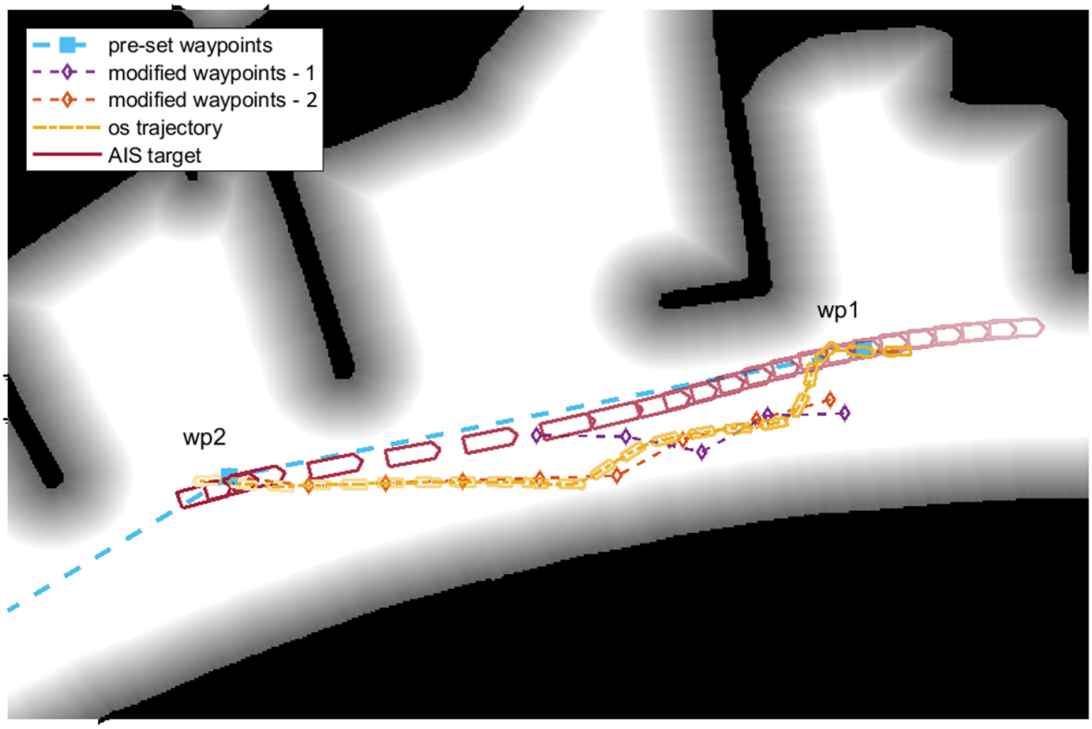

Figure 13: Experiment 3: head-on case

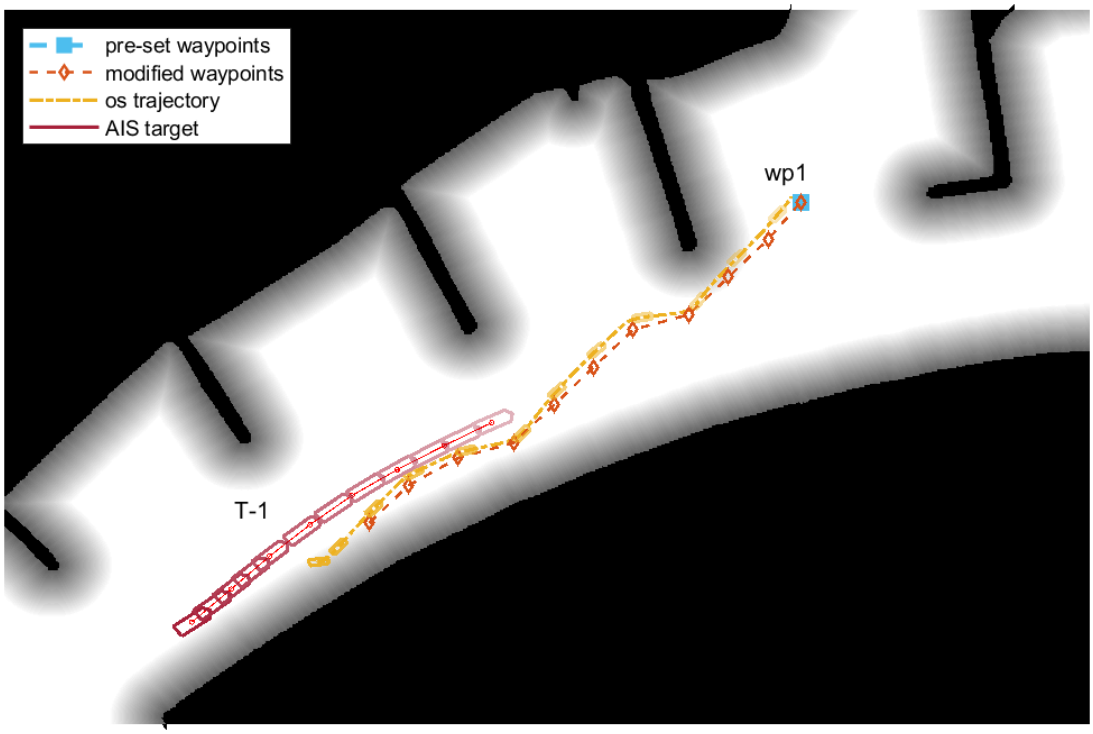

Figure 14: Experiment 4: overtaken case

The computation time is relatively longer in the static map case. That is due to some global waypoints being set in the concave areas (i.e., the water area between two land sticks). To search all the possibilities, the algorithm searched paths also in this blind alley, which costs extra time. That means, for inland waterway scenarios, different shore geometries impact the search progress and consequently impact the computation speed. One way for improvement is to change the heuristic, such that a potential 'trap' can be foreseen.

The motion prediction of OS is simplified using a kinematic ship model in this study. However, it might not fit other cases, such as the ship with low maneuverability. Thus, testing with more ship types and better motion models is one of our future works. For the movement prediction of the obstacles, a Kalman filter was used. Better methods including intention prediction can be selected to give more accurate predictions, which can also be one of our future works. Besides, the communication between own ship and obstacles can provide vital information, e.g., sharing waypoints to each other, for motion prediction.

In addition, more scenarios need to be tested to fully validate this collision avoidance algorithm, such as crossing and overtaking an obstacle. More complex scenarios, such as multi-obstacle encountering, can be set to test this algorithm's performance further. Therefore, further studies are required.

5 Conclusions

In this paper, a real-time collision avoidance method has been implemented based on the improved hybrid A* with collision velocity check for the inland waterway scenarios. We successfully implemented this algorithm under the ROS architecture of MARIN. Real-ship experiments were conducted and various scenarios were tested. From the experiment results, the RHIB successfully avoided the groundings and collisions with the modified intermediate waypoints. Furthermore, multiple aspects, including safety, path length, sailing time, rule compliance, and smoothness, are considered, and comprehensive solutions are generated during all the missions.

Acknowledgement

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/.

References

Autobarge, . Europe training and network on autonomous barges for smart inland shipping. URL: https://etn-autobarge.eu/project/.

Brekke, E.F., Wilthil, E.F., Eriksen, B.H., Kufoalor, D., Helgesen, Ø.K., Hagen, I.B., Breivik, M., Johansen, T.A., 2019. The autosea project: Developing closed-loop target tracking and collision avoidance systems, in: journal of physics: conference series, IOP publishing. p. 012020.

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.

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., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271. doi:10.1007/bf01386390.

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

Dubins, L.E., 1957. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of mathematics 79, 497–516.

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.

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.

Koenig, S., Likhachev, M., . Dˆ* lite .

Koenig, S., Likhachev, M., Furcy, D., 2004. Lifelong planning a. Artificial Intelligence 155, 93–146.

Miao, T., El Amam, E., Slaets, P., Pissoort, D., 2022. An improved real-time collision-avoidance algorithm based on hybrid a* in a multi-object-encountering scenario for autonomous surface vessels. Ocean Engineering 255, 111406.

move_base, R., . Ros move_base description. http://wiki.ros.org/move_base. Accessed: 2022-06-30.

nav_core, R., . Ros nav_core description. http://wiki.ros.org/nav_core. Accessed: 2022-06-30.

Pearl, J., 1984. Heuristics: intelligent search strategies for computer problem solving. Addison-Wesley Longman Publishing Co., Inc.

Pohl, I., 1970. Heuristic search viewed as path finding in a graph. Artificial Intelligence 1, 193–204. doi:10.1016/0004-3702(70)90007-x.

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.

Stentz, A., 1997. Optimal and efficient path planning for partially known environments, in: Intelligent unmanned ground vehicles. Springer, pp. 203–220.

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.