**Proceedings of the ASME 2022**
**International Design Engineering Technical Conferences and**
**Computers and Information in Engineering Conference**
**IDETC/CIE2022**
**August 14-17, 2022, St. Louis, Missouri**

**DETC2022-89760**

# LONG-RANGE RISK-AWARE PATH PLANNING FOR AUTONOMOUS SHIPS IN COMPLEX AND DYNAMIC ENVIRONMENTS

**Chuanhui Hu**
IMPACT Laboratory
Dept. of Aerospace & Mechanical Engineering
University of Southern California
Los Angeles, USA
chuanhui@usc.edu

**Yan Jin***
IMPACT Laboratory
Dept. of Aerospace & Mechanical Engineering
University of Southern California
Los Angeles, USA
yjin@usc.edu
(*corresponding author)

## ABSTRACT

*Path planning and collision avoidance are common problems for researchers in vehicle and robotics engineering design domains. In the case of autonomous ships, the navigation is guided by the Regulations for Preventing Collisions at Sea (COLREGs). However, COLREGs do not provide specific guidance for collision avoidance, especially for multi-ship encounter situations, which is a challenging task even for humans. In short-range path planning and collision avoidance problems, the motion of target ships is often considered as moving at a constant velocity and direction, which cannot be assumed in long-range planning and complex environments. The research challenge here is how to factor in the uncertainty of the motion of the target ships when making long-range path plans. In this paper, we introduce a long-range path planning algorithm for autonomous ships navigating in complex and dynamic environments to reduce the risk of encountering other ships during the future motion. Based on the information of position, speed over ground, and course over ground of other ships, our algorithm can estimate the intentions and future motions of them based on the probabilistic roadmap algorithm and use a risk-aware A* algorithm to find the optimal path that has low accumulated risk of encountering other ships. A case study is carried out on the real Automatic Identification Systems (AIS) datasets, and the result shows that our algorithm can help reduce multi-ship encounters in long-term path planning.*

**Keywords**: Autonomous ship, path planning, probabilistic roadmap, dynamic environment, risk assessment

## 1 INTRODUCTION

Maritime transport is the most economical way of transporting goods globally. Over 80% of international trading is carried out by sea [1]. Among all types of accidents, ship collisions are the main type on the ocean [2]. The International Maritime Organization (IMO) has carried out the International Regulations for Preventing Collisions at Sea (COLREGs) [3] to guide collision avoidance actions. COLREGs have defined three different situations of ship encounters, which are *overtaking*, *head-on*, and *crossing*. The responsibility of the encountered ships is also defined in COLREGs, which helps a ship determine whether it is a give-way ship or a stand-on ship and what action it should take to avoid collisions. However, COLREGs do not provide specific guidance for collision avoidance, and mariners must tell themselves what the situation is and what action they should take. In addition, the three situations defined in COLREGs only cover two-ship encounters. This leaves the multi-ship encounter a complex and risky situation, where a ship can be the give-way ship to one ship and be the stand-on ship to another ship at the same time. The safety of navigation still highly depends on the mariner's experience and judgment. Nevertheless, human errors are the main cause of ship collisions [2].

To overcome the human error in collision avoidance of ships, the autonomous or unmanned ship has become an important direction of research in the shipping industry. In most cases, the state of a ship can be described on a 2-D plane with its position and course (orientation). Many algorithms have been carried out to do path planning and collision avoidance in dynamic environments where other ships are detected. Although these algorithms can make a short-range plan and avoid collision with the existence of other dynamic obstacles, most of them assume that all target ships are moving at a constant direction and velocity, which may be violated in long-range cases. Another common assumption is that the encounter of ships happens in the open area, which is not true in complex environments, such as San Francisco Bay area.

In this paper, we propose a path planning algorithm to reduce the risks of the autonomous ship encountering other ships in long-range motion in complex and dynamic environments. Our approach does not rely on the constant direction and velocity assumption, and the path planned by our algorithm is based on the estimation of the intentions of other ships. Our algorithm can help autonomous ships, as well as human mariners, to avoid complex encountering situations and thus improve the safety of navigation.

The rest of the paper is organized as follows: Section 2 provides a review of related work in path planning in complex and dynamic environments, including their assumptions and limitations. Section 3 describes our proposed method to estimate the intention and future position of target ships and plan the path with a probabilistic roadmap and risk-aware A* algorithm to reduce the risk of encountering target ships. Section 4 shows the case study conducted on the real Automatic Identification Systems (AIS) data. In Section 5, a detailed discussion on the result of the case study is presented. The last section concludes and points out the limitation and future directions.

## 2 RELATED WORK

Path planning has been a hot research topic for years. The collision avoidance of ships and Autonomous Surface Vehicles (ASV) can be modeled as a path planning problem in dynamic environments. Due to the limited control capability of ships and ASVs, constraints on ship dynamics are usually considered in the planning, which makes the problem more complicated.

Many researchers used variants of a rapidly-exploring random tree (RRT) to do the planning. RRT is a sampling-based algorithm that is efficient at finding a feasible path from the starting point to the goal. By adding constraints in the process of tree growth, RRT can be used to produce a smooth path that is friendly to the ship dynamics. Sun, Zhao, and Zhang used bi-directional RRT and Dijkstra's algorithm to plan the path in narrow water areas [4]. Zaccone and Martelli designed a multi-objective cost function of RRT* and generated a collision-free path in dynamic environments [5]. This work is later expanded to find COLREGs-compliance path by introducing the vector representation of collision avoidance rules in the RRT* algorithm [6]. Chiang and Tapia stored the joint state of ships in each RRT node, so a forward simulation can be conducted to find a COLREGs-compliance path [7]. Assuming that the target ships are moving at constant velocities, RRT can be used to find a collision-free path that satisfies the COLREGs regulations and ship kinematic or dynamic constraints.

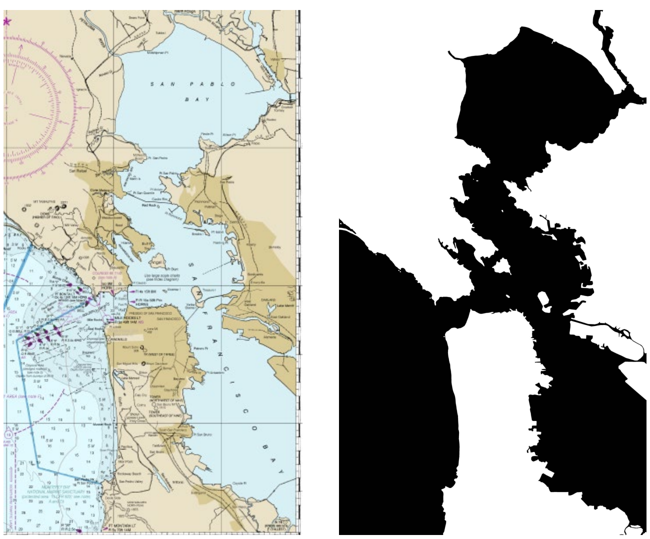
As a popular path planning algorithm in the robotics area, the artificial potential field (APF) algorithm can also be applied to find safe trajectories for ships. Naeem, Henrique, and Hu introduced the COLREGs zones to the target ships so that the trajectory produced by APF can adhere to the COLREGs regulations [8]. Mei and Arshad proposed a smart algorithm that can identify the encounter situation and determine whether the ASV should obey the COLREGs while avoiding other ships [9]. Lyu and Yin modified the repulsion potential field function and their algorithm showed impressive performance in simulation with 5 static obstacles and 11 target ships randomly changing courses in a large open area [10].

Some other researchers modeled the path planning problem as an optimization problem and use heuristic methods to solve it. Evolutionary algorithms can be applied to find an optimized path in complex or dynamic environments. Lazarowska proposed an approach of path planning in dynamic environments based on the Ant Colony algorithm [11]. Tam and Bucknall designed a path-planning algorithm based on an evolutionary algorithm to find a collision-free trajectory when the future motions of all obstacles are known [12]. Wang, Yao, and Duo developed a trajectory optimization algorithm with an improved grey wolf optimizer [13]. The algorithm can find the optimal path in complex situations with multiple statical obstacles and known environmental disturbances such as water current. Kang and his team carried out a ship domain model and used particle swarm optimization to find the collision-free trajectory in two-ship encounters [14]. A path optimization method using a genetic algorithm is introduced by Kim and his team, which considers the environmental loads [15]. These studies have shown that evolutionary algorithms can be used in the path planning of ships in different complex situations, and constraints can be applied to the problem by setting proper fitness/evaluation functions.

Recent progress in deep reinforcement learning has pointed out a new way to solve collision avoidance problems. Liu and Jin studied the knowledge transfer in reinforcement learning-based collision avoidance [16]. Wu and his team proposed the deep reinforcement learning method ANOA and achieved a higher success rate than Recast navigation method in dynamic environments [17]. Both results have shown that when properly designed and trained, reinforcement learning-based methods can achieve excellent performance in collision avoidance.

Besides the methods mentioned above, researchers have also tried many other algorithms to deal with the collision avoidance problem. Singh and his team proposed an A* approach that can deal with both static and dynamic obstacles and environmental conditions such as current and wind [18]. Different variants of the fast marching method are developed and can produce decent solutions to collision avoidance in dynamic environments [19] [20] [21]. Williams and Jin designed a risk assessment method and provided a flexible and safe path in situations where the future motion of target ships is unknown [22]. He et al. modeled a fuzzy PID controller to meet the COLREGs during collision avoidance [23]. Song and his team applied fuzzy rules with the eccentric expansion of obstacles to produce COLREGs-compliant plans [24]. Campbell and Naeem designed a rule-based heuristic A* algorithm to meet the regulations of COLREGs [25]. All these algorithms have displayed the capability of finding collision-free paths in dynamic environments.

From the algorithms mentioned above, one can find that the current research is focused on the short-range collision avoidance problem. These algorithms usually assume that all the target ships are moving at a constant velocity, and the encounter happens in an open area. Although some of them do not rely on the constant velocity assumption [10] [22] or can handle encounters in complex environments [19], none of them discusses the situations where the environment is complex, and the future motion of target ships is unknown. In this paper, we address the problem of design under uncertainty: how we can model the risks in complex environments when the future motion of target ships is unknown, and how this risk modeling can help autonomous ships in path planning.



(a) (b)

**Figure 1:** (a) San Francisco Bay on a regular nautical chart; (b) The extracted pixel map from ENC (ENC ID: US3CA14M). The land areas are in white, and the water areas are in black.

## 3 METHODS

When the future motion of the target ships is unknown, we cannot predict their exact positions. Instead, our proposed algorithm will estimate the intention of target ships and find possible paths they will take. A risk model is developed to assess the level of probability of encountering target ships at a certain position in the future, and the risk-aware A* algorithm is applied to find a path with low accumulated risk of encounters. In this paper, the phrase “target ship” refers to all other ships, and the phrase “own ship” refers to the ship under our control.

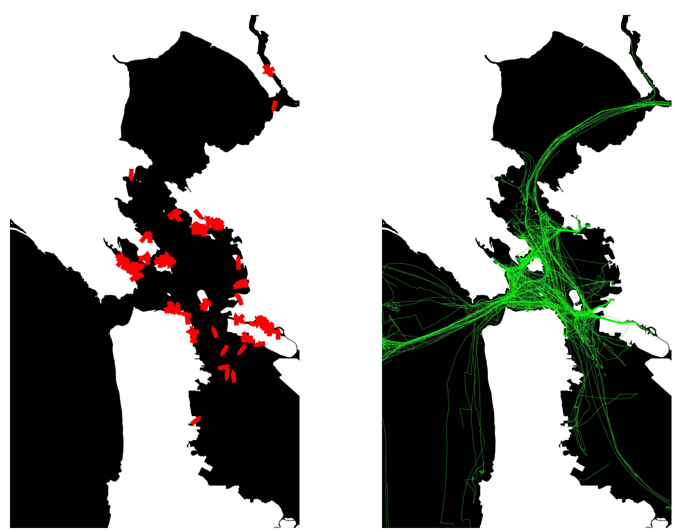
### 3.1 Studied area and data

We choose San Francisco Bay as the area of interest, as shown in Figure 1(a). Several ports locate in this area, and over 100 ships are anchored here. In busy hours, there are more than 50 ships moving in the Bay, from ocean to port, port to port, port to the ocean, etc. This makes it hard to predict the motion of target ships since we do not know their intentions. These ships will not move in straight lines, and they may also change the speed due to the complex environment and encounter situations.

The nautical chart we use in this research is from the National Oceanic and Atmospheric Administration[1] (NOAA), which provides such maps of the ocean in different formats. Our work is based on the electronic nautical chart (ENC). We selected the longitude from 122.67 W to 122.22 W and latitude from 37.54 N to 38.17 N, extracted the information of the land area from the ENC, and constructed a pixel map, as shown in Figure 1(b). The white areas represent the land areas, while the black areas are the water areas. The length change of a degree of longitude or latitude is neglected due to the scale problem and is estimated at (37.84 N, 122.40 W). The ratios are 87.81 km per longitude degree and 111.19 km per latitude degree. In this map, each pixel represents a “10m x 10m” square, which finally produces a map size of 7004 * 3951.

Currently, most ships are required to be equipped with the automatic identification system (AIS), which keeps publishing and receiving the ship information every 2-10 seconds, including the ship’s identity (Maritime Mobile Service Identity, MMSI), position (longitude and latitude), course (direction of motion, in degrees), and speed (in knots). The AIS system can build ship-to-ship and ship-to-port communications, which provides more information to mariners and helps improve navigation safety.

There exist many public AIS datasets, and one can also collect his own data with an AIS device. In this research, we use the public data from MarineCadastre[2]. MarineCadastre has provided daily AIS records since 2018, which includes the MMSI, record time, longitude (LON), latitude (LAT), course over ground (COG), speed over ground (SOG), and heading. We built our test case with the AIS data recorded on 07/04/2020.



(a) (b)

**Figure 2:** (a) AIS records at 21:00, 07/04/2020 ($SOG \leq 0.5$) (b) All ship trajectories recorded on 07/04/2020



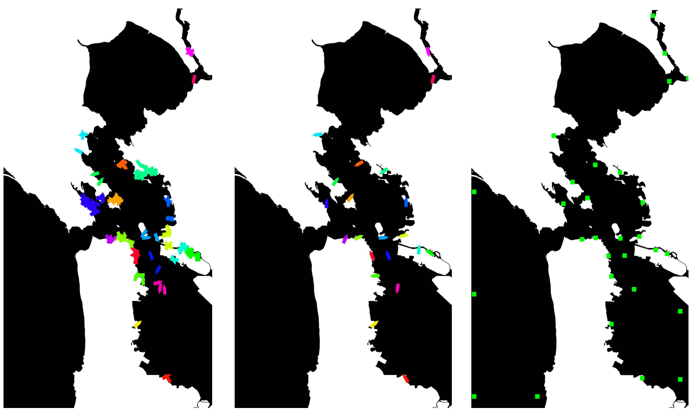

[1] https://www.noaa.gov/

[2] https://www.marinecadastre.gov/

### 3.2 Intention estimation

Figure 2(a) shows all the latest AIS data whose SOG is less than 0.5 knots (around 0.26 m/s) recorded at 21:00, 07/04/2020. These ships are regarded as anchored or stopped, and the speed is caused by either wind or water current. All the ship trajectories are plotted in Figure 2(b). The ships are not taking random actions or steers during the motion. Instead, most of them have a clear destination, e.g., moving to a port or into the ocean. This makes the intention estimation possible, and our approach assumes that each ship has its own destination.

The first step is to choose some positions as the possible destinations. Besides the water area on the margin of the map, a port or an anchoring area on the map can also be a destination. Thus, we collect the AIS data recorded at 21:00 with $SOG \leq 0.5$ and use hierarchical clustering to determine the possible destinations. The distance threshold of the clustering is set as 400 pixels, and the median of each cluster is then used as a possible destination. The position of all the possible destinations is shown in Figure 3.



(a) (b) (c)

**Figure 3:** (a) The clusters of all stopped ships shown in different colors; (b) The median of each cluster; (c) All the possible destinations shown in green squares (cluster medians + water area on margin of the map)

To estimate the level of intention to a certain destination, we need to compare the current movement with the path leading to that destination. If the current motion of the ship aligns with the path, there is a high probability that the ship will follow that path in the future motion. However, each pixel of the water area in the map can be the possible position of a ship. This requires the algorithm to find the path from each pixel to each destination efficiently.

Our approach uses a modified probabilistic roadmap algorithm (PRM). PRM is a popular sampling-based path planning algorithm in robotics [26] [27] [28]. In our problem, we need to find paths from different positions to the same destination. We initialized the vertex set with the position of all possible destinations and did random sampling in the water area to build a roadmap. The vertex number is set to 15,000, and the max neighbor number of each vertex is set to 16. This roadmap covers the water area on the map and can be used to generate a path from each vertex to each destination.

Noticing that the optimal path from a vertex to a destination is also the optimal path from the destination to that vertex, we can build a tree from the destination vertex and span it to cover the space so that a path on the tree is the optimal path from the vertex to the root. Such a tree is referred to as the *destination tree* in this paper. Here we use a modified Dijkstra's algorithm [4] to build the destination tree. Since ships have only the limited steering capability, a steering cost is introduced to build the tree. The path cost is defined as Eq. (1-4).

$$PathCost = DistCost + \alpha \cdot SteerCost \tag{1}$$

$$DistCost = \sum_{i,j} dist\left(v_i, v_j\right) \tag{2}$$

$$SteerCost = max\left(Steer\left(v_i, v_j, v_k\right)\right) + \beta \sum_{i,j,k} Steer\left(v_i, v_j, v_k\right) \tag{3}$$

$$Steer\left(v_i, v_j, v_k\right) = \frac{tan\left(0.5\left(\varphi_{j,k} - \varphi_{i,j}\right)\right)}{min\left(dist\left(v_j, v_k\right), dist\left(v_i, v_j\right)\right)} \tag{4}$$

$v_i, v_j, v_k \in V$ are vertices on the roadmap, and $(v_j, v_k), (v_i, v_j) \in E$ are edges on the roadmap. $dist(v_i, v_j)$ is the length of the edge $(v_i, v_j)$, and is calculated by the Euclidean distance. $\varphi_{i,j}$ is the course angle from $v_i$ to $v_j$. The pseudo-code of the construction of the destination tree is shown in Algorithm 1.

**Algorithm 1. construct_destination_tree**

```
Inputs: vertex_set (V), edge_set (E), root_vertex (d)
heap = [ [0, d] ]    # heap of [cost, Vertex()]
destination_tree = empty
visited = empty set
while heap is not empty:
    cost, vertex = heap.pop()
    if vertex not in visited:
        add vertex to visited
        add vertex to destination_tree
        for neighbor in E[vertex]:
            calculate the new_cost of neighbor by Eq. (1-4)
            push [new_cost, neighbor] into heap
        end for
    end if
end while
return destination_tree
```

The *PathCost* consists of the cost of path length and accumulated steering cost. The steering cost is defined in Eq. 4 and encourages smooth turn. A max function is used in the *SteerCost* in Eq. 2 to avoid the situation where a sharp turn is made to reduce future steering costs. In our case, $\alpha = 1000$,

$\beta = 0.1$. Some examples of the destination trees are shown in Figure 4.

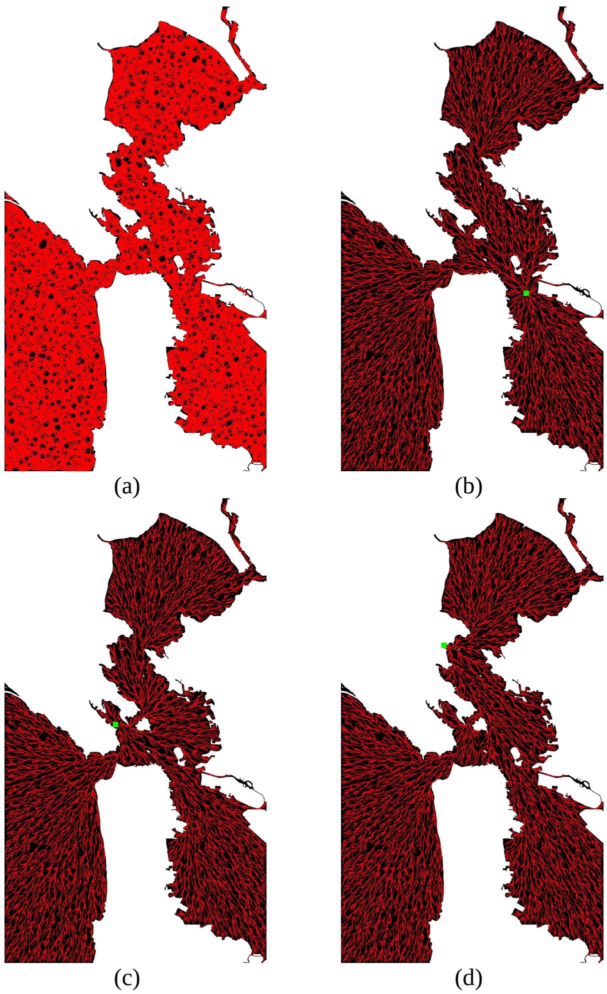



**Figure 4:** The path from each node of the tree to the root is the optimal path to the root regarding the distance and steering cost. The roots are shown in green squares. (a) roadmap built by PRM; (b) root = (1674, 3409); (c) root = (2788, 4320); (d) root = (1500, 2211)

The intention of each ship is then modeled by the level of alignment between the current motion of the ship and the optimal path to each destination. Due to the wind/water current on the ocean, a ship may change its heading to counteract the drifting. Thus, we use the true direction of motion, which is the COG, to evaluate the intention of a ship. A cosine distance is used to determine the intention weight for each destination. Given the current position and COG of a ship, the closest vertex on the roadmap is selected as the starting point, and a path from this vertex to each destination is found on the destination trees. The intention score is calculated by Eq. (5).

$$IntentionScore = \cos\left( COG - \varphi_{start,\ m} \right) \qquad (5)$$

In Eq. (5), COG is the course angle of the ship, and $\varphi_{start,m}$ is the course angle from $v_{start}$ to $v_m$, where $v_m$ is the first vertex along the path that satisfies the length of $Path(v_{start}, v_m)$ is greater than 100-pixel length. The weight of each destination is calculated by the *IntentionScore* by Eq. (6).

$$Weight(d) = \frac{e^{\lambda \cdot IntentionScore(d)}}{\sum_d e^{\lambda \cdot IntentionScore(d)}} \qquad (6)$$

By applying Eq. (6), the weight of each destination will sum up to 1. $d$ is the index of possible destinations. The $\lambda$ is a user-defined coefficient and is set to $\lambda = 7$ in our case.

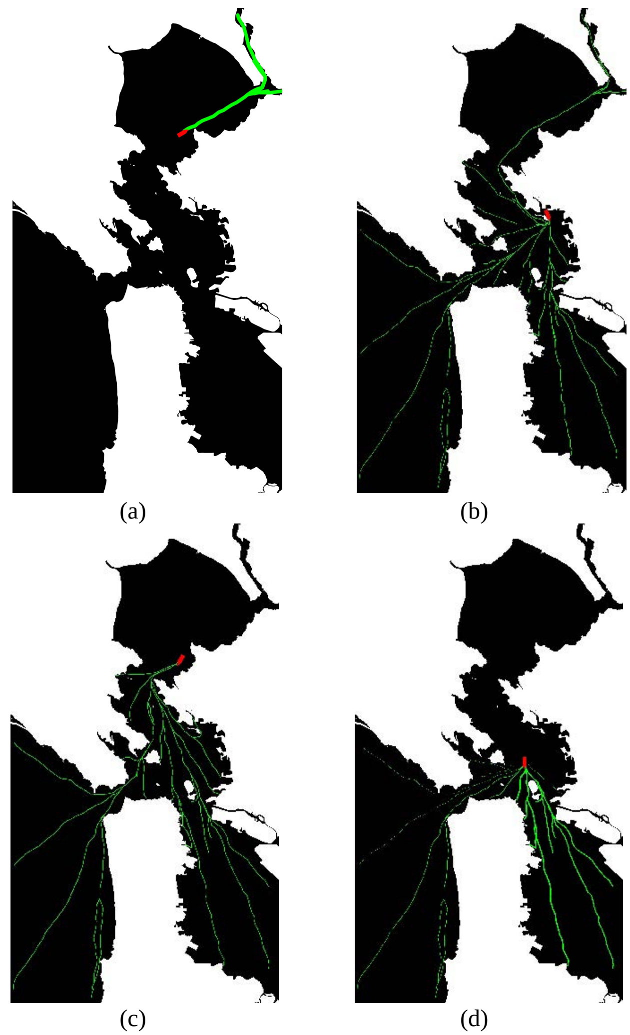



**Figure 5:** Possible paths a ship may take given its initial position and course angle. The line width represents the weight of the path. (a) pos = (2500, 1800), COG = 60; (b) pos = (2800, 3000), COG = 160; (c) pos = (2500, 2000), COG = 210; (d) pos = (2500, 3500), COG = 180.

A visualized result of the weight destination and path is shown in Figure 5. The path to each possible destination is drawn with a green line, and the line width represents the weight of the path, which shows the level of intention that the ship will follow that path.

### 3.3 Risk modeling

After we get all possible paths that a ship may take and the corresponding weights of the paths, we can assess the risk of the own ship encountering a certain ship at a position in a future time. The position is estimated by assuming the ship is moving at a constant speed along the path, but a circle centered at that future position will be regarded as a risky area since the ship will not strictly move along the estimated path, and its speed may change during the motion.

Considering that the average speed of the ship can be different from that in the latest AIS record, as time goes on, the error of future position estimation will become larger and larger. Thus, we use a Gaussian model with time-varying variance in the sense that the long-term estimation will not be as reliable as a short-term estimation. The risk model is described in Eq. (7-8).

$$Risk(x,y,t) = \sum_k \sum_d Weight(d) \cdot \frac{1}{\sigma(t)} \cdot e^{-\frac{(x-x_k(t))^2 + (y-y_k(t))^2}{2\cdot\sigma(t)^2}} \tag{7}$$

$$\sigma(t) = \sigma_0 + \sigma_1 \cdot t \tag{8}$$

In Eq. (7), *d* is the index of possible destinations, and *k* is the index of target ships. The total risk value is calculated by summing up the weighted risk value received at position *(x, y)* from each target ship. The time-varying variance of the risk function is defined in Eq. (8). The selection of coefficients in Eq. (8) should be careful. On one hand, when the variance is too large, almost everywhere will be identified as risky, which is not meaningful; on the other hand, when the variance is too small, most of the area will be identified as not risky, which does not help the decision making. A good risk model should show better performance in the risk value by intention estimation than that by the constant speed assumption. It remains an open question how to select the coefficients. In this paper, the coefficients are selected arbitrarily, so that the average risk value received by the intention estimation at the true future position of target ships is higher than that by the constant speed estimation. In our case, $\sigma_0 = 10$ and $\sigma_1 = 0.02$.

### 3.4 Risk-aware A* algorithm

A modified risk-aware A* algorithm is used in this study to find a path on the roadmap with a low cost for the own ship. The cost function is defined as a weighted sum of the distance cost, steering cost, and risk cost, as defined in Eq. (9-13).

$$ownPathCost = ownDistCost + \alpha \cdot ownSteerCost + \gamma \cdot ownRiskCost \tag{9}$$

$$ownDistCost = \sum_{i,j} dist(v_i, v_j) \tag{10}$$

$$ownSteerCost = max(ownSteer(v_i, v_j, v_k)) + \beta \sum_{i,j,k} ownSteer(v_i, v_j, v_k) \tag{11}$$

$$ownSteer(v_i, v_j, v_k) = \tan(0.5(\varphi_{j,k} - \varphi_{i,j})) \tag{12}$$

$$ownRiskCost = \sum_i Risk(x_i, y_i, t_i) \tag{13}$$

Like the definition in Eq. (1-4), $v_i, v_j, v_k \in V$ are vertices on the roadmap, and $(v_j, v_k), (v_i, v_j) \in E$ are edges on the roadmap. The $x_i, y_i$ in Eq. (13) is the position of vertex $v_i$, and $t_i$ is the arrival time of that vertex. Thus, for the same vertex on the road, the arrival time and the cost may differ if the parent path is different.

Notice that when there is no target ship, the risk value will be zero, and the path cost will only depend on the distance and steering cost. Thus, like how we build the destination tree, we can build a tree from the goal vertex using the cost function from Eq. (9-12) with zero risk cost. The path from each vertex to the goal vertex on the goal tree is the optimal path when there is no target ship, and the cost received at each vertex on the goal tree will be used as the heuristic distance in the risk-aware A*. The pseudo-code of the risk-aware A* is shown in Algorithm 2.

**Algorithm 2. risk-aware A***

**Inputs:** vertex_set (V), edge_set (E), start_vertex (s), goal_vertex (g), goal_tree(GT)

```
heap = [ [0, 0, None, s] ]    # heap of [cost_f, cost_g, parent, Vertex()]
child_parent_set = empty set
visited = empty set
while heap is not empty:
    cost_f, cost_g, parent, vertex = heap.pop()
    if vertex == g:
        break
    end if
    if vertex not in visited:
        add vertex to visited
        add [vertex, parent] to child_parent_set
        for neighbor in E[vertex]:
            calculate cost_g of neighbor by Eq. (9-13)
            cost_h = cost_from_GT(neighbor, GT)
            cost_f = cost_g + cost_h
            push [cost_f, cost_g, vertex, neighbor] into heap
        end for
    end if
end while
path = find_path_from(child_parent_set)
return path
```

The risk-aware A* algorithm is complete in space, which guarantees to visit every vertex in the connected graph (V, E). However, each node will only be visited once, which implies that the solution is not complete in the time domain. That is to say, the risk-aware A* will try to find a smooth path with low risk instead of taking a detour or even circular motions to find a risk-free path.

## 4 CASE STUDY

Our proposed algorithm is applied to make a long-range plan in complex environments. Our algorithm aims to reduce the duration of encounters and the number of ships encountered at the same time, thus reducing the complexity of the encountering situation and helping autonomous ships make safe navigation decisions.

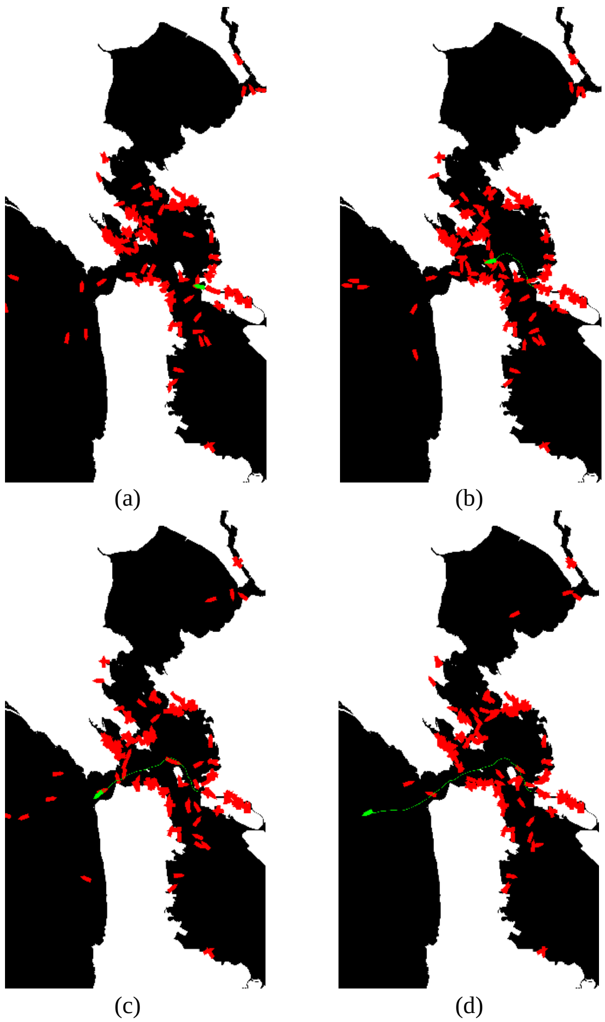


**Figure 6:** A sample path taken by the proposed risk-aware A* in one of the test cases. The own ship and the path are shown in green. (a) t = 0; (b) t = 1970; (c) t = 3940; (d) t = 5910.

### 4.1 Environment setup

The test data comes from the true AIS records on 7/4/2020. We used the data recorded from 21:56:57 to the end of the day and selected the ship with MMSI 477655900 as our reference ship.

The position of the reference ship at 21:56:57 is used as the starting point, and the position recorded at 22:59:56 is used as the goal point.

The SOG recorded at the starting point is 10 knots (around 5.14 m/s). The length of the recorded path of the reference ship is 29.62 km. The reference ship accelerated during the motion, and it took the reference ship about an hour to reach the goal point.

In our case study, we remove the AIS records of the reference ship and use all the remaining AIS records to rebuild the situation. Our proposed algorithm is used to estimate the risk in the environment, and the risk-aware A* is applied to find a path from the starting point to the goal point. Considering that many of the ships are stopped, we set a sog_threshold to trigger the intention estimation. Only those whose SOG is greater than sog_threshold will use the intention estimation to assess future risk. Otherwise, the future position of the ship will be predicted by assuming it is moving at a constant velocity. For example, when sog_threshold is 2, the risk value from those whose SOG is less than 2 knots will be estimated by letting them move forward at the speed of 2 knots, and the future position and risk of those moving faster than 2 knots will be estimated by letting them follow the paths leading to the possible destinations.

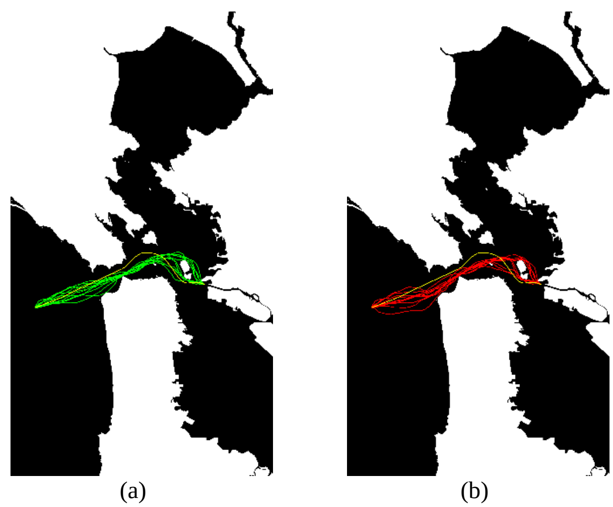


***Figure 7:** The paths taken by the planner based on intention estimation and constant velocity assumption on 10 randomly generated roadmaps. The yellow path is a human-taken path recorded in AIS data. (a) the 10 paths taken by the planner based on intention estimation (green); (b) the 10 paths taken by the planner based on constant velocity assumption (red).*

The own ship plans a path at the beginning of motion with the latest AIS data received at that moment and follows the planned path with constant speed without further planning. The planned path of one test case is shown in Figure 6. The own ship needs to find a safe path to go through the narrow area, where there are ports on either side, and a lot of ships are moving in this area. It is inevitable to encounter other ships in such a situation,

**Table 1.** Performance comparison of different planners on the 10 randomly generated roadmaps.

| Planner | Weight of RiskCost ($\gamma$) | SOG threshold (knots) | Path length (km) | Duration (encounter ≥ 1) | Duration (encounter ≥ 2) | Duration (encounter ≥ 3) |
|---|---|---|---|---|---|---|
| Human | / | / | 29.62 | 1910.0 | 657.0 | 10.0 |
| Constant velocity | 10000 | infinity | 29.45 | 2206.9 | 728.1 | 143.1 |
| Constant velocity | 20000 | infinity | 30.05 | 2054.1 | 623.4 | 158.9 |
| Intention based | 10000 | 2 | 29.46 | **2065.0** | **609.0** | **54.7** |
| Intention based | 20000 | 2 | 30.19 | **1900.2** | **475.2** | **45.1** |

and our objective is to minimize the duration of encounters and reduce the number of ships encountered at the same time.

### 4.2 Results

In this case study, we compare the result of the plan using intention estimation with the plan assuming all ships moving at constant velocity. This comparison is realized by setting different sog_threshold. When sog_threshold is set to 2 knots, the intention estimation is activated, and the risk will be analyzed based on the possible path a ship may take. When sog_threshold is set to infinity, the future position will be barely predicted with the constant velocity assumption.

The effect of different weights $\gamma$ of risk cost is also shown in the result of the case study. By setting different values, the level of risk tolerance can be modified to decide whether the own ship will take a shortcut with a higher risk of encounters. Since the roadmap construction depends on random sampling, we generate 10 roadmaps and test the performance of both planners on them. The result of the case study is shown in Table 1, and the human performance is also listed in Table 1.

The performance is evaluated by the path length and the duration of encounters. In this case, if a ship appears within 0.5 nautical miles (0.926 km) to the own ship, it is regarded as encountered. Considering that encountering multiple ships at the same time is more complex, we show the duration of encounters by the number of ships encountered. The duration of encountering at least 1 ship, at least 2 ships, and at least 3 ships are shown in Table 1. We can see that the mean path lengths of the two planners are similar, but the encounter duration of the one based on intention estimation is much shorter than the one based on the constant velocity assumption. The paths taken by the two planners are plotted in Figure 7. Although the paths by the two planners look similar, Table 1 shows that the performance of our proposed algorithm is much better, which proves that the intention estimation can make a better assessment of the future risk.

## 5 DISCUSSION

As shown in Table 1, our proposed algorithm for autonomous ships outperforms the planner, which assumes all ships are moving at a constant speed. The average lengths of the paths planned by the two planners are similar, but the duration of encounters by our proposed algorithm is much shorter. Our algorithm can better assess the risk of encountering other ships in the long term. It takes the own ship around 100 minutes to reach the goal, and the constant velocity and course direction assumption no longer hold in such a long-time horizon in complex environments.

Our algorithm utilizes the information gained from the map and the AIS data and finds possible destinations for each ship. Since the optimal path from the goal to a starting point is also the optimal path from that starting point to the goal, it is possible to construct a goal tree for each destination that spans the water area and provides a near-optimal path from each point in the water area to the destination. By this approach, the pathfinding task for each ship to each destination can be reduced to the problem of finding the closest vertex on the destination tree and retrieving the path from the tree.

This makes it possible to include the risk assessment in the risk-aware A* algorithm. Since the construction of the roadmap and the destination trees just need to be done once, the time complexity of the risk calculation at each vertex is $O(N \cdot D \cdot \log(p))$, where N is the number of ships, D is the number of possible destinations, and p is the number of vertices on the path. The overall complexity of the risk-aware A* algorithm is $O((|V| + |E|) \cdot N \cdot D \cdot \log(p))$, where |V| and |E| are the number of vertices and edges on the roadmap, respectively.

From Table 1, the path planned by human shows better performance in the duration of encountering at least 3 ships. This result is reasonable since the human keeps receiving real-time information and has more flexibility in the control of the ship. The human can change the speed of the ship to drive through risky areas quickly, and the real-time information also helps the human to modify the plan according to the situation.

Currently, this algorithm does not satisfy the real-time requirement. It takes around 16 minutes and 40 seconds for a gaming laptop to generate a plan in such a complex environment in the test case. The code is written in Python, and the calculation is done by an AMD Ryzen 7 5800H CPU with 3.20 GHz speed and 16 GB RAM. During the planning, the risk value at each vertex is the sum of the risk value by each target ship, and the risk value of each target ship is calculated in parallel by 16 threads. The complexity of the proposed algorithm grows linearly with the number of ships. In the test case, there are more than 200 ships in this area, and over 50 of them are moving. Future studies on intention modeling and risk assessment are needed to improve the efficiency of the algorithm.

## 6 CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a risk-aware path planning algorithm for autonomous ships navigating in complex and dynamic environments. Hierarchical clustering is applied to determine all possible destinations of other ships, and a pathfinding algorithm based on probabilistic roadmap algorithm and modified Dijkstra's algorithm is developed to find the possible path an unmanned ship may take. This approach does not rely on the constant velocity assumption, which is common in similar research.

A time-varying Gaussian model is used to assess the risk of encountering a ship at a future time. Our proposed algorithm will find a path with a low risk of encountering other ships. The risk-aware A* algorithm can be used to reduce the duration of encounters and reduce the number of ships encountered at the same time. This helps lower the complexity of encounter situations and can help autonomous ships make safe navigation.

The proposed algorithm can make a long-range plan in complex and dynamic environments. In the test case, it takes the own ships 100 minutes to reach the goal, and the risk of encountering other ships can be properly assessed by our algorithm. Our algorithm has a much shorter duration of encounters against the same planner with the constant velocity assumption. This implies that the constant velocity assumption does not hold in such a complex situation, and our algorithm can handle the situation well. The average encountering time of the paths planner by our algorithm is comparable to that of the human.

However, the human path has a longer duration encountering at least two ships and a shorter duration encountering at least three ships. This implies that the human is sacrificing the overall duration of encounters to avoid the complex situation of encountering many ships at a time. Thus, the detailed modeling of different encountering situations will be one of our future research directions. Another direction is to improve the efficiency of the risk assessment algorithm. The current approach takes all target ships into consideration, while many of them may not be threatening to the own ship. A better risk assessment process will be developed in our future research for developing a highly intelligent path planning and collision avoidance framework for autonomous ships.

## ACKNOWLEDGEMENTS

This paper is based on the work supported by the Autonomous Ship Consortium (ASC) with members of BEMAC Corporation, ClassNK, MTI Co. Ltd., Nihon Shipyard Co. (NSY), Tokyo KEIKI Inc., and National Maritime Research Institute of Japan. The authors are grateful for their support and collaboration on this research.

## REFERENCES

[1] Asariotis,R., Ayala,G., Assaf,M., Bacrot,C., Benamara,H., Chantrel,D.,Cournoyer,A.,et al., 2021, Review of Maritime Transport 2021. https://unctad.org/webflyer/review-maritime-transport-2021

[2] EMSA., 2021, "Annual Overview of Marine Casualties and Incidents". http://www.emsa.europa.eu/newsroom/latest-news/download/6955/4266/23.html

[3] International Maritime Organization. 1972. "COLREGs: Convention on the International Regulations for Preventing Collisions at Sea."

[4] Sun, L., Zhao, Y., and Zhang, J., 2021, "Research on Path Planning Algorithm of Unmanned Ship in Narrow Water Area." Journal of Physics. Conference Series 2029 (1) (Sep 01,)

[5] Zaccone, R., Martelli, M., 2018, "A Random Sampling Based Algorithm for Ship Path Planning with Obstacles" In Proceedings of the International Ship Control Systems Symposium (iSCSS), vol. 2, p. 4.

[6] Zaccone, R., Martelli, M., and Figari M., 2019, "A COLREG-Compliant Ship Collision Avoidance Algorithm." In 2019 18th European Control Conference (ECC), pp. 2530-2535. IEEE.

[7] Chiang, H. and Tapia, L., 2018, "COLREG-RRT: An RRT-Based COLREGS-Compliant Motion Planner for Surface Vehicle Navigation." IEEE Robotics and Automation Letters 3 (3) (Jul): 2024-2031

[8] Naeem, W., Henrique, S. C., and Hu, L., 2016, "A Reactive COLREGs-Compliant Navigation Strategy for Autonomous Maritime Navigation." IFAC PapersOnLine 49 (23): 207-213

[9] Mei, J. and Arshad, M. R., 2018, "A Smart Navigation and Collision Avoidance Approach for Autonomous Surface Vehicle." Indian Journal of Geo Marine Sciences 46 (12): 2415-2421

[10] Lyu, H. and Yin, Y., 2019, "COLREGS-Constrained Real-Time Path Planning for Autonomous Ships using Modified Artificial Potential Fields." Journal of Navigation 72 (3) (May): 588-608

[11] Lazarowska, A., 2015, "Ship's Trajectory Planning for Collision Avoidance at Sea Based on Ant Colony Optimisation." Journal of Navigation 68 (2) (Mar): 291-307

[12] Tam, C. and Bucknall R., 2010, "Path-Planning Algorithm for Ships in Close-Range Encounters." Journal of Marine Science and Technology 15 (4) (Jun 02,): 395-407

[13] Wang, Y., Yao, P., and Dou, Y., 2019, "Monitoring Trajectory Optimization for Unmanned Surface Vessel in Sailboat Race." Optik (Stuttgart) 176 (Jan): 394-400

[14] Kang,Y., Chen,W., Zhu,D., Wang,J., Xie,Q., 2018, "Collision Avoidance Path Planning for Ships by Particle Swarm Optimization". Vol. 26. doi:10.6119/JMST.201812_26(6).0003

[15] Kim, H., Kim, S., Jeon, M., Kim, J., Song, S., and Paik, K., 2017, "A Study on Path Optimization Method of an Unmanned Surface Vehicle Under Environmental Loads using Genetic Algorithm." Ocean Engineering 142 (Sep 15,): 616-624
[16] Liu,X., Jin,Y., "Artificial Intelligence for Engineering Design, Analysis and Manufacturing Reinforcement Learning-Based Collision Avoidance: Impact of Reward Function and Knowledge Transfer", Artificial Intelligence for Engineering Design,Analysis and Manufacturing: 1–16, https://doi.org/ 10.1017/ S0890060420000141
[17] Wu, X., Chen, H., Chen, C., Zhong, M., Xie, S., Guo, Y., and Fujita, H., 2020, "The Autonomous Navigation and Obstacle Avoidance for USVs with ANOA Deep Reinforcement Learning Method." Knowledge-Based Systems 196 (2020): 105201
[18] Singh, Y., Sharma, S., Sutton, R., Hatton, D., and 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 (Dec 01,): 187-201
[19] Liu, Y. and Bucknall, R., 2015, "Path Planning Algorithm for Unmanned Surface Vehicle Formations in a Practical Maritime Environment." Ocean Engineering 97 (Mar 15,): 126-144
[20] Yan, X., Wang, S., Ma, F., Liu, Y., and Wang, J., 2020, "A Novel Path Planning Approach for Smart Cargo Ships Based on Anisotropic Fast Marching." Expert Systems with Applications 159 (2020): 113558
[21] Beser, F. and Yildirim, T., 2018, "COLREGS Based Path Planning and Bearing Only Obstacle Avoidance for Autonomous Unmanned Surface Vehicles." Procedia Computer Science 131: 633-640
[22] Williams, E. and Jin, Y., 2019, "Dynamic Probability Fields for Risk Assessment and Guidance Solutions." Annual of Navigation 26 (1) (Dec 01,): 33-45
[23] He, Y., Li, Z., Mou, J., Hu, W., Li, L., and Wang, B., 2021, "Collision-Avoidance Path Planning for Multi-Ship Encounters Considering Ship Manoeuvrability and COLREGs." Transportation Safety and Environment 3 (2): 103-113
[24] Song, L., Chen, Z., Dong, Z., Xiang, Z., Mao, Y., Su, Y., and Hu, K., 2019, "Collision Avoidance Planning for Unmanned Surface Vehicle Based on Eccentric Expansion." International Journal of Advanced Robotic Systems 16, no. 3 (2019): 1729881419851945.
[25] Campbell, S. and Naeem, W., 2012, "A Rule-Based Heuristic Method for COLREGS-Compliant Collision Avoidance for an Unmanned Surface Vehicle." IFAC Proceedings Volumes 45 (27): 386-391
[26] Karaman, S. and Frazzoli, E., 2011, "Sampling-Based Algorithms for Optimal Motion Planning." The International Journal of Robotics Research 30 (7) (Jun): 846-894
[27] Hsu, D., Kindel, R., Latombe, J., and Rock, S., 2002, "Randomized Kinodynamic Motion Planning with Moving Obstacles." The International Journal of Robotics Research 21 (3) (Mar): 233-255
[28] Bohlin, R. and Kavraki, L.E., 2000, "Path Planning using Lazy PRM." In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), vol. 1, pp. 521-528. IEEE, 2000.