A Search-Based Collision Avoidance Strategy with COLREGs for Autonomous Surface Vehicles

K.K. Jaiswal, M.S. Khan, R.V. Kulkarni and P.B. Sujit

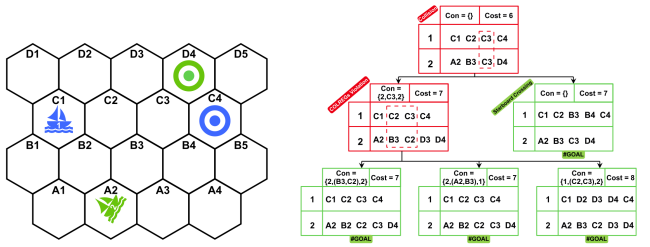
***Abstract*— Autonomous surface vehicles (ASVs) need to navigate in the sea while following COLREG rules. The collision avoidance planners should provide guarantees on collision avoidance while maintaining near-optimal plans. Due to the inclusion of the COLREGs, most of the planners are reactive and do not provide guarantees. To meet these requirements, in this paper, we present a search-based planning strategy that is computationally efficient, COLREGs-compliant, complete, and provides guarantees on the collision-free path. The proposed strategy is evaluated through simulations for different scenarios. We also perform real-world experiments to demonstrate the performance of our approach.**

I. INTRODUCTION

Collision avoidance and path planning for Autonomous Surface Vehicles (ASVs) represent critical research areas in maritime robotics. ASVs serve diverse applications, including maritime surveillance, hydrographic surveying, and patrol operations [1]. All vessels (manned and unmanned) operating on the high seas need to adhere to International Regulations for Preventing Collisions at Sea (COLREGs) [2] to ensure safe and predictable navigation. This regulatory compliance requirement has driven the development of various COLREGs-compliant path planning strategies. Several approaches have emerged to address this challenge.

Potential field-based methods [3] employ reactive potential fields that generate repulsive forces from obstacles and other vessels while attracting the ASV toward its target destination. Optimization-based strategies [4] utilize Model Predictive Control (MPC) frameworks that predict system behaviour over finite horizons and optimize control actions at each timestep to maintain safe trajectories. However, these conventional approaches face significant limitations in complex multi-vessel scenarios. When an ASV encounters four or more vessels simultaneously, traditional potential field and optimization methods often fail to generate feasible collision-free paths due to local minima problems and computational constraints [5]. Recent advances in reinforcement learning have shown promise in addressing these limitations. Multi-ship collision avoidance systems based on deep reinforcement learning [5] demonstrate improved performance in complex scenarios while maintaining compliance with COLREGs through collaborative decision-making frameworks. These learning-based approaches can handle intricate multi-vessel encounters that challenge traditional methods. Nevertheless, reinforcement learning solutions present their own challenges. Learning-based models trained on specific scenarios and datasets may not generalize effectively to novel situations, potentially compromising collision avoidance guarantees in real-world deployments. This raises the need to formulate a multi-vessel path-finding strategy that provides theoretical guarantees while being COLREGs-compliant. Multi-Agent Pathfinding (MAPF) [7] is a class of Artificial Intelligence problems that involves finding collision-free optimal or sub-optimal paths for all agents in a shared environment. Conflict-Based Search (CBS) [6] is an optimal search-based MAPF algorithm that guarantees optimality and completeness within its search space. We propose a CBS-based path-finding strategy that preserves the claims of optimality and completeness and is also COLREGs-compliant. The main contributions of this paper are:

- A COLREGs-compliant multi-agent path-finding strategy for ASVs.
- Theoretical proofs of the algorithm's completeness and optimality
- Demonstration of the algorithm via simulations and field experiments.



(a) CBS Problem (b) CBS Tree

Fig. 1. Path finding problem involving two agents, along with the corresponding CBS tree, which also incorporates COLREGs. A vertex constraint in the first node results in the creation of two child nodes; however, a crossing rule violation within one of these child nodes, where the stand-on vessel deviates and the give-way vessel maintains its course, creates three child nodes and ensures conflict resolution.

II. PRELIMINARIES

A. Conflict-Based Search

CBS is a centralized, complete, and optimal MAPF that efficiently computes collision-free trajectories by iteratively resolving conflicts between agents [6]. The algorithm works by first using a low-level path planner such as A* [9] to generate individual paths for each agent. It then checks these paths for collisions–either vertex conflicts (agents occupying

This work was partially supported by SERB CRG/2021/007916.

The authors are with Department of Electrical Engineering and Computer Science, IISER bhopal, bhopal – 462066, India. Email: (kkatyayanj24,mohammad21,rahulv21,sujit)@iiserb.ac.in

the same location at the same time) or edge conflicts (agents crossing the same path segment simultaneously).

When the conflicts are detected, CBS resolves them by adding constraints and re-planning the affected paths using the low-level planner. This process repeats until all conflicts are eliminated. The core innovation of CBS lies in its strategy of splitting constraints among agents and incrementally applying them, ensuring an optimal and conflict-free solution for the entire multi-agent system. An illustration of CBS is shown in Figure 1.

B. COLREGs

Within the path planning algorithm, we incorporate two key regulations specified in the COLREGs [10]. These two rules are incorporated as they cover the primary encounter scenarios relevant to most multi-ASV scenarios. Since all ASVs are assumed to operate at the same speed, overtaking situations (covered by Rule 13) do not occur and are therefore excluded from the scope of our problem formulation.

1) Head-On Rule (Rule No. 14): When two vessels are approaching each other head-on (i.e., on nearly opposite courses with a risk of collision), both must alter course to starboard (right) so that they pass port-to-port (left side to left side) as shown in Figure 2(d).

2) Crossing Rule (Rule No. 15): When two vessels are crossing paths with a risk of collision, the vessel that has the other on its starboard (right) side is the "give-way" vessel and must take early action to avoid crossing ahead of the other. The vessel on the port (left) side is the "stand-on" vessel and should maintain its course and speed unless a collision becomes imminent, as shown in figure 3(d).

III. Problem Formulation

To make the Pathfinding of ASVs computationally efficient, the space is considered as a hexagonal grid V where each ASV, from a set of k ASVs, searches for paths from one hexagonal cell v_1 to a destination cell v_3, where $v_1, v_3 \in V$. The trajectory returned by the algorithm is a sequence of cells, $v_1^0 v_2^1 v_3^2$, where the superscripts indicate the timestep of cell occupation. The action of waiting at a particular cell is prohibited due to the high costs involved in bringing surface vessels to an intermediate halt; therefore, the only action an ASV can take, at each timestep, is to move to an adjacent cell. The cost of each action is 1. A collision could be a vertex or an edge conflict as described within the CBS algorithm. Let M be the set of all collision-free paths for the k agents that are also COLREGs-compliant, then our objective is to generate a set of collision-free paths $\{p_1, \ldots, p_k\} \in M$ such that

$$\min F = \sum_{i=1}^{k} t_{p_i}, \tag{1}$$

where F is the cost function and t_{p_i} is the number of actions agent i makes to traverse path p_i. Note that within this text, an optimal path to a node refers to the minimum number of actions, or centroid-to-centroid traversals, it takes for the agent to reach that node from the start node. Since an agent cannot wait, the optimal path can also be defined as the minimum number of timesteps it takes to reach that node.

IV. Method

A. COLREGs-violation to Conflicts

We develop a strategy to detect COLREGs violations among the trajectories of ASVs within the Conflict-Based Search framework. We categorize these violations as additional collisions. Next, appropriate constraints are formulated for resolving these conflicts. Within our grid setup, we create grid-based distance metrics to quantify various qualitative phrases, such as "approaching each other head-on" and "at-risk of collision", mentioned within the COLREGs rules. The quantification for both phrases is shown within Figure 2(a) and 3(a). Note that, within the mentioned figures, extrapolation of the vessel trajectories by one timestep could lead to a collision. The CBS framework already incorporates collision resolution constraints for the vertex and edge collisions. Within the current problem scope, only two types of collisions are considered: vertex collisions and edge collisions. Out of all the non-colliding extrapolations, some are classified as COLREGs-violated behaviour, in Figures 2 and 3, and all such extrapolations are identified.

B. COLREGs Conflict Resolution

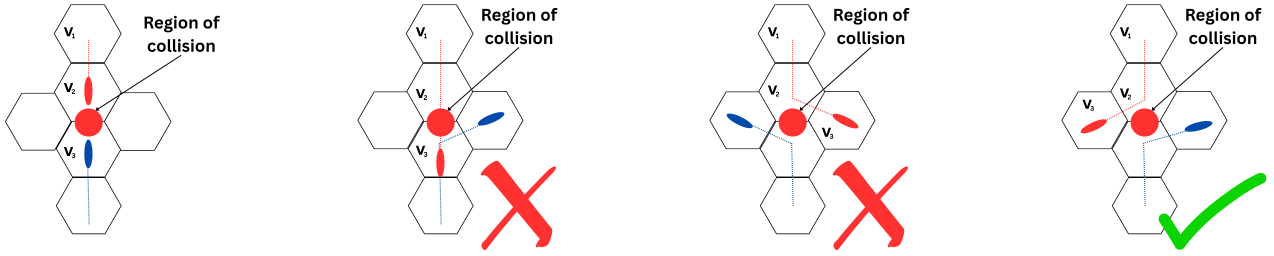
Similar to collision resolution in CBS, we create appropriate constraints to inhibit the identified extrapolated trajectories labeled as COLREGs-based conflict. We discuss the COLREGs-conflict resolution for each case separately. Figure 2(a) illustrates two vessels approaching each other head-on. Now, if the extrapolated trajectory of at least one of the vehicles violates the COLREGs, we classify this scenario as a COLREGs conflict. It can be noted that, according to the metric introduced in figure 2(a), a vehicle involved in a head-on collision avoidance scenario traverses across exactly three vertices. The waypoints between the grids are assumed to be distant enough to execute the intermediate motion. To inhibit any such traversal, we use multiple edge constraints where for an agent a traversing vertices $v_1^t, v_2^{t+1}, v_3^{t+2}$, an edge constraint can be used to inhibit the traversal from v_2 to v_3 at timestep t+1 while another inhibits the movement from v_1 to v_2 at timestep t can be used. We resolve the violations of the crossing rule in a similar manner.

C. Low-Level Planner

We use A* and its variants as our low-level planners [16] [9]. We elaborate further on the properties of low-level planners in the theoretical analysis section.

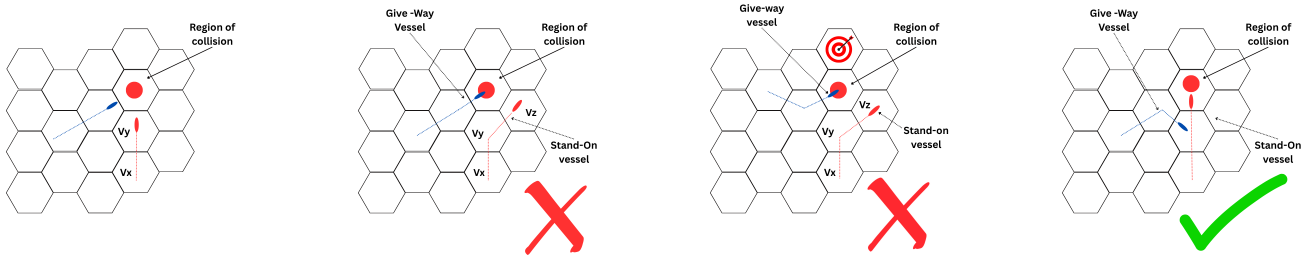
D. COLREGs-compliant CBS

The high-level node generation is similar to CBS, except for the number of nodes generated while resolving the COLREGs-based conflicts. While resolving a head-on-rule-based conflict, as shown in Figure 2, we create two high-level nodes for each agent. Within the first node, we add an edge constraint to inhibit the vessel from approaching the other vessel head-on; that is, for the red vessel, we add an edge



(a) At-risk of head-on collision (b) Incorrect Collision Avoidance (c) Incorrect Collision Avoidance (d) Correct Collision Avoidance

Fig. 2. Different head-on collision avoidance scenarios. (a) Scenario depicting a case where two vehicles are at risk of collision, that is, extrapolating the trajectory over the prior timestep leads to a collision in the region of collision. (b) The give-way vessel travels without a change in direction; however, the stand-on vessel changes direction. (c) The red vessel changes direction to avoid a vessel to its left (portside); however, its goal lies along the path it was following. (d) Correct head-on avoidance strategy.



(a) At-risk of crossing collision (b) Incorrect Collision Avoidance (c) Incorrect Collision Avoidance (d) Correct Collision Avoidance

Fig. 3. Different crossing collision avoidance scenarios. (a) Scenario depicting a case where two vehicles are at risk of collision, that is, extrapolating the trajectory over the prior timestep leads to a collision in the region of collision. (b) The give-way vessel travels without a change in direction; however, the stand-on vessel changes direction. (c) The red vessel changes direction to avoid a vessel to its left (portside); however, its goal lies along the path that it was following. (d) Correct crossing avoidance strategy.

constraint to inhibit the movement from v_1 to v_2. Within the second high-level node, we add an edge constraint to constrain the vessel from making the transition from v_2 to v_3. Therefore, a total of four high-level nodes are created while resolving a conflict. For resolving a crossing conflict as shown in figure 3(a), the give-way vessel is constrained using an edge constraint to alter its direction and prevent it from entering the collision-prone region as shown in figure 3(d). To constrain the stand-on vessel, we create two high-level nodes. Within the first high-level node, we inhibit the agent from being in the collision-risk region as shown in Figure 3(b) by using an edge constraint to inhibit the motion from v_x to v_y. For the second node, we use an edge constraint to prevent the vessel from performing the collision-avoiding action.

V. Theoretical Analysis

The following lemma, whose proof we include for completeness, may be known, but we could not locate a reference. Within the proof, costs mean the number of timesteps or actions required for traversing the path, which is different from the f-cost or K-cost used by A* for expanding nodes.

Lemma 1: For A* employed as a low-level planner in Conflict-Based Search, where agents face vertex and edge conflict constraints but lack wait capabilities, the A* stores optimal and sub-optimal sub-trajectories to nodes if it is complete and optimal.

Proof: We show that A*, which only stores optimal paths to each node, is neither optimal nor complete. Consider the case, as shown in the figure 4, where the orange vertices are all obstacles and A* only preserves the optimal trajectories to each vertex. Note that to reach the goal, the agent must pass through the vertices $C3$ and $D3$. Let t be the optimal time cost for reaching $C3$, and let there be an edge constraint inhibiting the agent from making the transition from $C3$ to $D3$ at timestep t. Therefore, the low-level planner is neither optimal nor complete since there is no other path to $D3$. Therefore, a complete and optimal A* implies that it stores optimal and sub-optimal costs to a node. ■

Traversing to an adjacent node takes one timestep. Let n' and n be adjacent nodes, such that the path to n via n' is suboptimal. For storing this path of higher cost to n, we create a state for the node n and add it to the open list with cost $K(n,T+1) = K(n',T)+c(n',n)+(h(n)-h(n'))$ for expansion, where T is time step corresponding to the state with cost $K(n',T)$ & $c(n',n)$ denotes the distance between n & n'. Rather than using the f-heuristic for arranging nodes in a heap, we use the K-costs, allowing multiple states of the same node, corresponding to different timesteps, to be stored within the open list and expanded incrementally according to their K-costs.

Lemma 2: For A* employed in CBS, where agents face vertex and edge conflict constraints but lack wait capabilities,

Algorithm 1: COLREGs-compliant CBS

Input: Multi-ASV Pathfinding problem
Output: COLREGs-Compliant trajectories
$Root.constraints$ = 0;
$Root.solution$ = find individual paths using the low level();
$Root.cost$ = Sum of the costs of individual paths ;
insert $Root$ to OPEN;
while *OPEN is not empty* **do**
  P ← best node from OPEN // lowest solution cost;
  Search the paths in P for a conflict;
  if *No conflict found* **then**
    return $P.solution$;
  end
  Q ← first conflict in P at timestep t b/w agents a_i and a_j;
  if $Q = (a_i, a_j, v, t)$ *// Vertex Conflict* **then**
    Constraints = $((a_i, v, t), (a_j, v, t))$
  end
  else if $Q = (a_i, a_j, v_1, v_2, t)$ *// Edge Conflict* **then**
    Constraints = $((a_i, (v_1, v_2), t), (a_j, (v_2, v_1), t))$
  end
  else if $Q = (a_i, a_j, v', t)$ *//Crossing Conflict* **then**
    a_i ← give-way vessel, a_j ← stand-on vessel
    a_i.Constraint ← $(a_i, (v_{a_i}^{t-1}, v), t)$
    a_j.Constraint ← $(a_j, (v_{a_j}^{t-1}, v), t)$
    a_j.Constraint ← $(a_j, (v_{a_j}^{t-2}, v), t-1)$
  end
  else if $Q = (a_i, a_j, (v_1', v_2'), t)$ *//Head-On Conflict* **then**
    a_i.Constraint ← $(a_i, (v_1', v_{a_i}^{t+1}), t+1)$
    a_i.Constraint ← $(a_i, (v_{a_i}^{t-1}, v_1'), t)$
    a_j.Constraint ← $(a_j, (v_2', v_{a_j}^{t+1}), t+1)$
    a_j.Constraint ← $(a_j, (v_{a_j}^{t-1}, v_2'), t)$
  end
  for *each q in* $Constraints$ **do**
    A ← A new node;
    $A.constraints \leftarrow P.constraints + q$;
    $A.solution \leftarrow P.solution$;
    a ← Agent constrained using q;
    Use low-level to find path for a;
    if *Path Found* **then**
      Update A.solution;
      A.cost = Sum of path costs for all agents;
    end
    else
      continue
    end
  end
end
return No Solution Found;

completeness and optimality are guaranteed if A* stores multiple states of nodes, defined by their K-costs.

Proof: We consider the h-heuristic to be consistent. First, we prove that if A* is suboptimal, then it does not store multiple states of nodes. Let v_n & v_0 be the goal and start nodes, respectively. Let the cost of the optimal path, that is, $v_0 v_1 v_2 ... v_{n-1} v_n$ be C^*. A* returns a path of cost C such that $C^* < C$. Let the sub-path to each node within the optimal path also be optimal. However, it is proven that A* finds optimal paths to nodes that have optimal sub-paths. Therefore, the optimal path must have nodes with suboptimal sub-paths that remain undiscovered; otherwise, A* will return the optimal path. Let v_{k+1} be a node such that sub-path $v_0 v_1 .. v_{k+1}$ is suboptimal. For a path to v_{k+1} to be discovered from v_k, within the open list, a state node corresponding to this path cost must be added & expanded, even though a state of v_{k+1} with a lower K-cost already exists and will be expanded before this one. The suboptimal path to v_{k+1} remains undiscovered, implying that multiple states with different K-costs are not deployed. ■

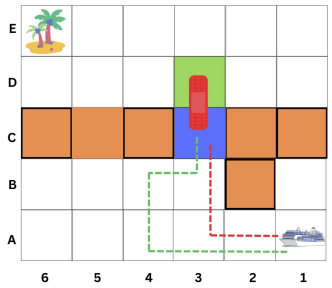


Fig. 4. The figure to aid the proof of Lemma 1. The red line shows the optimal path to C3, having cost t. There is an edge constraint inhibiting the vessel from transitioning from C3 to D3 during timestep t to t+1. Therefore, the optimal path to D3 has a sub-path to C3, which is sub-optimal since waiting is not an option.

Now, we prove completeness. Since the sub-optimal trajectories of varying costs to the same node have different K-costs and the number of nodes in the grid is finite, there can only be a finite number of sub-trajectories with a particular K-cost. Since the expansion of nodes occurs monotonically with respect to K-cost values, it will only take a finite number of expansions to reach a state with a particular K-cost. ■

Theorem 1: COLREGs-compliant CBS produces the optimal solutions.

Proof: Since CBS first explores all nodes of cost t before exploring any node of cost t + 1. Therefore, CBS will always return an optimal solution if the algorithm can return all possible solutions of a particular cost. We have already shown that our low-level planner is complete and optimal with edge and vertex constraints. To make CBS comply with COLREGs, we deployed vertex, edge, and turning constraints. For an agent a that is under the turning constraint that $v_1^{t-1} v_2^t v_3^{t+1} \notin T_a$, where T_a is the trajectory of a, we create two high level nodes. The first high-level node permits all solutions for a that contain $v_1^{t-1} v_2^t$ while the second one allows all those solutions that contain $v_2^t v_3^{t+1}$. Therefore, permitting all possibilities with the constraint of avoiding the sub-trajectory $v_1^{t-1} v_2^t v_3^{t+1}$. ■

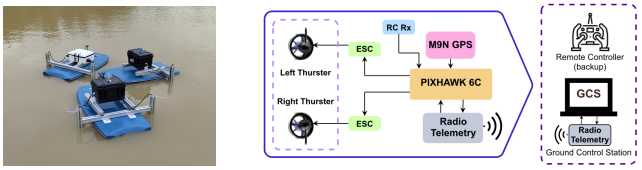
Theorem 2: COLREGs-compliant CBS gives a guarantee of completeness.

Proof: The number of constraints that can be deployed in a solution of cost t is finite, since the number of nodes is finite, and the time at which the edge or vertex constraints are deployed is less than t. Hence, there can only be a finite number of solutions with a cost t. ■

VI. Experiments

A. Empirical Analysis

We created sets of 100 problem instances for each case of 2 to 12 agents on a grid of 72 hexagonal cells. All the experiments were done on a single core of an Apple M1 processor. We plot the average run time for each case 6(a). We also sum the cost of the straight-line path, between start and goal, for each agent in an MAPF problem and compare it with the cost of the returned solution to analyze the variation between the two. We use two low-level solvers, that is, A* [9] and Partial Expansion A* [16], within our algorithm 1.



(a) (b)

Fig. 5. (a) Three agent ASVs (b) System Architecture of ASVs

B. Field Experiments

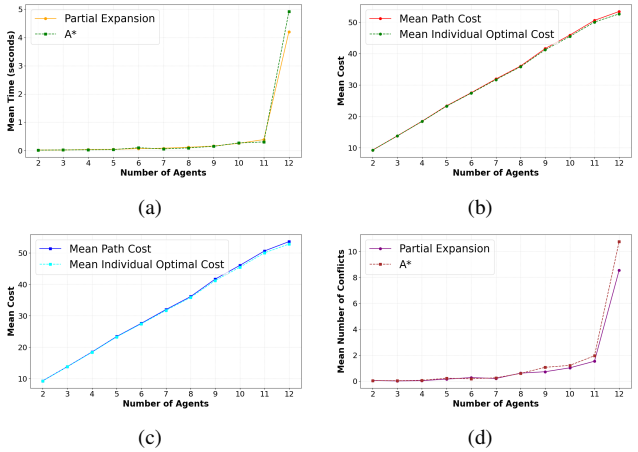
1) Experimental Setup: We demonstrate the proposed COLREGs-compliant CBS algorithm with two and three ASV systems for different COLREGs-compliant collision conditions. The search space is of 25 × 25 $[m^2]$ of open water, which is discretised into 52 hexagonal cells. Each cell has an edge length of 2.31 $[m]$. Our custom-built ASVs are skid-steering surface vessel operating at speed of 0.5 $[m/s]$ and a maximum steering rate of 30 $[deg/s]$. The minimum turning radius is 0.9 $[m]$ (shown in Figure 5(a)). The vehicle dimension is 1.1 $[m]$ × 0.85 $[m]$ × 0.65 $[m]$ and has a Pixhawk 6C autopilot, Blue Robotics T200 thrusters, M9N GPS, SiK 433 MHz telemetry, FS-iA6B RC receiver, and a 14.8 V Li-Ion battery. The system architecture is given in Figure 5(b). The autopilot firmware is based on ArduPilot surface boat firmware.

2) Experiment Workflow: In our experimental setup, the proposed algorithm is executed on an off-board laptop serving as the centralized ground control station (GCS). The GCS communicates with the ASVs via telemetry radio using the MAVLink communication protocol, enabling the GCS to send mission commands to individual agents and receive real-time sensor data. This interaction is managed through a ROS framework [15]. The centralized GCS runs a COLREGs-compliant CBS algorithm to generate conflict-free trajectories for all ASVs. These paths are first planned in Cartesian coordinates, with units defined in meters to match the actual field scale. By selecting a reference origin GPS coordinate in the field, the planned trajectories are then converted to GPS coordinates and they are transmitted to each ASV via the telemetry link. Each ASV autonomously follows the assigned path by sequentially navigating through the designated waypoints to perform COLREGs-compliant collision avoidance.

VII. RESULTS

A. Simulation Results

We observe a surge in all the quantities as the number of agents rises from 2 to 12. Note that COLREGs-compliant CBS gives solutions with path costs identical to the sum of straight-line path costs, from start to goal, for all cases, Figure 6(b). This trend is due to the exhaustive search of the solution space of CBS for a solution with cost K. After all the solutions with cost K are explored, the algorithm exhaustively searches for solutions with cost K + 1. The sudden spike in the Mean time coupled with the rise in the number of conflicts, as the number of agents increases from



(a) (b) (c) (d)

Fig. 6. (a) Mean time vs number of agents. (b) Mean cost vs number of agents for partial expansion as a low-level planner. (c) Mean cost vs number of agents for A* as a low-level planner. (d) Number of COLREG-related conflicts.

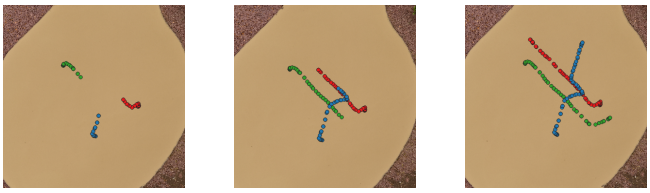


Fig. 7. Trajectory snapshots of the Head-on scenario of three ASVs

11 to 12 are observed due to the limited space of the MAPF environment; therefore, as the number of agents increases, the probability that the paths of two agents are coupled, that is, portions of their trajectories overlap or cross each other, also increases. This creates a higher number of conflicts due to the exhaustive and best-first search property of CBS. The mean time for the Partial Expansion A* is slightly less than that of A* for the case of 12 agents, indicating partial expansion saves time by avoiding the surplus nodes that standard A* would store in the open list.

B. Field Experiment Results

We validated the proposed algorithm through field experiments involving two and three ASVs operating in open water. The GPS trajectories of each ASV were logged and analyzed.

The Figures 7, 8, 9, and 10 illustrate the trajectories recorded at three different steps as the ASVs encounter each other during the field experiments, demonstrating COLREGs-compliant collision avoidance. The trajectories

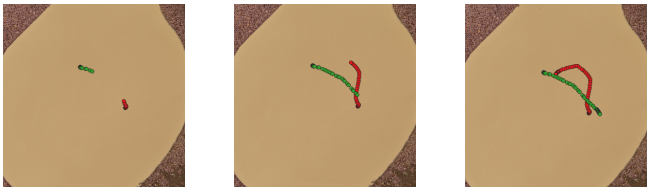


Fig. 8. Trajectory snapshots of the Head-on scenario of two ASVs

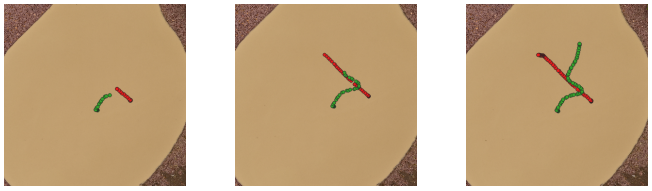


Fig. 9. Trajectory snapshots of the Crossing scenario of two ASVs

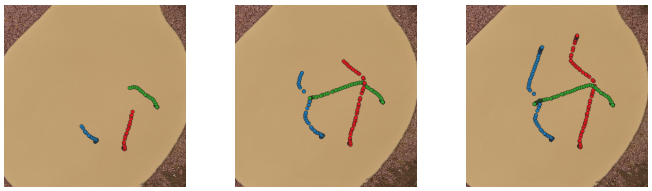


Fig. 10. Trajectory snapshots of the Crossing scenario of three ASVs

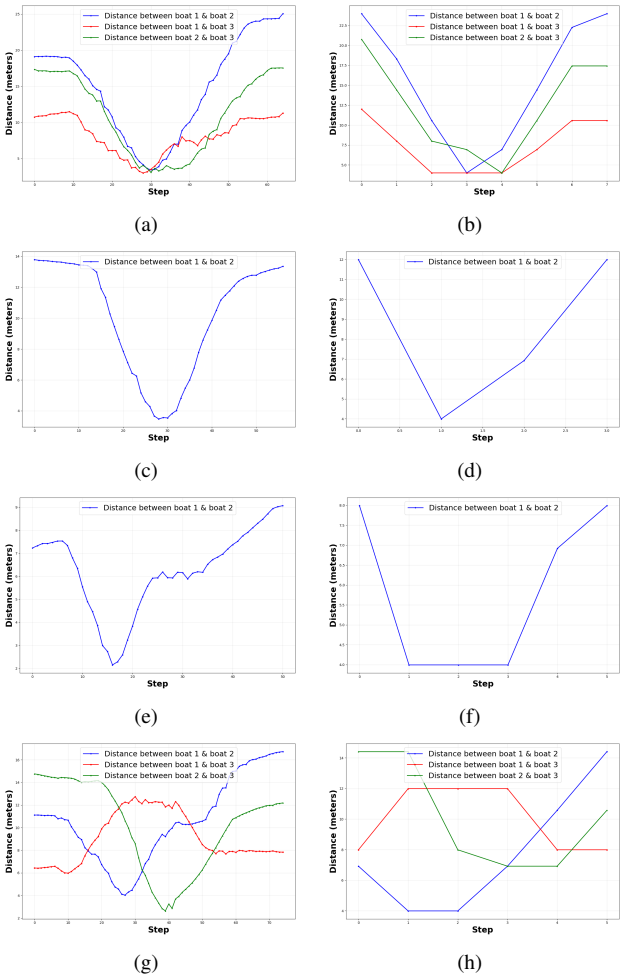


Fig. 11. Distances between any ASVs at each step during the field experiments (left) and simulations (right). (a–b) show the three-ASV head-on collision scenarios, (c–d) depict the two-ASV head-on scenarios, (e–f) show the two-ASV crossing scenarios, and (g–h) present the three-ASV crossing scenarios.

were generated using partial expansion as a low-level planner. The performance of the proposed algorithm in real-world conditions was evaluated by comparing the pairwise distances between ASVs at each step (Figure 11) and by comparing the planned trajectories with those recorded during field deployment. To ensure generality, the problem instances were designed to be asymmetrical. Figures 11(a), 11(c), 11(e), and 11(g) show the observed distances between ASVs for the scenarios depicted in Figures 7–10. For reference, Figures 11(b), 11(d), 11(f), and 11(h) present the corresponding distances as predicted by the simulation.

The plots in Figure 11 show the minimum distance between any two ASVs. The minimum separation is higher, denoting that the collision avoidance was successful, and the experimental results closely match the simulated paths generated by the algorithm. The mean minimum simulated separation between any two ASVs was found to be 4.0 $[m]$ for all conflict-free generated paths, indicating that the COLREGs-compliant CBS algorithm guarantees a minimum safe separation of 4.0 $[m]$ under ideal conditions. In the actual field experiments, the mean minimum separation was measured at 3.48 $[m]$. This slight reduction can be attributed to environmental factors such as wind, waves, GPS measurement inaccuracies and waypoint overshoot by the ASV.

VIII. Conclusion and Future Works

This paper demonstrates a novel search-based collision avoidance strategy with COLREGs compliance for autonomous surface vehicles. We extended the traditional CBS approach by treating COLREG violations as additional conflicts, providing theoretical guarantees of completeness and optimality. Validation via simulation and field experiments confirms its practical viability, demonstrating successful, COLREGs-compliant collision avoidance in multi-ASV scenarios. Unlike existing reactive methods that often fail in complex multi-vessel encounters or learning-based approaches that may lack generalization capabilities, our search-based method provides systematic conflict resolution with guarantees. Currently, our search-based algorithm searches for waypoints on a grid, and we illustrate the execution of the algorithm through actual ASVs. We plan to include the kinematic constraints of ASVs within the problem formulation of search-based path planning. Using decentralized and online approaches for path planning is another direction towards which this work can be extended.

REFERENCES

[1] Y. Wu, T. Wang, S. Liu. A Review of Path Planning Methods for Marine Autonomous Surface Vehicles. Journal of Marine Science and Engineering. 2024; 12(5):833.

[2] A. Vagale, R. Bye, R. Oucheikh, O. Osen, T. Fossen. Path planning and collision avoidance for autonomous surface vehicles II: a comparative study of algorithms. J Mar Sci Technol 26, 1307–1323 (2021).

[3] H. Lyu, Y. Yin. COLREGS-Constrained Real-time Path Planning for Autonomous Ships Using Modified Artificial Potential Fields. Journal of Navigation. 2019;72(3):588-608.

[4] T. Johansen, T. Perez, A. Cristofaro. Ship Collision Avoidance and COLREGS Compliance Using Simulation-Based Control Behavior Selection With Predictive Hazard Assessment, in IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 12, pp. 3407-3422, Dec. 2016.

[5] G. Wei, W. Kuo (2022). COLREGs-Compliant Multi-Ship Collision Avoidance Based on Multi-Agent Reinforcement Learning Technique. Journal of Marine Science and Engineering, 10(10), 1431.

[6] G. Sharon, R. Stern, A. Felner, N. Sturtevant, Conflict-based search for optimal multi-agent pathfinding, Artificial Intelligence, Volume 219, 2015, Pages 40-66.

[7] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, E. Boyarski. (2021). Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. Proceedings of the International Symposium on Combinatorial Search.

[8] L. Wang, K. Sun, F. Tang, Y. Song, C. Wang, X. Feng, F. Zhou, Z. Guo, and Z. Wang. 2025. Search multi-unmanned ship path planning algorithm. In Proceedings of the 5th International Conference on Artificial Intelligence and Computer Engineering (ICAICE '24). Association for Computing Machinery, New York, NY, USA, 230–234.

[9] P. Hart, N. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, July 1968.

[10] L. Hu, H. Hu, W. Naeem, Z. Wang, A review on COLREGs-compliant navigation of autonomous surface vehicles: From traditional to learning-based approaches, Journal of Automation and Intelligence, Volume 1, Issue 1, 2022, 100003.

[11] E. Ebeid, M. Skriver, K. Terkildsen, K. Jensen, U. Schultz, A survey of Open-Source UAV flight controllers and flight simulators, Microprocessors and Microsystems, Volume 61, 2018, Pages 11-20.

[12] Pixhawk Project. Pixhawk.org: The open standards for drone hardware. Available at: `https://pixhawk.org/`

[13] ArduPilot Project. ArduPilot: A trusted, versatile, open source autopilot system. Available at: `http://ardupilot.org/`

[14] MAVROS. MAVROS – MAVLink extendable communication node for ROS. Available at: `https://wiki.ros.org/mavros`

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Ng. "ROS: an open-source Robot Operating System." In ICRA workshop on open source software, vol. 3, no. 3.2, p. 5. 2009.

[16] A. Felner, M. Goldenberg, G. Sharon, R. Stern, T. Beja, N. Sturtevant, J. Schaeffer, R. Holte, (2021). Partial-Expansion A* with Selective Node Generation. Proceedings of the AAAI Conference on Artificial Intelligence, 26(1), 471-477.