COLREG-RRT: A RRT-based COLREGS-Compliant Motion Planner for Surface Vehicle Navigation

Hao-Tien (Lewis) Chiang and Lydia Tapia

***Abstract*— Recent reports indicate that roughly half of maritime casualties are of a navigational nature, and over 80% of maritime collisions are due to human error. Autonomous navigation offers promise toward reducing maritime collisions. However, motion planning for Autonomous Surface Vehicles (ASVs) is challenging since surface vessels are nonlinear under-actuated kinodynamic systems with large inertia. Thus, ASV planners must identify long-term trajectories in order to avoid guiding the ASV into inevitable collision states. Furthermore, maritime vessels are required to follow COLlision REGulationS (COLREGS), which dictates collision avoidance patterns. Consequently, the motion of any nearby surface vessel can affect the motion of other vessels, including the ASV. Current state of the art methods are based on Model Predictive Control (MPC) and assume other vessels move at constant velocities.**

In this paper, we propose COLREG-RRT, a RRT-based planner capable of identifying long-term, COLREGS-compliant trajectories with a high navigation success rate. This is achieved by conducting joint forward simulations of both the ASV and the other vessels during RRT growth in order to anticipate future collisions. The COLREGS-compliance is enforced by constructing virtual obstacles that inhibit tree growth. We demonstrate the COLREGS-compliance in single-ship encounters and compare against two state of the art methods in multi-ship encounters with up to 20 other vessels. Experiments indicate that COLREG-RRT has a 32% higher success rate and is real-time capable in the most difficult environment tested. Additionally, COLREG-RRT identifies longer trajectories, as compared to MPC. This property aids with collision avoidance with other ships.

I. Introduction

A recent report indicates that roughly half of the maritime casualties during 2011-2015 were of a navigational nature, such as contacts, groundings/strandings or collisions [1]. Since over 80% of the maritime collision accidents are caused by human decision failure [2], autonomous navigation for surface vessels may significantly reduce collision accidents. Unfortunately, autonomous navigation is very challenging as surface vessels are nonlinear under-actuated kinodynamic systems with large inertia. Additionally, maritime vessels are required to follow COLlision REGulationS (COLREGS) agreed by International Maritime Organization, which gives general guidelines regarding the collision avoidance patterns and responsibilities [3]. The action of the ship guided by autonomous navigation, an Autonomous Surface Vehicle (ASV), can affect other obstacle ships, nearby ships the ASV has no control over, thus forming a complex system. These constraints make planning difficult since long-term deliberative planning and joint forward simulation of both the ASV and obstacle ships is required.

Current state of the art COLREGS-compliant autonomous navigation includes the use of Model Predictive Control (MPC) [4] and dynamic window-based methods [5]. These methods forward simulate the motion of the ASV in order to generate dynamically feasible trajectories that avoid collisions and comply with COLREGs. However, obstacle ships are assumed to have a constant or random velocity [4], [5]. This assumption does not consider that obstacle ships are also required to follow COLREGS and react to the ASV. Other COLREGS-compliant planning methods include Artificial Potential Field (APF)-based [6], [2] and Velocity Obstacle (VO)-based methods [7]. These are fast reactive methods that avoid collision by constructing repulsive fields or velocity obstacles around obstacle ships. COLREGS-compliance is enforced by combining a line-of-sight-based algorithm with APF or by integrating into velocity obstacles [7]. However, since ASVs typically have a large inertia, it is often impossible for the ASV to achieve the desired course in time to avoid collisions.

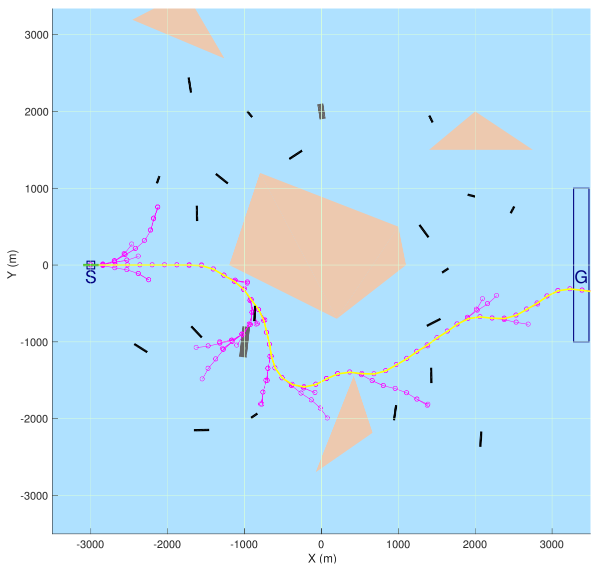


Fig. 1: Example of COLREG-RRT navigating the ASV (green rectangle) from start (S) to the goal region (blue box G). The RRT (magenta) and the selected trajectory (yellow) are also shown. The ASV must avoid collision with static obstacles, islands (sand colored polygons) and low-observable obstacles (gray rectangles), and dynamic obstacles, other ships (gray rectangles). Both the ASV and the other ships must select actions that are COLREGS-compliant.

In order to address these limitations, we propose a COLREGS-compliant RRT-based motion planner for ASV navigation, COLREG-RRT. The planner works in state-time space in order to identify collision-free and COLREGS-compliant trajectories in dynamic environments [8]. This means the planner requires knowledge of future positions of the obstacle ships. Therefore during tree-growth, our method utilizes joint forward simulations of the ASV and obstacle ships and constructions of virtual obstacles to generate long-term, high-safety, COLREGS-compliant trajectories. The joint forward simulation can utilize many existing COLREGS-compliant navigation algorithms in order to simulate the motion of obstacle ships. Due to real-time constraints and limited sensor range, the trajectory returned by COLREG-RRT may not always reach the goal. Therefore, as the robot executes the path, it is re-checked in order to enhance safety. Additionally, to reduce the time to reach the goal, attempts are made during path execution to grow a tree directly towards the goal.

The novelty of COLREG-RRT lies in the: 1) integration of joint forward simulations of the ASV and obstacle ships during RRT growth in order to anticipate collisions, and 2) use of virtual obstacles designed to enforce COLREGS-compliance for RRT-based methods. We demonstrate that these additions are efficient and provide a higher COLREGS-compliance rate than comparison methods.

To evaluate COLREG-RRT, we demonstrate COLREGS-compliance in single-ship encounters and compared against two state of the art methods in multi-ship encounters with up to 20 obstacle ships. The comparison methods include an APF-based method [2] and a MPC-based method [4]. The enclosed video demonstrates COLREG-RRT navigating an ASV in a multi-ship encounter environment with shorelines, low-observable obstacles and 20 obstacle ships.

II. Related Work

Previous COLREGS-compliant motion planners for ASV navigation include APF-based and MPC-based methods. The concept of APF for obstacle avoidance was proposed in [9] for manipulators and mobile robots. This idea has been extended to comply with COLREGS by incorporating more than 200 fuzzy logic rules designed by experts [6]. Another approach, which aims to create fast, accurate simulation of ship motion, combines APF and line-of-sight navigation [2]. MPC-based methods are also used for ASV planning. Due to the under-actuated nonlinear kinodynamic nature of surface vessels, state of the art MPC-based methods discretize the course deviation, i.e., the difference between the current and desired course. These course deviations are then used to generate finite horizon trajectories using a low-level controller and forward simulations of the ASV [4], [10], [5]. Various heuristic cost functions are assigned to each trajectory based on risk of collision, goal progression, and COLREGS-compliance [4], [10], [5]. The trajectory with the minimum cost is selected as the motion plan.

RRT was developed to navigate kinodynamic robots among obstacles [11]. However, two factors are critical for RRTs to be “rapidly-exploring” [12], [13]. The first factor is the availability of an optimal/near-optimal steering function to navigate the robot between two points in the state space. This is challenging for kinodynamic systems and is partially addressed by an optimal steering function for kinodynamic robots with controllable linear dynamics [12]. The second factor is the availability of a proper distance metric between two points in state space. This is also partially addressed by numerically computing the reachability of the robot offline [14] or approximating the reachable states via visibility graphs [12]. Lastly, RRT has been used to generate global guidance paths for a dynamic window-based planner for ASVs [5]. These global guidance paths are typically generated for maritime navigation by A* and its variants [15].

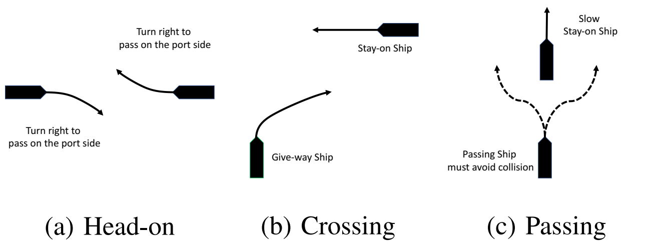


Fig. 2: Encounters defined by COLREGS and the corresponding evasion patterns and responsibilities.

Obstacles that interact with each other and with the robot pose a significant challenge for motion planning. This is because the motion of obstacles is affected by the action of the robot and other obstacles, yet, collision-free robot motion plans can only be generated given the future positions of obstacles. Since the robot has no direct control over the obstacles, this problem is different from multi-agent motion planning where all agents are controllable via centralized planning algorithms such as multi-agent RRT* [16]. Monte Carlo simulations have been previously used in order to predict the motion of strongly-interacting obstacles with stochastic dynamics [17]. However, the simulations do not consider the interaction between the robot and obstacles. A MPC that optimizes the cost over both the robot and human drivers (moving obstacles) was proposed in [18]. However, the high computation cost of this MPC makes it difficult to scale up to the many obstacle case. Lastly, a Gaussian process was developed to learn the interaction between obstacles in order to predict obstacles' trajectories [19]. By modeling the robot as another obstacle, these predicted trajectories can be used to generate control actions. However, since the Gaussian process does not consider dynamics constraints and surface vessels often have large inertia, the generated trajectories may not be executable by an ASV.

III. Preliminaries

We explain the ASV dynamics model and the basics of RRT in this section. A summary of encounters as defined by COLREGS is shown in Figure 2.

A. Ship Dynamics

In this paper we model the dynamics of all surface vessels by a standard 3 DOF ship model as described in [20]. This model describes a vessel's state as $x = [\boldsymbol{\eta}, \boldsymbol{v}]^T$, where $\boldsymbol{\eta} = [X, Y, \psi]^T$, and $\boldsymbol{v} = [u, v, r]^T$. X, Y, ψ are the coordinates and course of the vessel and u, v, r correspond to surge, sway and yaw rate, respectively. Figure 3 shows the definition of surge, sway and course of a surface vessel. The control inputs

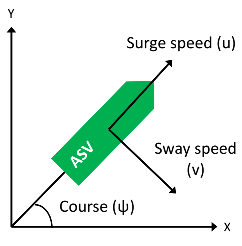


Fig. 3: Definition of surge, sway, and course of the ASV.

$\mathcal{U}$ of the ship are the forward thrust, F_{thrust}, and lateral force, $F_{lateral}$. The forward thrust and lateral force can be computed by propeller rpm and rudder angle, respectively [20]. The equations of motion are nonlinear,

$$\dot{\boldsymbol{\eta}} = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{v} \quad (1)$$

$$\boldsymbol{M}\dot{\boldsymbol{v}} + \begin{bmatrix} 0 & 0 & v \\ 0 & m & -u \\ -v & u & 0 \end{bmatrix} \boldsymbol{v} + \begin{bmatrix} d_{2u}u|u| + d_{1u}u \\ d_{2v}v|v| + d_{1v}v \\ d_{1r}r \end{bmatrix} = \tau \quad (2)$$

and

$$\boldsymbol{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_Z \end{bmatrix}, \tau = \begin{bmatrix} F_{thrust} \\ F_{lateral} \\ l_r F_{lateral} \end{bmatrix} \quad (3)$$

where m, I_Z, and l_r are the mass, moment of inertia on the z-axis and the point of attack of the rudder force of the vessel, respectively. The drag coefficients are $d_{2u}, d_{1u}, d_{2v}, d_{1v}$, and d_{1r}. This model is chosen since it is simple yet it describes ship motion in calm water well [5]. It is therefore widely adopted by ASV navigation research [21], [2], [4].

Similar to [5], A low-level PD controller was employed in order to generate control inputs $\mathcal{U}$ to steer toward the desired course ψ_{sp},

$$F_{thrust} = F_{max} \quad (4)$$

$$F_{lateral} = \frac{I_Z}{l_r}(K_p(\psi_{sp} - \psi) - K_d r) \quad (5)$$

where K_p = 5 and K_d = 1 are tunable control gains.

B. Basics of RRT

We briefly explain how the basic version RRT works. First, the root node of the tree is created and it stores the the robot's start state $x^{start} \equiv [\boldsymbol{\eta}^{start}, \boldsymbol{v}^{start}]^T$. Next, the algorithm repeats the following steps for a fixed number of iterations. A state x_{rand} is randomly sampled from the state space. If x_{rand} is not in collision with obstacles, the node ($n_{selected}$) in the tree with the state $x_{nearest}$ that is the closest to x_{rand} is selected for extension according to some distance metric D. A control action u is then computed to steer the robot from $x_{nearest}$ to x_{rand}. Next, a forward simulation of the robot that starts from $x_{nearest}$ and executes u for Δ_{tree} seconds is conducted. If the resulting new state x_{new} is not in collision, a new node of the tree that stores x_{new}, the control input u, and a pointer to the parent node $n_{selected}$ is created. If any node reaches the goal, a path, namely, a sequence of control inputs can be extracted. A goal bias, i.e., choosing the goal state as x_{rand} with probability $P_{goalBias}$ is often used in order to speed up tree growth.

IV. Method

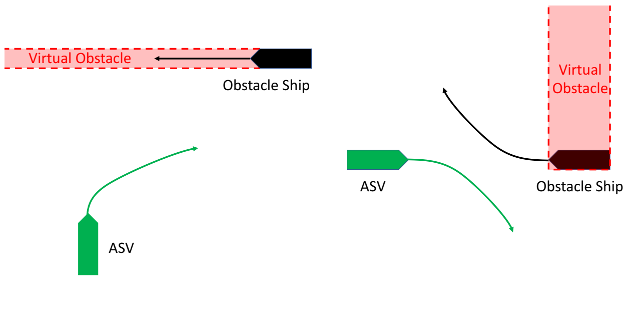
One consequence of COLREGS is the motion of obstacle ships is affected by the action of the ASV, yet COLREGS-compliant collision-free plans can only be generated given the future positions of obstacle ships. COLREG-RRT breaks this deadlock by storing the joint state of the ASV and the obstacle ships in each RRT node in order to conduct joint forward simulations during tree growth. The joint state is the Cartesian product of the state of the ASV, x_r, and the states of N obstacle ships, $x_1, x_2, ..., x_N$. The joint forward simulations are general in the underlying COLREGS-compliant navigation algorithms of obstacle ships. To comply with COLREGS, virtual obstacles are created around obstacle ships. As the ASV executes the path, the tree continues to expand and the path is checked for collision with suddenly appearing obstacles such as submerged rocks.

A. Joint Forward Simulator

In this paper, joint forward simulations are conducted by a ship motion simulator described in [2]. A brief description of how it works is given below. To avoid grounding, i.e., collision with known static obstacles such as shorelines, the simulator computes the gradient of the sum of attractive and repulsive potentials for each obstacle ship. The gradient is then used as the desired course for that ship. To avoid other obstacle ships, the simulator first determines the encounter types as defined by COLREGS for each ship. If the encounter type demands evasive action, i.e., Head-on and Crossing Give-way types, the desired course of that ship is changed to the course needed to avoid other obstacle ships. This avoidance course is computed geometrically by assuming other ships travel in constant velocities. The simulator repeats these steps at each time step. For numerical stability reasons, the simulator has a small simulation time step Δ_{world} = 0.1s. We chose this simulator since it has been demonstrated to produce COLREGS-compliant motion plans for up to 4 vessels. Other methods, such as VO-based methods [7], can also be used as the simulator.

B. Virtual Obstacles for COLREGS-Compliance

Instead of assigning penalty costs to non-COLREGS-compliant trajectories like MPC-based methods, we choose a different approach designed specifically to work with RRT. When an obstacle ship is within range d_{colreg}, of the ASV, a virtual obstacle is created according to the encounter type dictated by COLREGS. Figure 4 shows these virtual obstacles in the Head-on and the Crossing Give-way encounters. Similar to other ASV planners, COLREG-RRT only considers the Head-on (Rule 14) and Crossing Give-way (Rule 15) rules, since only these two encounters require the ASV to move in a restricted manner, i.e., turn right (starboard) [7], [4].



(a) Crossing Give-way (b) Head-on

Fig. 4: Virtual obstacles for COLREGS encounters.

For RRT-based planners, the virtual obstacles have three benefits over the penalty cost approach. 1) RRT-based planners require an optimal distance metric and steering function in order to identify asymptotically optimal paths. Since no computationally inexpensive method exists for either the distance metric nor the steering function, the identified paths are sub-optimal and therefore may not be COLREGS-compliant. 2) Ideally, ASV should always choose COLREGS-compliant paths unless it leads to collisions. Therefore, RRT-based planners should focus on identifying COLREGS-compliant paths first by avoiding the virtual obstacles. 3) The virtual obstacles do not have a penalty cost parameter that may be difficult to tune.

C. *COLREG-RRT*

In order to modify the basic RRT method to consider both the kinodynamic constraints of the ASV and COLREGS-compliance while navigating with dynamic obstacles, we have decomposed RRT construction into three phases. In the first phase, a direct route to the goal is checked for the possibility that there are no encounters with obstacles. If this is not possible, the second phase grows a tree while applying virtual obstacles to maintain COLREGS-compliance during tree growth. If a collision-free path of sufficient time-horizon is not able to be found, then tree growth continues in phase three without consideration of virtual obstacles for COLREGS-compliance. The three phases combine to provide COLREGS-compliance when possible and continued navigation even when compliance is not possible. Algorithm 1 provides details of COLREG-RRT that starts with a root node that stores the current joint state of the ASV and obstacle ships (line 4).

Important to all phases is the testing for collisions with obstacles. In the results shown, this is done via joint forward simulations. Since obstacle ships and the ASV interact, each node of the RRT stores the joint state of the ASV and the obstacle ships. An action, u, to steer the ASV towards x_{rand} from the ASV state stored in the selected node ($n_{selected}$.ASVState) is computed using the low-level PD controller described by Eq. 5. Using u, the simulator returns a new joint state, x_{new}, by simulating the motion of all ships for the duration of the tree time step Δ_{tree}. At each time step of the simulation ($\Delta_{world} \ll \Delta_{tree}$), the simulated

Algorithm 1 COLREG-RRT

Input: Other ship motion simulator $\mathcal{S}$, ASV current state x_r, current states of other ships $\{x_1, x_2 ... x_N\}$, reGrowTree=true

```
1:  for t=0; t< $T_{max}$; t=t+$\Delta_{world}$ do
2:      if reGrowTree == true then
3:          $\mathcal{T}$.clear(); reGrowTree = false
4:          $\mathcal{T}$.insertRootNode($x_r$, $x_1, x_2, ..., x_N$)
    ▷ /* Phase 1 Tree-Growth */
5:          [$\mathcal{T}$, reachedGoal] = growTree($\mathcal{T}$.rootNode, $\mathcal{T}$, $\mathcal{S}$, goalMode=true, ignoreColreg=false, maxIter)
6:          if reachedGoal == false then
    ▷ /* Phase 2 Tree-Growth */
7:              [$\mathcal{T}$, reachedGoal] = growTree($\mathcal{T}$.rootNode, $\mathcal{T}$, $\mathcal{S}$, goalMode=false, ignoreColreg=false, maxIter)
8:          end if
9:          $\mathcal{P}$ = getPath($\mathcal{T}$);
10:         if $\mathcal{P}$.length() $< \tau$ && reachedGoal == false then
    ▷ /* Phase 3 Tree-Growth */
11:             [$\mathcal{T}$, reachedGoal] = growTree($\mathcal{T}$.rootNode, $\mathcal{T}$, $\mathcal{S}$, goalMode=false, ignoreColreg=true, maxIterNoColreg)
12:         end if
13:         $\mathcal{P}$ = getPath($\mathcal{T}$);
14:     end if
15:     $x_r$ = $x_r$+getAction($\mathcal{P}$, t) $\cdot\Delta_{world}$
16:     if $\mathcal{P}$.remainingTime(t) $< \tau$ && !reachedGoal then
17:         reGrowTree = true
18:     end if
19:     if t mod $T_{growGoal} == 0$ then
20:         [$\mathcal{T}$, reachedGoal] = growTree($\mathcal{T}$.currentNode, $\mathcal{T}$, $\mathcal{S}$, goalMode=true, ignoreColreg=false, maxIter);
21:         $\mathcal{P}$ = getPath($\mathcal{T}$);
22:     end if
23:     if $\mathcal{P}$.isAtANode(t) == true then
24:         reGrowTree = checkFuturePath($\mathcal{P}$, t, $\mathcal{S}$)
25:     end if
26: end for
```

ASV executes u and the motion of obstacle ships is simulated as described in Section IV-A. The new joint state, x_{new}, is checked if the simulated ASV is in collision with obstacles. In phase three of tree growth this includes static obstacles and obstacle ships. In phase one and two of tree growth this also includes virtual obstacles. Collision checking takes place during tree growth (Algorithm 2 lines 9-12).

In the first phase of tree growth, COLREG-RRT attempts to grow a tree directly towards the goal (Algorithm 1, line 5). This step is beneficial since maritime navigation can often involve sparse obstacles in open waters. In this case, a tree grown directly towards the goal returns trajectories with reduced control effort and the time required to reach the goal. This tree can be constructed by calling the GrowTree()

Algorithm 2 GrowTree

Input: Starting node n_{start}, Tree $\mathcal{T}$, Simulator $\mathcal{S}$, goalMode, ignoreColreg, iter

Output: $\mathcal{T}$, reachGoal

```
    $n_{last}$ = $n_{start}$
1:  for i=0; i<iter; i++ do
2:      if goalMode == true then
3:          $x_{rand}$ = getGoal()
4:          $n_{selected}$ = $n_{last}$
5:      else
6:          $x_{rand}$ = goalBiasedSampling($P_{chooseGoal}$)
7:          $n_{selected}$ = getClosestNode($\mathcal{T}$, $x_{rand}$)
8:      end if
9:      $u$ = steer($n_{selected}$.ASVState, $x_{rand}$)
10:     $x_{new}$ = $\mathcal{S}$.evolve($\Delta_{tree}$, $u$, $n_{selected}$.jointState)
11:     colregFailed = !ignoreColreg &&
        inCollisionWithVirtualObs($x_{new}$)
12:     if inCollision($x_{new}$) || colregFailed == true then
13:         if goalMode == true then
14:             return [$\mathcal{T}$, reachGoal=false]
15:         else
16:             Continue
17:         end if
18:     end if
19:     $\mathcal{T}$.insertNode($x_{new}$, $n_{selected}$)
20:     $n_{last}$ = $\mathcal{T}$.lastInsertedNode
21:     if reachedGoal($x_{new}$) == true then
22:         return [$\mathcal{T}$, reachGoal=true]
23:     end if
24: end for
25: return [$\mathcal{T}$, reachGoal=false]
```

function (Algorithm 2) while setting the goalMode flag to true. This forces the GrowTree() function to always select the last inserted node in each iteration for extension and use the goal state as x_{rand} (Algorithm 2, lines 3 and 4). The tree grows directly towards the goal until a collision with obstacles or virtual obstacles is detected (line 12). If a collision is detected, the GrowTree() function in the goal mode returns with the reachedGoal flag set to false (line 5 in Algorithm 1). This triggers the next phase of tree growth.

In the second phase of tree growth, the GrowTree() function grows a RRT and terminates if any node reaches the goal (line 21) or when the max number of iterations is reached (line 25). After the construction of the tree, a path is extracted by selecting the node with the ASV state that is the closest to the goal (line 9 in Algorithm 1). If the extracted path does not reach the goal, it is a partial path, which may lead the ASV to Inevitable Collision States (ICSs) [22]. One way to avoid ICSs is the τ-safety criterion [23]. A path satisfies this criterion if it is at least τ seconds long. Therefore, if the extracted path is shorter than τ seconds, the third phase of tree growth is used.

In the third phase, tree growth is extended without considering COLREGS-compliance by ignoring the virtual obstacles (line 11). After the tree is grown, a path is extracted from the tree again (line 13). COLREGS in multi-vessel encounters is not well-understood [24], [25], so this third phase provides a trade-off between the violation of COLREGS and path safety. We empirically found that COLREG-RRT only ignores COLREGS-compliance when there are at least 2 obstacle ships and shorelines within range d_{colreg}.

Once a path is returned by one of the three phases, the ASV executes the path (line 15). In order to shorten the time required to reach the goal, COLREG-RRT attempts to grow a tree directly towards the goal every $T_{growGoal}$ seconds (line 20-21). Also, full tree growth, potentially including all phases, is performed if the remaining partial path is shorter than τ seconds (line 16-17)

Since low-observable obstacles, such as buoys, fishing nets and submerged rocks, pose significant navigation challenges for ASV since they are only detectable at close range [21]. To avoid these during path execution, future nodes up to τ seconds away are rechecked for collisions using the newly acquired information every time the ASV reaches a node (line 24). The tree is regrown if collisions are identified.

V. Experiments

A. Setup

COLREGS do not precisely define the difference between the Crossing and Head-on encounters. Therefore, in this paper we chose the same criteria as [2] (shown in Table I).

Encounter Types	Criteria
Head-on	$\lvert\psi\rvert \leq 168.75°$
Crossing Give-way	-168.75° $< \psi <$-11.25°
Crossing Stay-on	11.25° $< \psi <$ 168.75°

TABLE I: Criteria used to defined encounter types dictated by COLREGS. ψ is the course difference between the ASV and the obstacle ship, ($\psi \equiv \psi_{ObstacleShip} - \psi_{ASV}$)

The parameters used by all methods are: world simulation time step (Δ_{world} = 0.1s), range which COLREGS are considered (d_{colreg} = 2000m). Parameters exclusive to COLREG-RRT are: tree time step (Δ_{tree} = 20s), number of RRT iterations (maxIter=300), number of RRT iteration while ignoring COLREGS (maxIterNoColreg = 100), τ-safety criterion (τ = 120s), goal bias probability ($P_{goalBias}$ = 0.1), weighted euclidean weights (w_d = 10 and w_a = 1000), grow goal period ($T_{growGoal} = 100s$). For the results shown, we empirically found that a weighted Euclidean distance metric developed for unicycle robots [26] works well for our standard 3 DOF ship with kinodynamic dynamics. The parameters of the ship simulator are given in [2].

The motion of all obstacle ships are determined by an implementation of the ship simulator described in [2]. All methods were implemented in C++ and all experiments were run on a single core of an Intel i7-6820HQ at 2.7GHz with 16GB of RAM. All experiments were repeated 100 times. Uncertainty in success rates due the limited number of experiments is captured using the 99% confidence level derived from the central limit theorem.

Ship Types	Length	Beam	m	I_Z	d_{2u}	d_{1u}	d_{2v}	d_{1v}	d_{1r}	F_{thrust}	$F_{lateral}$	l_r
Small	100	30	3.3K	1.3K	8.25	16.6	330	10K	3.3K	700	29	4
Large/ASV	200	30	30M	1.0M	8.25	16.6	330	10K	50K	60K	10	100

TABLE II: Parameters of ships. All parameters have units measured by standard mks units.

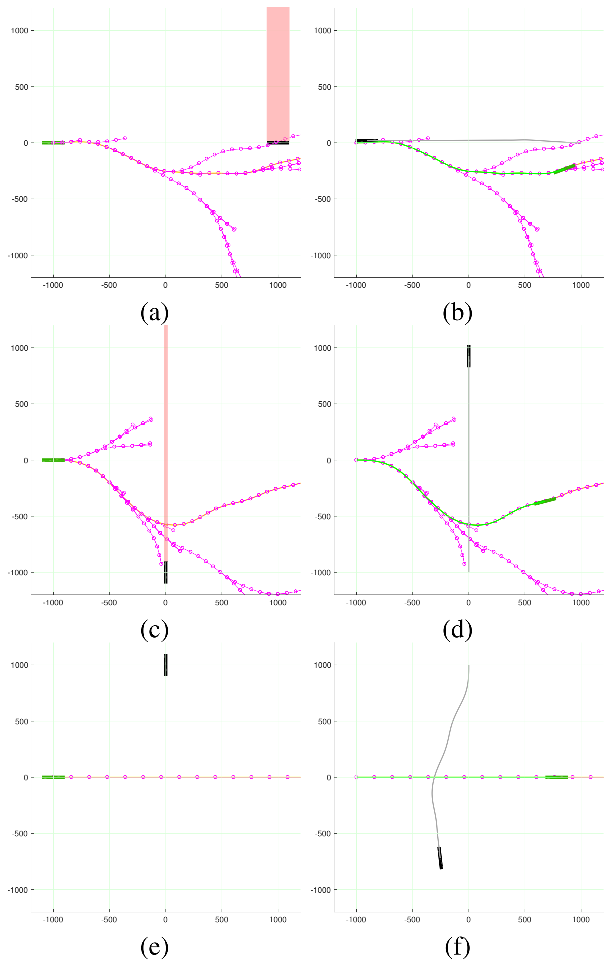


Fig. 5: The single ship Head-on (a,b), Crossing Give-way (c,d) and Crossing Stay-on (e,f) encounters. The left column (a,c,e) shows the ASV (Green rectangle, starts at (-1000m, 0m)), the obstacle ship (black rectangle), the virtual obstacles (red boxes), and the tree (magenta curves) generated by COLREG-RRT. The right column (b,d,f) shows the trajectories of the ASV (green curves) and the obstacle ship (gray curves)

B. Single Ship Encounters

To demonstrate the COLREGS-compliance, we tested COLREG-RRT in the Head-on, Crossing Give-way and Crossing Stay-on encounters with a single obstacle ship. The left column of Figure 5 demonstrates that the use of virtual obstacles to prevent non-COLREGS-compliant trajectories in the tree from being chosen, i.e., passing from starboard in a Head-on encounter (Figure 5a) or turn to port in a Crossing Give-way encounter (Figure 5c). As shown in Figure 5, the virtual obstacles inhibit the tree-growth, thus preventing the non-COLREGS-compliant trajectories from being generated. The resulting trajectories taken by COLREG-RRT are shown in Figure 5 b,d as green curves. In addition, Figures 5e,f show that in a Crossing Stay-on encounter, COLREG-RRT correctly anticipates that the obstacle ship will move in a COLREGS-compliant manner. Therefore, a stay-on course is generated.

C. Multi-Ship Encounters

To compare COLREG-RRT with other state of the art methods for ASV navigation, we tested in the environment shown in Figure 1. This environment has four islands and two low-observable obstacles (gray rectangles) whose existence and location are observable within 1000m. The ASV starts at (-3000m, 0m) and must navigate to the goal region without collision within T_{max} = 3000s. There are also up to 20 obstacle ships composed of both small and large ships. The parameters used for all ships are described in Table II. The starting locations of obstacle ships are chosen randomly from a 5000m wide square centered around the origin. These locations are restricted to be at least 500m away from any obstacle. The initial courses of obstacle ships are randomly chosen, and the goal of each obstacle ship is 10000m ahead. The comparison methods include a state of the art MPC-based method (MPC) [4] and an APF-based algorithm (APF) [2]. The parameters used for the MPC-based comparison method are: time horizon (T = 200s), temporal resolution (Δ_{MPC} = 10s), Collision cost penalty (K^{coll} = 1000), COLREGS violation penalty (κ = 10000), target course deviation cost (Δ_χ = 1).

Figure 6a indicates that the success rate of COLREG-RRT is much higher than other methods, e.g., 35% higher in the most difficult 20 obstacle ships environment. In addition, unlike MPC and APF, the success rate of COLREG-RRT is not heavily impacted by the increasing number of obstacle ships. This is expected as the COLREGS-compliant joint forward simulator is more accurate in anticipating the motion of obstacle ships compared to the linear extrapolation used by MPC.

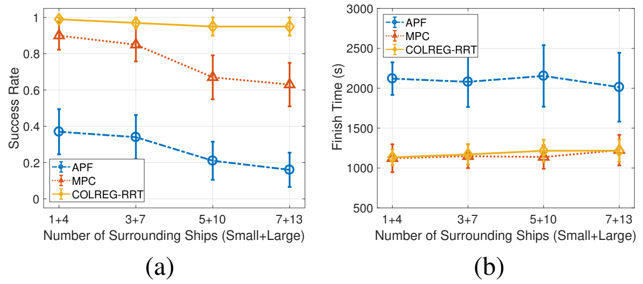


Fig. 6: Success rate (a) and finish time of successful runs (b).

We also explore the cause of collision for each method by observing the failure cases in the most crowded environment with 20 obstacle ships. We found that all 5 collision events of COLREG-RRT occurred between two nodes that were not

# of obstacle ships	COLREG-RRT	MPC	APF
1 Small + 4 Large	1.69±/0.65 max: 3,407.36	0.45±0.07 max: 739.84	0.02±0.004 max: 15.33
3 Small + 7 Large	3.67±2.57 max: 11,230.82	0.66±0.11 max: 856.68	0.02± 0.004 max: 21.54
5 Small + 10 Large	6.19±4.51 max: 15,310.11	0.86±0.13 max: 651.39	0.02± 0.004 max: 24.32
7 Small + 13 Large	7.99±4.67 max: 18,940.89	0.90±0.21 max: 945.52	0.02±0.004 max: 27.97

TABLE III: Computation time per planning step in ms. Max indicates the maximum planning time per planning step among all planning steps in all 100 runs.

in collision with any obstacle. This suggests that reducing the tree step size (Δ_{tree}), thus increasing the collision check frequency, may improve success rates. Among the 37 collision events of MPC, 8 of them also occurred between two nodes that were not in collision with any obstacle. The remaining 29 events were due to the inaccurate linear extrapolation prediction of obstacle ships, as all these events involved at least one ship that changed course within 30s of the collision event. Lastly for APF, 5 out of 84 collision events were due to the ASV being stuck in a potential field local minimum, and the remaining 79 were due to the ASV's inability to follow the desired avoidance course in time.

Figure 6b shows the finish time of various methods. The finish time is defined as the amount of time for the ASV to reach the goal region. Figure 6b indicates that the finish time of COLREG-RRT and MPC are comparable and are both much lower than that of the APF. The APF method suffers from local minima formed by obstacles and therefore has much higher finish times. The fact that COLREG-RRT achieves similar finish time with MPC is interesting since our RRT is sub-optimal. MPC identifies the trajectory that minimizes the cost function every time step therefore should achieve better finish times. Two factors may explain this result. 1) The cost function used by MPC is determined heuristically by considering collision, COLREGS-compliance, as well as finish time. As a result, trajectories identified may not be optimal in finish time alone. On the other hand, COLREG-RRT uses virtual obstacles to enforce COLREGS-compliance unless such the identified path likely leads to collisions. This allows COLREG-RRT to separate COLREGS-compliance and identifying low finish time trajectories. 2) COLREG-RRT attempts to expand the tree in a straight-line toward the goal every time the ASV reaches a node. This greatly reduces the finish time when there is no risk of collision or COLREGS violation.

Table III shows the average, standard deviation and the max of computation time per planning step of the various methods. It indicates that the computation time per planning step of COLREG-RRT is much larger than MPC or APF. This is due to the joint forward simulations conducted during tree growth which can be time-consuming. However, since COLREG-RRT plans an action every Δ_{tree}=20s, it is still real-time capable even in the most crowded environment tested with 20 obstacle ships.

Since COLREGS-compliance is paramount in surface vessel collision avoidance, we further investigate the COLREGS-compliance of COLREG-RRT and comparison methods in the environment with 20 obstacle ships. Since multi-ship COLREGS-compliance is difficult to define and often subjective [24], we quantify the COLREGS-compliance of a trajectory by an indicator function proposed in [4]. This indicator function determines the ASV violates COLREGS if an obstacle ship within range d_{colreg} is on the starboard side of the ASV in a Head-on encounter or an obstacle ship within range d_{colreg} is on the starboard side of the ASV and is not being overtaken by the ASV in a Crossing Give-way encounter. After the each run, we compute this indicator function along the trajectory of the ASV in 0.1s intervals. We found that COLREG-RRT violates COLREGS 4.47% of the time while APF violates 7.26% of the time. Most interestingly, MPC violates 7.78% of the time despite the fact that it uses this indicator function to penalize non-COLREGS-compliant trajectories. This indicates that virtual obstacles are an effective way of enforcing COLREGS-compliance.

Since joint forward simulations are computationally expensive, real-time ASV planners should minimize the amount of simulations used. Table IV shows the trajectory temporal length and the number of joint forward simulation calls. The trajectory temporal length is defined as the amount of time of the returned plan. For COLREG-RRT it is the number of nodes in the path times the tree time step (Δ_{tree}). The values shown are measured at the beginning of each run since the length of the path returned varies based on the distance to the goal. Since the number of iterations per stage of COLREG-RRT is limited, and tree growth terminates upon reaching the goal, the number of joint forward simulation calls is lower than maxIter + maxIterNoColreg, resulting in an upper limit of 400 simulation calls. The trajectory temporal length of MPC is the time horizon. MPC has 52 discrete thrust and course deviation combinations and a 20 step horizon, and therefore uses 1040 simulation calls. Table IV indicates that COLREG-RRT utilizes forward simulation calls approximately 6 times more efficiently, i.e., capable of identifying longer trajectories using the same amount of forward simulation calls. This high efficiency is inherit to sampling-based methods, such as RRT, compared to methods that discretize actions, such as MPC-based methods.

Method	Avg. trajectory temporal length	# of forward simulation calls	Efficiency
COLREG-RRT	457±231s min: 140s max: 1140s	<400	>1.14
MPC	200±0s	1040	0.19

TABLE IV: Temporal length of identified trajectories and the number of forward simulation calls. The efficiency is defined as the average trajectory temporal length divided by the number of forward simulation calls.

In order to evaluate the impact of joint forward simulations used by COLREG-RRT in comparison to linear extrapolations used by MPC, we also tested a version of

COLREG-RRT with linear extrapolation used for obstacle ship motion. Table V lists the success rate and finish time. Using linear extrapolation results in more collisions than joint forward simulations, as seen in the lower success rate. However, the success rate of COLREG-RRT with linear extrapolation is higher than that of MPC. This is likely due to the fact that COLREG-RRT identifies collision-free paths while MPC optimizes a heuristic cost function that linearly combines risk of collision, time to collision, finish time, and COLREGS-compliance. As a result, paths that are in collision may be chosen. The finish time of both COLREG-RRTs are comparable since both methods use the same mechanisms to reduce finish time. The use of linear extrapolation increased the COLREGS-violation rate to 7.06% of the time in the most crowded problem tested, roughly the same MPC. This indicates that accurate simulation is important for COLREGS-compliance in multi-ship encounters.

# of Obstacles	1 Small + 4 Large	3 Small + 7 Large	5 Small + 10 Large	7 Small + 13 Large
Success rate	0.97±0.03	0.93±0.07	0.88±0.08	0.89±0.08
Finish time (s)	1146±113	1170±122	1185±132	1178±118

TABLE V: Success rate and finish time of a version of COLREG-RRT with linear extrapolation used for obstacle ship motion.

VI. Conclusion

In this paper, we proposed COLREG-RRT, a RRT-based motion planner for ASVs. We demonstrated that our algorithm has a higher navigation success rate and COLREGS-compliance compared to state of the art MPC-based and APF-based methods. Our method is also more efficient in identifying long-term trajectories given a limited amount of forward simulation calls, a property critical for real-time navigation of surface vessel with large inertia.

VII. Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Numbers IIS-1528047 and IIS-1553266 (Tapia, CAREER). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] European Maritime Safety Agency, "Annual overview of marine casualties and incidents 2016," http://www.emsa.europa.eu/emsa-documents/latest/download/4524/2903/23.html, accessed: 2017-09-10.

[2] Y. Xue, B. Lee, and D. Han, "Automatic collision avoidance of ships," *J. Engi. for the Maritime Envir.*, vol. 223, no. 1, pp. 33–46, 2009.

[3] International Maritime Organization, "COLREGS:convention on the international regulations for preventing collisions at sea," 1972.

[4] T. A. Johansen, T. Perez, and A. Cristofaro, "Ship collision avoidance and COLREGS compliance using simulation-based control behavior selection with predictive hazard assessment," *IEEE Tran. on Intel. Transp. Sys.*, vol. 17, no. 12, pp. 3407–3422, 2016.

[5] Ø. A. G. Loe, "Collision avoidance for unmanned surface vehicles," *Master thesis, Norwegian University of Science and Technology Department of Engineering Cybernetics*, 2008.

[6] S.-M. Lee, K.-Y. Kwon, and J. Joh, "A fuzzy logic for autonomous navigation of marine vehicles satisfying COLREG guidelines," *Intl. J. Control, Autom., and Sys.*, vol. 2, no. 2, pp. 171–181, 2004.

[7] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger, "Safe maritime autonomous navigation with COLREGS, using velocity obstacles," *IEEE J. Oceanic Engi.*, vol. 39, no. 1, pp. 110–119, 2014.

[8] T. Fraichard, "Trajectory planning in a dynamic workspace: a state-time space approach," *Adv. Rob.*, vol. 13, no. 1, pp. 75–94, 1998.

[9] O. Khatib, "Real–time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.

[10] P. Svec, B. C. Shah, I. R. Bertaska, J. Alvarez, A. J. Sinisterra, K. Von Ellenrieder, M. Dhanak, and S. K. Gupta, "Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic," in *Proc. IEEE Int. Conf. on Intel. Robot. Sys. (IROS)*, 2013, pp. 3871–3878.

[11] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[12] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2013, pp. 5054–5061.

[13] L. Palmieri and K. O. Arras, "Distance metric learning for RRT-based motion planning with constant-time inference," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2015, pp. 637–643.

[14] S. D. Pendleton, W. Liu, H. Andersen, Y. H. Eng, E. Frazzoli, D. Rus, and M. H. Ang, "Numerical approach to reachability-guided sampling-based motion planning under differential constraints," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1232–1239, 2017.

[15] E. Hernandez, M. Carreras, P. Ridao, J. Antich, and A. Ortiz, "A search-based path planning algorithm with topological constraints. application to an AUV," in *Proc. of Intl. Fed. of Automatic Control (IFAC)*, vol. 44, no. 1, 2011, pp. 13 654–13 659.

[16] M. Čáp, P. Novák, J. Vokrínek, and M. Pěchouček, "Multi-agent RRT*: sampling-based cooperative pathfinding," in *Proc. of Intl. Conf. on Auton. agents and multi-agent Sys.*, 2013, pp. 1263–1264.

[17] H.-T. Chiang, N. Rackley, and L. Tapia, "Runtime SES planning: Online motion planning in environments with stochastic dynamics and uncertainty," in *Proc. IEEE Int. Conf. on Intel. Robot. Sys. (IROS)*. IEEE, 2016, pp. 4802–4809.

[18] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions." in *Proc. Robotics: Sci. Sys. (RSS)*, 2016, pp. 66–73.

[19] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *Proc. IEEE Int. Conf. on Intel. Robot. Sys. (IROS)*, 2010, pp. 797–803.

[20] T. I. Fossen, *Marine control systems: guidance, navigation and control of ships, rigs and underwater vehicles*. Marine Cybernetics, 2002.

[21] Z. Liu, Y. Zhang, X. Yu, and C. Yuan, "Unmanned surface vehicles: An overview of developments and challenges," *Annual Rev. in Control*, vol. 41, pp. 71–93, 2016.

[22] T. Fraichard and H. Asama, "Inevitable collision states: A step towards safer robots?" *Adv. Rob.*, vol. 18, no. 10, pp. 1001–1024, 2004.

[23] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," in *Ameri. Control Conf. (ACC)*, vol. 1. IEEE, 2001, pp. 43–49.

[24] T. Statheros, G. Howells, and K. M. Maier, "Autonomous ship collision avoidance navigation concepts, technologies and techniques," *The J. of Navig.*, vol. 61, no. 1, pp. 129–142, 2008.

[25] S. Campbell, W. Naeem, and G. W. Irwin, "A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres," *Annual Rev. in Control*, vol. 36, no. 2, pp. 267–283, 2012.

[26] J. J. Park and B. Kuipers, "Feedback motion planning via non-holonomic RRT* for mobile robots," in *Proc. IEEE Int. Conf. on Intel. Robot. Sys. (IROS)*, 2015, pp. 4035–4040.