Check for
updates

Hybrid Collision Avoidance for ASVs Compliant With COLREGs Rules 8 and 13–17

Bjørn-Olav H. Eriksen*, Glenn Bitar, Morten Breivik and Anastasios M. Lekkas

Department of Engineering Cybernetics, Centre for Autonomous Marine Operations and Systems, Norwegian University of Science and Technology, Trondheim, Norway

This paper presents a three-layered hybrid collision avoidance (COLAV) system for autonomous surface vehicles, compliant with rules 8 and 13–17 of the International Regulations for Preventing Collisions at Sea (COLREGs). The COLAV system consists of a high-level planner producing an energy-optimized trajectory, a model-predictive-control-based mid-level COLAV algorithm considering moving obstacles and the COLREGs, and the branching-course model predictive control algorithm for short-term COLAV handling emergency situations in accordance with the COLREGs. Previously developed algorithms by the authors are used for the high-level planner and short-term COLAV, while we in this paper further develop the mid-level algorithm to make it comply with COLREGs rules 13–17. This includes developing a state machine for classifying obstacle vessels using a combination of the geometrical situation, the distance and time to the closest point of approach (CPA) and a new CPA-like measure. The performance of the hybrid COLAV system is tested through numerical simulations for three scenarios representing a range of different challenges, including multi-obstacle situations with multiple simultaneously active COLREGs rules, and also obstacles ignoring the COLREGs. The COLAV system avoids collision in all the scenarios, and follows the energy-optimized trajectory when the obstacles do not interfere with it.

Keywords: hybrid collision avoidance, autonomous surface vehicle (ASV), COLREGs, COLREGs compliant, model predictive control (MPC), energy-optimized control

1. INTRODUCTION

Motivated by the potential for reduced costs and increased safety, the maritime industry is rapidly moving toward autonomous operations. Following groundbreaking advances in the automotive industry, many sectors within the maritime industry are considering the benefits of autonomy, which includes more environmentally friendly operations. For instance, the agricultural chemical company *Yara* together with the maritime technology supplier *Kongsberg Maritime* are developing the electrical autonomous container vessel *Yara Birkeland*, which aims to replace 40,000 yearly truck journeys in urban eastern Norway[1]. Another example is the world's first autonomous car ferry, *Falco*, developed by *Rolls-Royce* (recently bought by Kongsberg Maritime) and *Finferries*. In

---

[1] https://www.wsj.com/articles/norway-takes-lead-in-race-to-build-autonomous-cargo-ships-1500721202 (accessed May 22, 2019).

2018, *Falco* navigated autonomously between two ports in Finland[2]. Reports state that in excess of 75 % of maritime accidents are due to human errors (Chauvin, 2011; Levander, 2017), indicating that there is also a potential for increased safety in addition to the economical and environmental benefits.

An obvious prerequisite for autonomous ship operations is the development of robust and well-functioning COLAV systems. In addition to generating collision-free maneuvers, a COLAV system must adhere to the "rules of the road" of the oceans, i.e., the COLREGs (Cockcroft and Lameijer, 2004). These rules are written for human ship operators and include qualitative requirements on how to perform safe and readily observable maneuvers. Part B of the COLREGs concern steering and sailing, and includes the following rules that are the most relevant to a motion control system:

Rule 8   Requires maneuvers to be readily observable and to be done in ample time.

Rules 13–15 Describe the maneuvers to perform in cases of overtaking, head-on and crossing situations. Participants in crossing situations are defined by the terms *give-way* and *stand-on* vessels.

Rule 16   Requires that a give-way vessel must take early and substantial action to keep clear of the stand-on vessel.

Rule 17   Consists of two main parts. The first part requires a stand-on vessel to maintain its course and speed, while the second part allows/requires[3] a stand-on vessel to take action to avoid collision if the give-way vessel is not taking action.

Since the rules are written for humans, with few quantitative figures, a challenge for autonomous operation is to quantify them into behaviors that can be executed algorithmically. The focus of the work in this paper is to do that, and to design a hybrid COLAV system that performs motion planning and generates maneuvers in compliance to rules 8 and 13–17 of the COLREGs.

A number of COLAV approaches considering the COLREGs have been proposed in the past. This includes algorithms using simulation-based model predictive control (Hagen et al., 2018), velocity obstacles (Kuwata et al., 2014), rule-based repairing A* (Campbell et al., 2014), and interval programming (Benjamin et al., 2006). All these approaches are single-layer approaches, where one algorithm solves the complete COLAV problem.

Another approach to the COLAV problem is to use a hybrid architecture, where the task of planning an obstacle-free path or trajectory, complying with the COLREGs and ultimately
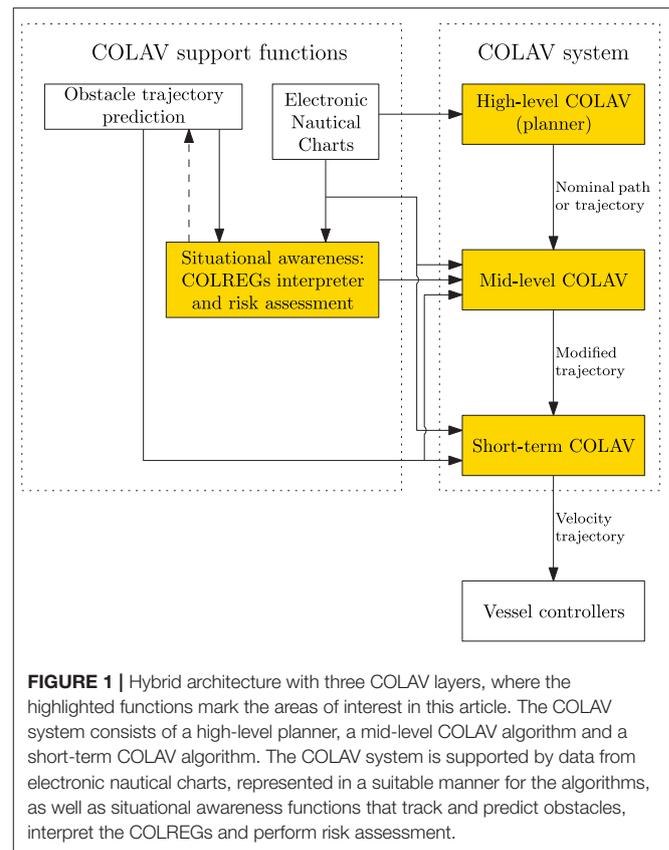

FIGURE 1 | Hybrid architecture with three COLAV layers, where the highlighted functions mark the areas of interest in this article. The COLAV system consists of a high-level planner, a mid-level COLAV algorithm and a short-term COLAV algorithm. The COLAV system is supported by data from electronic nautical charts, represented in a suitable manner for the algorithms, as well as situational awareness functions that track and predict obstacles, interpret the COLREGs and perform risk assessment.

performing safe maneuvers is divided into layers in a control hierarchy. The idea of hybrid architectures is to divide the subtasks of the COLAV problem into multiple algorithms, exploiting their complementary strengths. This also has the side effect of making it easier for human operators or supervisors to understand the system. Most single-layer algorithms use sample-based approaches that consider a finite number of discrete control inputs, as opposed to conventional gradient-based search algorithms. The reason for this is that many gradient-based algorithms are not sufficiently numerically robust, not allowing a COLAV system to solely rely on such an algorithm. This issue can be handled in hybrid architectures, constrained by the bottom-level algorithm being numerically robust and able to handle extraordinary situations where the other algorithms fail. Hence, hybrid architectures also allows using gradient-based algorithms, which are able to solve problems with large search spaces more efficiently than sample-based algorithms. The works by Loe (2008) and Švec et al. (2013) are examples of two-layered hybrid COLAV architectures. The top layers perform trajectory planning among static obstacles, while the bottom layers perform moving obstacle avoidance in compliance with COLREGs rules 13–16. Casalino et al. (2009) presents a three-layered hybrid COLAV system where the top layer also performs trajectory planning amongst static obstacles. The middle layer avoids moving obstacles, while the bottom layer implements safety functions for handling cases where the two

---

[2]https://www.marinemec.com/news/view,rollsroyce-and-finferries-demonstrate-worlds-first-fully-autonomous-ferry_56102.htm (accessed April 11, 2019).
[3]Rule 17 *allows* the stand-on vessel to maneuver when it becomes apparent that the give-way vessel does maneuver to avoid collision. If the vessels are so close that the give-way vessel cannot avoid collision by itself, Rule 17 *requires* the stand-on vessel to maneuver.

other layers fail. This approach does, however, not consider the COLREGs.

Figure 1 shows a three-layered hybrid COLAV system for an autonomous surface vehicle (ASV). The authors have previously worked extensively on different components of this architecture. Examples include high-level COLAV algorithms (Bitar et al., 2018, 2019b), a mid-level algorithm (Eriksen and Breivik, 2017b; Bitar et al., 2019a), short-term algorithms (Eriksen et al., 2018, 2019; Eriksen and Breivik, 2019) and the development of high-performance vessel controllers (Eriksen and Breivik, 2017a, 2018).

In this paper, we demonstrate the three-layered hybrid COLAV shown in **Figure 1** by combining and extending the COLAV algorithms developed in Eriksen and Breivik (2017b, 2019), Bitar et al. (2019a,b), Eriksen et al. (2019). The high-level planner has a long temporal horizon, and finds an energy-optimized nominal trajectory from an initial to a goal position. It considers static obstacles, which may include bathymetric constraints. Since the high-level planner only considers static information, it is intended to be run offline, but it can also be run online, for instance if new static obstacles are detected. The mid-level algorithm attempts to follow the nominal trajectory, while performing COLAV of static and moving obstacles in compliance with COLREGs rules 8, 13–16, and the first part of Rule 17. The mid-level algorithm is run periodically with a shorter temporal horizon than the high-level algorithm, and produces a modified trajectory which is passed to the short-term layer. Both the high-level and mid-level algorithms use gradient-based optimization. The short-term algorithm attempts to follow the modified trajectory, while it in compliance with the second part of Rule 17 handles situations where obstacles ignore the COLREGs. This algorithm also handles other emergency situations, and uses sample-based optimization to achieve a high level of robustness, ensuring safe operation if the mid-level algorithm fails to find a solution. The following list summarizes our contributions:

- The high-level planner from Bitar et al. (2019b) is modified to include the mathematical model of the *Telemetron* ASV in Bitar et al. (2019a), including ocean currents.
- The development of a state-machine-based COLREGs interpretation scheme.
- The mid-level COLAV from Bitar et al. (2019a) is modified to include rules 13–16 and the first part of Rule 17.
- The branching-course model predictive control (BC-MPC) algorithm for short-term COLAV is modified to reduce oscillatory behavior in turns.
- The three-layered COLAV system is verified in simulations and shown to be compliant with rules 8 and 13–17.

The rest of the paper has the following structure: The mathematical model of the ASV *Telemetron* is described in section 2. The high-level planner, mid-level and short-term COLAV algorithms are described in sections 3–5, respectively. In section 6 we present and discuss the simulation scenarios and results, and we conclude the paper in section 7.

FIGURE 2 | The Telemetron ASV, designed for both manned and unmanned operations. Courtesy of Maritime Robotics.

2. ASV MODELING

The vessel of interest in this article is the Telemetron ASV, which is owned and operated by the Norwegian company Maritime Robotics and shown in **Figure 2**. The Telemetron ASV is a high-speed dual-use vessel propelled by a steerable outboard engine, capable of speeds up to 18 m/s.

Eriksen and Breivik (2017a) presents a model of the Telemetron ASV, which is extended to include ocean currents in Bitar et al. (2019a). The model has the form

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{x}_r + \begin{bmatrix} \boldsymbol{V}_c^\top & 0 \end{bmatrix}^\top$$
$$\boldsymbol{M}(\boldsymbol{x}_r)\dot{\boldsymbol{x}}_r + \boldsymbol{\sigma}(\boldsymbol{x}_r) = \boldsymbol{\tau} \,, \tag{1}$$

where $\boldsymbol{\eta} = [x, y, \psi]^\top \in \mathbb{R}^2 \times S$ is the vessel pose and $\boldsymbol{V}_c = [V_x, V_y]^\top$ describes the ocean current, both in the Earth-fixed North-East-Down frame $\{n\}$. The vector $\boldsymbol{x}_r = [u_r, r]^\top \in \mathbb{X}_r \subset \mathbb{R}^2$ is the vessel velocity under the assumption of zero relative sway motion (Bitar et al., 2019a), where the set $\mathbb{X}_r$ describes the vessel-feasible steady-state velocities where (1) is valid. The transformation matrix $\boldsymbol{R}(\psi)$ is given by the heading $\psi \in S$ as

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos\psi & 0 \\ \sin\psi & 0 \\ 0 & 1 \end{bmatrix} \,, \tag{2}$$

while $r \in \mathbb{R}$ describes the vessel yaw-rate. The matrix $\boldsymbol{M}(\boldsymbol{x}_r)$ is a state-dependent inertia matrix, while $\boldsymbol{\sigma}(\boldsymbol{x}_r)$ and $\boldsymbol{\tau} = [\tau_m, \tau_\delta]^\top \in \mathbb{U} \subset \mathbb{R}^2$ describe the vessel damping and control input, respectively. The set $\mathbb{U}$ describes the control inputs where (1) is valid.

In this work, we assume that the ocean current $\boldsymbol{V}_c$ is constant and known. For practical applications, the ocean current can be measured via appropriate instrumentation, estimated via sensor fusion methods, or predicted based on e.g., tide tables or sensor networks, such as the European marine observation and data network[4].

---

[4]http://www.emodnet.eu/ (accessed December 11, 2019).

3. HIGH-LEVEL PLANNER

To plan the ASV's nominal trajectory, we use a high-level trajectory planner developed in Bitar et al. (2019b). This trajectory planner uses the ASV model described in section 2 to generate an energy-optimized trajectory between the start and goal positions. The planning algorithm combines an A⋆ implementation and an optimal control problem (OCP) solver to generate a feasible and optimized trajectory.

The high-level planning algorithm consists of three steps: First the A⋆ implementation finds the shortest piecewise linear path between the start and goal position. Secondly, artificial temporal information is added to the path, converting it to a trajectory of states and inputs. Finally, the trajectory is used as an initial guess for an OCP solver, which finds a locally energy-optimized trajectory near the shortest path. All steps account for static obstacles in the form of elliptical boundaries.

3.1. Static Obstacles

The elliptical boundaries are described with the inequality:

$$\left(\frac{x - x_c}{x_a}\right)^2 + \left(\frac{y - y_c}{y_a}\right)^2 \geq 1, \tag{3}$$

where x_c and y_c is the ellipsis center, and $x_a, y_a > 0$ are the ellipsis major and minor axes, respectively. To allow for angled obstacles, the ellipses are rotated clockwise by an angle α. We add a small constant $\epsilon > 0$ to each side of the inequality, and take the logarithm to arrive at the following obstacle representation:

$$h_o(x, y, x_c, y_c, x_a, y_a, \alpha) = -\log\left[\left(\frac{(x - x_c)\cos\alpha + (y - y_c)\sin\alpha}{x_a}\right)^2 \right.$$
$$\left. + \left(\frac{-(x - x_c)\sin\alpha + (y - y_c)\cos\alpha}{y_a}\right)^2 + \epsilon\right] + \log(1 + \epsilon) \leq 0. \tag{4}$$

The logarithm operation is applied to reduce the numerical range of the inequality, which helps with numerical stability of the subsequently described solver, and the constant ϵ is included to avoid singularities when (4) is evaluated for $(x, y) \rightarrow (x_c, y_c)$ (Bitar et al., 2019a).

Modeling static obstacles as ellipses poses a challenge for handling obstacles of various shapes, from e.g., electronic nautical charts (ENCs). Generic obstacle shapes can be approximated as a set of elliptical obstacles (Wu, 2019), although this may require a large number of constraints for complex environments. Alternatively, the obstacle modeling can be modified to allow for generic shapes. Zhang et al. (2018) present an interesting solution to handle polygon-shaped obstacles by introducing a signed distance function in the optimization problem. Unfortunately, this approach introduces a large number of slack variables and constraints, limiting feasibility for more than a few static obstacles.

3.2. Trajectory Generation and Optimization

From a scenario consisting of static obstacles, as mentioned in section 3.1, we find the piecewise linear shortest path by performing an A⋆ search on a uniformly decomposed grid. The resulting path is converted to a time-parameterized full-state trajectory by assuming a constant forward velocity, and connecting the shortest path with straight segments and circle arcs. The constant forward velocity is

$$u_{\text{nom}} = \frac{L_{\text{path}}}{t_{\text{max}}}, \tag{5}$$

where L_{path} is the length of the connected path, and t_{max} is the maximum allowed time to complete the trajectory. This full-state trajectory is then used as an initial guess to solve the OCP that gives the energy-optimized trajectory:

$$\min_{z(\cdot), \tau(\cdot)} \int_0^{t_{\text{max}}} F_{\text{hi}}(z(t), \tau(t)) dt \tag{6a}$$

subject to

$$\dot{z}(t) = f(z(t), \tau(t)) \ \forall t \in [0, t_{\text{max}}] \tag{6b}$$

$$h_{\text{hi}}(z(t), \tau(t)) \leq 0 \ \forall t \in [0, t_{\text{max}}] \tag{6c}$$

$$e_{\text{hi}}(z(0), z(t_{\text{max}})) = 0. \tag{6d}$$

The solution of this OCP is a trajectory of states $z(\cdot)$ and inputs $\tau(\cdot)$ that minimizes the cost functional in (6a). The ASV model from section 2 is rewritten as $\dot{z} = f(z, \tau)$, where $z = [\eta^\top, x_r^\top]^\top$ and $f(z, \tau)$ represents (1).

The cost functional (6a) is chosen to minimize energy. The cost-to-go function is

$$F_{\text{hi}}(z, \tau) = K_e F_e(z, \tau) + K_\delta \tau_\delta^2, \tag{7}$$

with tuning parameters $K_e, K_\delta > 0$. The first term consists of a function that is proportional to mechanical work performed by the ASV:

$$F_e(z, \tau) = \underbrace{|n(\tau_m)^2 \cdot \cos\delta(\tau_\delta) \cdot u_r|}_{\propto \text{ surge force}} + \underbrace{|n(\tau_m)^2 \cdot \sin\delta(\tau_\delta) \cdot L_m \cdot r|}_{\propto \text{ yaw moment}}. \tag{8}$$

The function $n : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ maps the control input τ_m to propeller angular velocity. The function $\delta : \mathbb{R} \rightarrow S$ maps the control input τ_δ to outboard motor angle. The second term in (8) is a quadratic cost to yaw control, included to avoid issues with singularity when solving the OCP.

The inequality constraints (6c) observe state boundaries as well as the static obstacles as represented in section 3.1. The boundary conditions (6d) denote initial and final constraints, i.e., start and end states.

A detailed description of the transcription of the OCP (6) to a non-linear program (NLP) using multiple shooting with N_{hi} shooting intervals is found in Bitar et al. (2019b).

4. MID-LEVEL COLAV

The mid-level algorithm, initially presented in Eriksen and Breivik (2017b) and further developed in Bitar et al. (2019a), is a model predictive control (MPC)-based algorithm intended for long-term COLAV. The algorithm utilizes gradient-based

optimization, and takes both static and moving obstacles into account while attempting to follow an energy-optimized nominal trajectory from the high-level planner. The algorithm produces maneuvers complying with Rule 8 of the COLREGs, which requires maneuvers to be made in ample time and be readily observable for other vessels. The optimization problem is formulated as a NLP, which gives flexibility in designing the optimization problem.

In this section, the algorithm is extended to also consider COLREGs rules 13–16 and the first part of Rule 17.

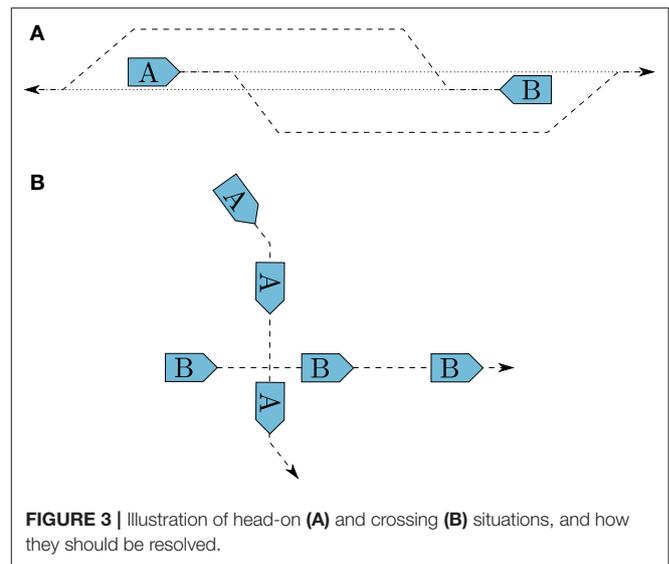## 4.1. The International Regulations for Preventing Collisions at Sea (COLREGs)

The COLREGs consists of a total of 37 rules and is divided into five parts (Cockcroft and Lameijer, 2004), where part B (rules 4–19) contains relevant rules on the conduct of vessels in proximity of each other. The most relevant rules for designing COLAV systems in part B are rules 8 and 13–17:

Rule 8   **Action to avoid collision.** This rule states that actions taken to avoid collision should be large enough to be readily observable of other ships, implying that series of small alternations in speed and/or course should not be applied. The rule also recommends that course changes should be prioritized over speed changes if there is enough free space available, and that maneuvers must be made in ample time.

Rule 13   **Overtaking.** This rule states that a vessel is overtaking another if it approaches the other vessel with a course more than 22.5° abaft her beam. The overtaking vessel has to stay clear of the overtaken vessel, but there is no statements on which side of the vessel one should pass.

Rule 14   **Head on.** When two power-driven vessels approach each other on reciprocal, or nearly reciprocal, courses, they are in a head-on situation. In such a situation, both vessels should change their course to starboard, passing each other port-to-port, as shown in **Figure 3A**. This rule states no explicit definition on what should be considered to be reciprocal, or nearly reciprocal, courses, but court decisions indicate head-on situations exist for opposing courses ±6°. Notice that the rule does not include sailing vessels, which are covered by Rule 12.

Rule 15   **Crossing.** When two vessels approach each other such that the situation is not a head on or an overtaking, it is a crossing situation. The vessel with the other one to her starboard side is deemed the give-way vessel, while the other vessel is deemed the stand-on vessel. As shown in **Figure 3B**, the give-way vessel should maneuver to avoid collision, preferably by passing behind the stand-on vessel, while the stand-on vessel should keep her speed and course.

FIGURE 3 | Illustration of head-on **(A)** and crossing **(B)** situations, and how they should be resolved.

Rule 16   **Action by the give-way vessel.** Every vessel which is required to keep out of the way of another vessel should take early and large enough action to safely avoid collision.

Rule 17   **Action by the stand-on vessel.** This rule requires that a stand-on vessel should keep its current speed and course. The stand-on vessel may, however, maneuver to avoid collision if it becomes apparent that the give-way vessel is not taking appropriate actions to avoid collision. Furthermore, if the stand-on vessel finds itself so close to the obstacle that collision can not be avoided by the give-way vessel alone, the stand-on vessel should take such action which best aids to avoid collision. In a crossing situation, the stand-on vessel should avoid maneuvering to port, since this could lead to a collision if the give-way vessel maneuvers to starboard.

In the hybrid architecture illustrated in **Figure 1**, the mid-level algorithm is given the task of strictly enforcing COLREGs rules 13–16 and the stand-on requirement of Rule 17, while also complying with Rule 8.

In addition, we want the mid-level algorithm to comply with the first part of Rule 17, by not maneuvering to avoid collision in crossing situations if the ownship is the stand-on vessel. The hybrid COLAV system is inherently capable of adhering to the remaining requirement of Rule 17, where the stand-on vessel is allowed or required to maneuver, by having different prediction horizons and safety margins in the mid-level and short-term layers. The BC-MPC algorithm does not have any limitations of not maneuvering in stand-on situations, and will hence maneuver in stand-on situations if we come sufficiently close to the obstacle.

The mid-level algorithm as presented in Bitar et al. (2019a) only complies with Rule 8. Further in this section, we therefore present improvements to the mid-level algorithm to make it

comply with rules 13–16 and the stand-on requirement of Rule 17.
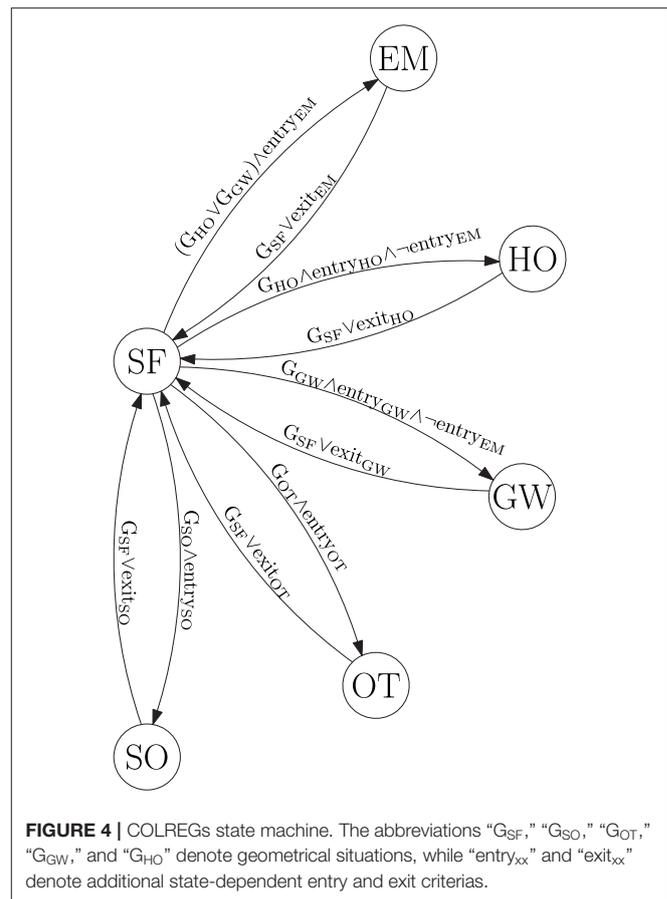
4.2. COLREGs Interpretation

A commonly used concept for interpreting obstacles in COLAV algorithms is to assign a spatial region to obstacles, which the ownship should not enter. This approach is commonly referred to as a *domain-based* approach. Specially designed ship domains are commonly used for interpreting the COLREGs in COLAV algorithms, where the required clearance to an obstacle is significantly larger if the maneuver violates the COLREGs (Szlapczynski and Szlapczynska, 2017; Eriksen et al., 2019). This approach is attractive since it continuously captures multiple COLREGs rules, and does not require logic or discrete decisions. However, such an approach does not strictly enforce the COLREGs rules, since it will allow maneuvers violating the rules if they are large enough. In addition, a ship-domain approach will not be able to strictly enforce the stand-on requirement of Rule 17, since a domain-based approach will avoid collision with all obstacles. One could ignore obstacles with give-way obligations, but this would require an explicit COLREGs interpretation which conflicts with domain-based approaches' core idea of implicit COLREGs interpretation. Therefore, we pursue an alternative approach to handling the COLREGs in the mid-level algorithm.

To simplify the COLREGs interpretation task, we look at the situation from a static perspective, assuming that the current COLREGs situations are valid throughout the entire prediction horizon of the mid-level algorithm. In reality, the COLREGs situations may, however, change during the prediction horizon depending on both the ownship's and obstacles future trajectory. For instance, an obstacle approaching from head on, but far enough away to not be considered as a danger may be put in a safe state. Hence, the mid-level algorithm will (for the current iteration) act like no COLREGs rule applies to this vessel for the entire prediction horizon, while the obstacle may get close enough during the prediction horizon to be considered as a head-on situation. An MPC scheme of only implementing a small part of the prediction horizon will reduce the implications of this, since the situation is reassessed each time mid-level algorithm is run, which justifies the assumption of considering the COLREGs from a static perspective. Investigating the possibilities for dynamically predicted future COLREGs situations as part of the MPC prediction will be considered as future work.

4.2.1. State Machine

We propose to utilize a state machine in order to decide which COLREGs rule is active with respect to each obstacle in the vicinity of the ownship. The state machine contains the states:

SF   Safe state. This implies that the COLREGs do not enforce any rule with respect to this obstacle.

OT   Overtaking state. This implies that COLREGs Rule 13 applies with respect to this obstacle. The state machine does not discriminate on whether the ownship is overtaking another vessel or is being overtaken, but this can be done by looking at which vessel has the higher speed (Tam and Bucknall, 2010).

FIGURE 4 | COLREGs state machine. The abbreviations "G_{SF}," "G_{SO}," "G_{OT}," "G_{GW}," and "G_{HO}" denote geometrical situations, while "$entry_{xx}$" and "$exit_{xx}$" denote additional state-dependent entry and exit criterias.

HO   Head-on state. This implies that COLREGs Rule 14 applies with respect to this obstacle.

GW   Give-way state. This implies that COLREGs Rule 15 applies with respect to this obstacle, and the ownship has to give way.

SO   Stand-on state. This implies that COLREGs Rule 15 applies with respect to this obstacle, and the ownship has to stand on.

EM   Emergency state. This implies that the obstacle is so close and/or behaves unpredictably, such that special considerations must be made.

As shown in **Figure 4**, all transitions have to go either from or to the safe state.

This implies that when the state machine decides that a COLREGs (or emergency) situation exists with respect to an obstacle, it will not allow switching to another state without the situation being considered as safe first. One could argue that it should be able to transition between specific states, like e.g., from head-on, give-way and overtaking to emergency. This is an interesting topic, which should receive attention in the future. To control the transitions between the different states, we combine the time to and distance at the CPA, a CPA-like measure of the

time until a critical point and a geometrical interpretation of the situation.

4.2.2. Entry and Exit Criteria

CPA is a common concept in maritime risk assessment. Given the current speed and course of the ownship and an obstacle, CPA describes the time to the point where the two vessels are the closest, and the distance to the obstacle at this point. Given the position and velocity vector of the ownship $\boldsymbol{p}, \boldsymbol{v}$ and an obstacle $\boldsymbol{p}_o, \boldsymbol{v}_o$, the time to CPA is calculated as (Kufoalor et al., 2018)

$$t_{\mathrm{CPA}} = \begin{cases} 0 & \text{if } \|\boldsymbol{v} - \boldsymbol{v}_0\|_2 \leq \epsilon \\ \frac{(\boldsymbol{p} - \boldsymbol{p}_o) \cdot (\boldsymbol{v} - \boldsymbol{v}_o)}{\|\boldsymbol{v} - \boldsymbol{v}_o\|_2^2} & \text{else,} \end{cases} \tag{9}$$

where $\epsilon > 0$ is a small constant in order to avoid division by zero in the case where the relative velocity between the ownship and obstacle is zero. Given t_{CPA}, we calculate the distance between the vessels at CPA as

$$d_{\mathrm{CPA}} = \|(\boldsymbol{p} + t_{\mathrm{CPA}}\boldsymbol{v}) - (\boldsymbol{p}_o + t_{\mathrm{CPA}}\boldsymbol{v}_o)\|_2. \tag{10}$$

While the CPA is the point where the distance to an obstacle is at its minimum, the critical point is where the distance to an obstacle crosses underneath a critical distance d_{crit}. This critical distance describes a minimum obstacle distance that the mid-level algorithm is designed for. The time to the critical point t_{crit} can be calculated by solving the equation

$$\|(\boldsymbol{p} + t_{\mathrm{crit}}\boldsymbol{v}) - (\boldsymbol{p}_o + t_{\mathrm{crit}}\boldsymbol{v}_o)\|_2 = d_{\mathrm{crit}}. \tag{11}$$

In the cases where the distance between the ships does not fall below d_{crit}, t_{crit} is undefined. Otherwise, there are generally two solutions. The interesting solution is the one with the lowest t_{crit} value, as this is when the obstacle enters the d_{crit} boundary.

The state-machine entry criteria in **Figure 4** are defined as

$$\mathrm{entry}_i = \begin{cases} \text{true} & \text{if } d_{\mathrm{CPA}} < \overline{d}_{\mathrm{CPA}}^{i,\mathrm{enter}} \wedge t_{\mathrm{CPA}} \in [\underline{t}_{\mathrm{CPA}}^{i,\mathrm{enter}}, \overline{t}_{\mathrm{CPA}}^{i,\mathrm{enter}}], \\ \text{false} & \text{otherwise} \end{cases}$$

$$\forall i \in \{\mathrm{SO,OT,GW,HO}\} \tag{12}$$

$$\mathrm{entry}_{\mathrm{EM}} = \begin{cases} \text{true} & \text{if } t_{\mathrm{crit}} < \overline{t}_{\mathrm{crit}}^{\mathrm{EM,enter}} \wedge t_{\mathrm{CPA}} > 0 \\ \text{false} & \text{otherwise,} \end{cases}$$

where $\overline{d}_{\mathrm{CPA}}^{i,\mathrm{enter}}, \underline{t}_{\mathrm{CPA}}^{i,\mathrm{enter}}$, and $\overline{t}_{\mathrm{CPA}}^{i,\mathrm{enter}}$ for $i \in \{\mathrm{SO,OT,GW,HO}\}$ are tuning parameters denoting thresholds on d_{CPA} and t_{CPA} in order to satisfy the entry criteria for the stand-on, overtaking, give-way and head-on states. The tuning parameter $\overline{t}_{\mathrm{crit}}^{\mathrm{EM,enter}}$ denotes an upper limit on t_{crit} in order to enter the emergency state. The idea behind the stand-on, overtaking, give-way and head-on entry criterias are that in order for the obstacle to represent a risk, both t_{CPA} and d_{CPA} need to be within some tunable thresholds. Situations with a very low d_{CPA}, but with a high t_{CPA}, will not trigger the entry criteria, since the situations will not occur in the near future. Similarly, if t_{CPA} is within the thresholds, but d_{CPA} is large, this indicates a safe passing where risk of collision does not exist. The lower bound on t_{CPA} will typically be selected as zero,

and is useful to distinguish between obstacles moving toward of away from the ownship. For the emergency state, the entry criteria is based on the critical point, at which we are so close that the mid-level algorithm may struggle with providing meaningful maneuvers. In addition to t_{crit} being under the threshold $\overline{t}_{\mathrm{crit}}^{\mathrm{EM,enter}}$, we require that t_{CPA} is positive, indicating that we are getting closer to the obstacle. Currently, we only allow entering the emergency state if the situation is a geometrical give-way or head-on, since an overtaking situation represents a smaller danger and has less requirement for special consideration.

The state-machine exit criterias in **Figure 4** are defined as

$$\mathrm{exit}_i = \begin{cases} \text{true} & \text{if } d_{\mathrm{CPA}} \geq \underline{d}_{\mathrm{CPA}}^{i,\mathrm{exit}} \vee t_{\mathrm{CPA}} \notin [\underline{t}_{\mathrm{CPA}}^{i,\mathrm{exit}}, \overline{t}_{\mathrm{CPA}}^{i,\mathrm{exit}}], \\ \text{false} & \text{otherwise} \end{cases}$$

$$\forall i \in \{\mathrm{SO,OT,GW,HO}\} \tag{13}$$

$$\mathrm{exit}_{\mathrm{EM}} = \begin{cases} \text{true} & \text{if } t_{\mathrm{crit}} \geq \underline{t}_{\mathrm{crit}}^{\mathrm{EM,exit}} \vee t_{\mathrm{CPA}} \leq 0 \\ \text{false} & \text{otherwise,} \end{cases}$$

where $\underline{d}_{\mathrm{CPA}}^{i,\mathrm{exit}}, \underline{t}_{\mathrm{CPA}}^{i,\mathrm{exit}}$, and $\overline{t}_{\mathrm{CPA}}^{i,\mathrm{exit}}$ for $i \in \{\mathrm{SO,OT,GW,HO}\}$ are tuning parameters denoting thresholds on d_{CPA} and t_{CPA} in order to satisfy the exit criteria for the stand-on, overtaking, give-way and head-on states. The exit criteria for the emergency state is satisfied if t_{crit} is larger than the tuning parameter $\underline{t}_{\mathrm{exit}}^{\mathrm{EM,enter}}$, or t_{CPA} is negative, implying that the obstacle is moving further away from the ownship. Note that the exit criterias are obtained by negating the entry criterias, but with other thresholds in order to implement hysteresis to avoid shattering. In general, we allow for different tuning parameters for the different states, but in our simulations we see that selecting the same tuning parameters for all states provides good results. Therefore, we define:

$$\begin{aligned} \overline{d}_{\mathrm{CPA}}^{i,\mathrm{enter}} &= \overline{d}_{\mathrm{CPA}}^{\mathrm{enter}} \\ \underline{t}_{\mathrm{CPA}}^{i,\mathrm{enter}} &= \underline{t}_{\mathrm{CPA}}^{\mathrm{enter}} \\ \overline{t}_{\mathrm{CPA}}^{i,\mathrm{enter}} &= \overline{t}_{\mathrm{CPA}}^{\mathrm{enter}}, \end{aligned} \tag{14}$$

and

$$\begin{aligned} \underline{d}_{\mathrm{CPA}}^{i,\mathrm{exit}} &= \underline{d}_{\mathrm{CPA}}^{\mathrm{exit}} \\ \underline{t}_{\mathrm{CPA}}^{i,\mathrm{exit}} &= \underline{t}_{\mathrm{CPA}}^{\mathrm{exit}} \\ \overline{t}_{\mathrm{CPA}}^{i,\mathrm{exit}} &= \overline{t}_{\mathrm{CPA}}^{\mathrm{exit}} \end{aligned} \tag{15}$$

for all $i \in \{\mathrm{SO,OT,GW,HO}\}$.

4.2.3. Geometrical Situation Interpretation

Tam and Bucknall (2010) present a geometrical interpretation scheme for deciding COLREGs situations based on the relative position, bearing and course of the obstacle with respect to the ownship. We base our geometrical interpretation on a slightly modified version of this scheme, where we include the sign of t_{CPA} to distinguish between situations where the obstacle moves closer toward or farther away from the ownship. The geometrical interpretation is shown in **Figure 5**, where the geometrical situation is obtained by finding which region the obstacle position and course resides in.

Notice that the head-on region is larger than the threshold of $\pm 6°$ as described by the COLREGs. The reason for this is that

FIGURE 5 | Illustration of the geometrical COLREGs interpretation, where the ownship course is denoted as χ and $\theta_1, \theta_2, \theta_3$ denote symmetrical regions given as [22.5, 90, 112.5°] offset from ahead. The circles illustrate obstacles in different relative bearing regions, and have a fixed orientation with respect to the ownship. The geometrical situations are color-coded and denoted as $G_i, i \in$ {SF, SO, OT, GW, HO} for safe, stand-on, overtaking, give-way and head-on situations, respectively. When two situations are given, like e.g., $G_{SF/SO}$, we use the former (SF) if $t_{CPA} < 0$ and the latter (SO) if $t_{CPA} \geq 0$, analogous to the obstacle moving away or toward the ownship. To decide the geometrical situation, we first find which relative bearing region the obstacle resides in, before finding which obstacle region the obstacle's course resides in. The figure is inspired by Tam and Bucknall (2010).

Tam and Bucknall recommend using a larger region of 22.5° in order to increase the robustness of the geometrical COLREGs interpretation scheme.

4.3. Interface to the High-Level Planner

The high-level planner produces an energy-optimized nominal trajectory for the ownship to follow. However, since the high-level planner does not consider moving obstacles, the speed is the only time-relevant factor of the desired trajectory. In a case where the ownship for some reason, e.g., avoiding moving obstacles, lag behind the nominal trajectory, following the nominal trajectory in absolute time would cause a speed increase in order to catch up with it. Therefore, the mid-level algorithm performs *relative trajectory tracking*, where it tracks the nominal trajectory with a time offset $t_b \in \mathbb{R}$. This results in a relative nominal trajectory for the mid-level

algorithm:

$$\bar{\boldsymbol{p}}_d(t) = \boldsymbol{p}_d(t + t_b), \tag{16}$$

where $\boldsymbol{p}_d = [N_d(t), E_d(t)]^\top$ is the nominal trajectory from the high-level planner. The time offset t_b is calculated each time the mid-level algorithm is run by solving a separate optimization problem, and is selected such that $\bar{\boldsymbol{p}}_d(t_0)$ is the point on the nominal trajectory closest to the ownship. See Bitar et al. (2019a) for a detailed description of this concept.

4.4. Optimization Problem Formulation

The mid-level algorithm is formalized as an OCP:

$$\min_{\boldsymbol{\eta}(\cdot), \boldsymbol{x}_r(\cdot)} \phi(\boldsymbol{\eta}(\cdot), \boldsymbol{x}_r(\cdot)) \tag{17a}$$

subject to

$$\dot{\boldsymbol{\eta}}(t) = \boldsymbol{R}(\psi(t))\boldsymbol{x}_r(t) + \begin{bmatrix} \boldsymbol{V}_c \\ 0 \end{bmatrix} \forall t \in [t_0, t_0 + T_h] \qquad (17b)$$

$$\boldsymbol{h}_{\text{mid}}(\boldsymbol{\eta}(t), \boldsymbol{x}_r(t), t) \leq \boldsymbol{0} \, \forall t \in [t_0, t_0 + T_h] \qquad (17c)$$

$$\boldsymbol{e}_{\text{mid}}(\boldsymbol{\eta}(t_0)) = \boldsymbol{0}, \qquad (17d)$$

where $T_h > 0$ is the prediction horizon, $\phi(\cdot, \cdot)$ is the objective functional, (17b) contains a kinematic vessel model, (17c) contains inequality constraints and (17d) contains boundary constraints.

Analytical solutions of OCPs are in general not possible to find. A more common approach is to transcribe the OCP to an NLP, and solve that using a gradient optimization scheme. In our case, we transcribe (17) into an NLP with N_p samples using multiple shooting, where the vessel model is discretized using 4th order Runge Kutta and the cost functional is discretized using forward Euler. The resulting NLP is given as

$$\min_{\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu}, \boldsymbol{\xi}} \phi_p(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu}) + \phi_c(\boldsymbol{w}) + \phi_{\text{COLREGs}}(\boldsymbol{w}) + \phi_\xi(\boldsymbol{\xi})$$

subject to

$$\boldsymbol{g}(\boldsymbol{w}, \boldsymbol{\eta}(t_0)) = \boldsymbol{0}$$
$$\boldsymbol{h}(\boldsymbol{w}, \boldsymbol{\xi}) \leq \boldsymbol{0} \qquad (18)$$
$$\bar{\boldsymbol{h}}_k(\boldsymbol{\eta}_k, \boldsymbol{\omega}_k, \boldsymbol{\mu}_k, \bar{\boldsymbol{p}}_{d,k}) \leq \boldsymbol{0} \, \forall k \in \{1, \ldots, N_p\}$$
$$\boldsymbol{\xi} \geq \boldsymbol{0},$$

where $\boldsymbol{w} = [\boldsymbol{\eta}_0^\top, \boldsymbol{x}_{r,0}^\top, \ldots, \boldsymbol{\eta}_{N_p-1}^\top, \boldsymbol{x}_{r,N_p-1}^\top, \boldsymbol{\eta}_{N_p}^\top]^\top \in \mathbb{R}^{5N_p+3}$ is a vector of $5N_p + 3$ decision variables and $\bar{\boldsymbol{p}}_{d,1:N_p} = [\bar{\boldsymbol{p}}_{d,1}, \bar{\boldsymbol{p}}_{d,2}, \ldots, \bar{\boldsymbol{p}}_{d,N_p}]$ is a sequence of desired positions. The vectors $\boldsymbol{\omega} \in \mathbb{R}^{2N_p}$, $\boldsymbol{\mu} \in \mathbb{R}^{2N_p}$ and $\boldsymbol{\xi} \in \mathbb{R}^{MN_p}$ contain slack variables, where M is the number of moving obstacles to be included in the constraints.

The vector $\boldsymbol{g}(\boldsymbol{w}, \boldsymbol{\eta}(t_0)) \in \mathbb{R}^{3N_p+3}$ contains shooting and boundary constraints, while $\boldsymbol{h}(\boldsymbol{w}) \in \mathbb{R}^{(M+D+4)N_p}$, where D is the number of static obstacles, contain inequality constraints ensuring COLAV and steady-state vessel velocity feasibility. The vectors $\bar{\boldsymbol{h}}_k(\boldsymbol{\eta}_k, \boldsymbol{\omega}_k, \boldsymbol{\mu}_k, \bar{\boldsymbol{p}}_{d,k}) \in \mathbb{R}^6$, $k \in \{1, N_p\}$ contain constraints on the slack variables $\boldsymbol{\omega}$ and $\boldsymbol{\mu}$.

In the following subsections, we describe the terms in (18) in more detail.

4.4.1. Objective Function
The objective function contains four functions, where $\phi_p(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu})$ introduces cost on deviating from the relative nominal trajectory $\bar{\boldsymbol{p}}_d(t)$, $\phi_c(\boldsymbol{w})$ introduces cost on using control input, $\phi_{\text{COLREGs}}(\boldsymbol{w})$ is a COLREGs-specific function and $\phi_\xi(\boldsymbol{\xi})$ introduces slack variable cost.

To avoid that the NLP changes behavior when moving away from the nominal trajectory, we wish to have linear growth in the position error function $\phi_p(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu})$. This is achieved by instead of using quadratic terms in the position error function, we use the Huber loss function which is quadratic around the

origin and resembles the absolute value function above a given threshold $\sigma > 0$:

$$H(\rho) = \begin{cases} \frac{1}{2}\rho^2 & |\rho| \leq \sigma \\ \sigma(|\rho| - \frac{1}{2}\sigma) & |\rho| > \sigma . \end{cases} \qquad (19)$$

The Huber loss function has a discontinuous gradient, making it slightly complicated to implement in gradient-based optimization problems. It can, however, be implemented in a continuous fashion by utilizing lifting, where slack variables are introduced to create a problem of a higher dimensionality which is easier to solve. Using this technique, $\bar{\phi}_p(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu})$ is defined as

$$\bar{\phi}_p(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu}) = K_p \sum_{k=1}^{N_p} \sigma \boldsymbol{1}^\top \boldsymbol{\omega}_k + \frac{1}{2}\boldsymbol{\mu}_k^\top \boldsymbol{\mu}_k, \qquad (20)$$

where $K_p > 0$ is a tuning parameter, and $\boldsymbol{\omega}_k \in \mathbb{R}^2$ and $\boldsymbol{\mu}_k \in \mathbb{R}^2$ are slack variables constrained by

$$\bar{\boldsymbol{h}}_k(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu}, \bar{\boldsymbol{p}}_{d,k}) = \begin{bmatrix} \boldsymbol{v_k} + \boldsymbol{\mu_k} + \boldsymbol{p_k} - \bar{\boldsymbol{p}}_{d,k} \\ \boldsymbol{v_k} + \boldsymbol{\mu_k} - (\boldsymbol{p_k} - \bar{\boldsymbol{p}}_{d,k}) \\ -\boldsymbol{\omega_k} \end{bmatrix} \leq \boldsymbol{0}$$
$$\forall k \in \{1, \ldots, N_p\}, \qquad (21)$$

where $\boldsymbol{p}_k$ is the predicted vessel position at time step k, i.e., $\boldsymbol{\eta}_k = [\boldsymbol{p}_k^\top, \psi_k]^\top$. See Bitar et al. (2019a) for more details.

Rule 8 of the COLREGs requires that maneuvers are readily observable for other vessels, implying that speed and course changes should have a sufficiently large magnitude, and not be performed as a sequence of small changes. In order to enforce this in the optimization problem, the control cost function $\phi_c(\boldsymbol{w})$ introduces a non-linear cost on the change in speed and course, which makes the algorithm favor readily observable maneuvers. The function is defined as

$$\phi_c(\boldsymbol{w}) = \sum_{k=0}^{N_p-1} K_{\dot{U}} q_{\dot{U}}(\dot{U}_k) + K_{\dot{\chi}} q_{\dot{\chi}}(\dot{\chi}_k), \qquad (22)$$

where $K_{\dot{U}}, K_{\dot{\chi}} > 0$ are tuning parameters, while $q_{\dot{U}}(\dot{U}_k)$ and $q_{\dot{\chi}}(\dot{\chi}_k)$ are the non-linear cost functions. Notice that neither the speed over ground (SOG) U nor the course χ are elements of the search space, but they can be computed as $U = \sqrt{u^2 + v^2}$ and $\chi = \psi + \arcsin \frac{v}{U}$. Their derivatives are then calculated by finite differencing. See Eriksen and Breivik (2017a) and Bitar et al. (2019a) for more details on the control cost function.

The $\phi_{\text{COLREGs}}(\boldsymbol{w})$ function introduces a COLREGs-specific cost with respect to obstacles based on the rule currently applicable as defined by the state machine. We hence tailor the NLP to the current situation. The function is defined as

$$\phi_{\text{COLREGs}}(\boldsymbol{w}) = \sum_{k=1}^{N_p} \left[\sum_{i \in \mathcal{O}_{\text{HO}}} K_{\text{HO}} V_{\text{HO},i,k}(\boldsymbol{p}_k) + \sum_{i \in \mathcal{O}_{\text{GW}}} K_{\text{GW}} V_{\text{GW},i,k}(\boldsymbol{p}_k) \right. $$
$$\left. + \sum_{i \in \mathcal{O}_{\text{SO}}} K_{\text{SO}} V_{\text{SO},k}(\boldsymbol{w}) + \sum_{i \in \mathcal{O}_{\text{EM}}} K_{\text{EM}} V_{\text{EM},k}(\boldsymbol{w}) \right], \quad (23)$$

where $\mathcal{O}_{HO}, \mathcal{O}_{GW}, \mathcal{O}_{SO},$ and $\mathcal{O}_{EM}$ contain obstacles which are in the head-on, give-way, stand-on and emergency states, respectively, and $K_{HO}, K_{GW}, K_{SO}, K_{EM} > 0$ are tuning parameters. The functions $V_{HO,i,k}(\boldsymbol{p}_k)$, $V_{GW,i,k}(\boldsymbol{p}_k)$, $V_{SO,k}(\boldsymbol{w})$, and $V_{EM,k}(\boldsymbol{w})$ describe functions capturing head-on, give-way, stand-on and emergency behavior with respect to obstacle i, respectively. Notice that the head-on and give-way functions vary with both the obstacle number and time step number, which is due to the functions depending on the given obstacles position and course at time step k.

For head-on situations, we define a potential function with a positive value on the obstacle's starboard side, and a negative value on its port side. When used in the objective function, this will favor trajectories passing a head-on obstacle on its port side, in compliance with Rule 14 of the COLREGs. In addition, the potential function has an attenuation term, reducing the impact of the function when far away from an obstacle:

$$V_{HO,i,k}(\boldsymbol{p}) = \frac{\tanh\left(\alpha_{x,HO}(x_{0,HO} - x^{\{i,k\}})\right)}{2} \tanh(\alpha_{y,HO}y^{\{i,k\}}) \in (-1, 1),$$
$$(24)$$

where $\alpha_{x,HO}, \alpha_{y,HO} > 0$ are tuning parameters controlling the steepness of the head-on potential function and $\bar{x}_{0,HO} > 0$ is a tuning parameter controlling the influence of the attenuating potential. The coordinate $(x^{\{i,k\}}, y^{\{i,k\}})$ is $\boldsymbol{p}$ given in obstacles i's course-fixed frame (in which the x-axis points along the obstacle's course) at time step k, computed as

$$\begin{bmatrix} x^{\{i,k\}} \\ y^{\{i,k\}} \end{bmatrix} = \boldsymbol{R}(\chi_{i,k})^{\top}\left(\boldsymbol{p} - \boldsymbol{p}_{o,k,i}\right), \qquad (25)$$

where $\boldsymbol{p}_{o,k,i}$ and $\chi_{i,k}$ are the position and course of obstacle i at time step k. The head-on potential function with parameters $\alpha_{x,HO} = 1/500$, $\alpha_{y,HO} = 1/400$ and $x_{0,HO} = 1,000$ m is shown in **Figure 6A**.

For give-way situations, we define a similar potential function, but rotated such that the function is positive in front of an obstacle and negative behind it. This will favor trajectories passing behind an obstacle, as desirable with respect to Rule 15 when a give-way obligation is active. The give-way potential function is defined as

$$V_{GW,i,k}(\boldsymbol{p}) = \frac{\tanh\left(\alpha_{y,GW}(y^{\{i,k\}} - y_{0,GW})\right)}{2} \tanh(\alpha_{x,GW}x^{\{i,k\}}) \in (-1, 1),$$
$$(26)$$

where $\alpha_{x,GW}, \alpha_{y,GW} > 0$ control the steepness of the give-way potential function and $\bar{y}_{0,GW} < 0$ control the attenuation on the port side of an obstacle. The give-way potential function with parameters $\alpha_{x,GW} = 1/400$, $\alpha_{y,GW} = 1/500$ and $y_{0,GW} = -500$ m is shown in **Figure 6B**.

In stand-on situations, we want the mid-level algorithm to disregard the obstacle and keep the current speed and course
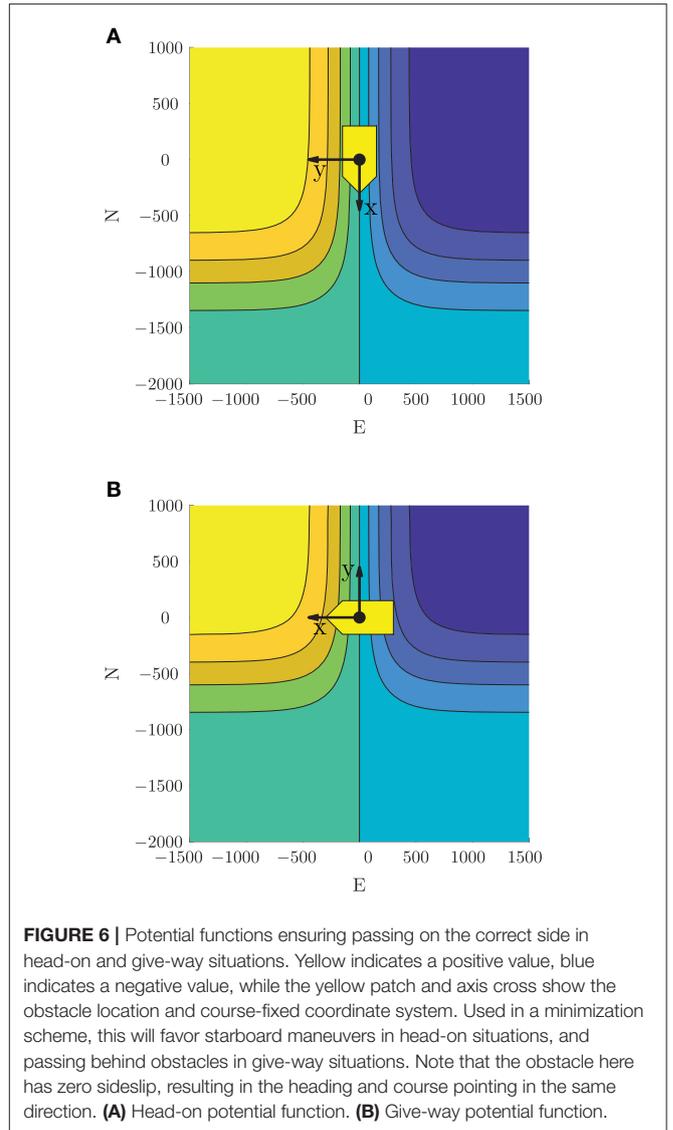
FIGURE 6 | Potential functions ensuring passing on the correct side in head-on and give-way situations. Yellow indicates a positive value, blue indicates a negative value, while the yellow patch and axis cross show the obstacle location and course-fixed coordinate system. Used in a minimization scheme, this will favor starboard maneuvers in head-on situations, and passing behind obstacles in give-way situations. Note that the obstacle here has zero sideslip, resulting in the heading and course pointing in the same direction. **(A)** Head-on potential function. **(B)** Give-way potential function.

in order to comply with the first part of Rule 17. One could simply constrain the algorithm to not maneuver, but this would be perilous in situations where the ownship simultaneously finds itself in a head-on or give-way situation. In such a situation it would be of extra importance to choose readily observable maneuvers, and we therefore design the stand-on cost with the same terms as used in the control cost (22) to amplify the effect:

$$V_{SO,k}(\boldsymbol{w}) = K_{\dot{U}}q_{\dot{U}}(\dot{U}_k) + K_{\dot{\chi}}q_{\dot{\chi}}(\dot{\chi}_k). \qquad (27)$$

If an obstacle is in an emergency state, the obstacle is disregarded in the mid-level algorithm and left for the short-term algorithm to handle. In such a situation, it is important that the mid-level algorithm behaves predictable, and we therefore use the same cost function as for stand-on situations:

$$V_{EM,k}(\boldsymbol{w}) = V_{SO,k}(\boldsymbol{w}). \qquad (28)$$

The slack variable $\boldsymbol{\xi}$ is used in a homotopy scheme, which we introduce to avoid getting trapped in local minima around moving obstacles. The homotopy scheme is described in further detail in section 4.5. The homotopy cost function $\phi_\xi(\boldsymbol{\xi})$ introduces slack cost on $\boldsymbol{\xi}$:

$$\phi_\xi(\boldsymbol{\xi}) = K_\xi \mathbf{1}^\top \boldsymbol{\xi}, \qquad (29)$$

where $K_\xi > 0$ is iteratively increased as part of the homotopy scheme.

4.5. Obstacle Handling and Steady-State Feasibility

The inequality constraint $\boldsymbol{h}(\boldsymbol{w}, \boldsymbol{\xi}) \leq \mathbf{0}$ ensures COLAV and steady-state feasibility with respect to actuator limitations.

Static obstacles are handled similarly as in the high-level algorithm, with (4) representing an elliptical obstacle with center (x_c, y_c), angle α and major and minor axes x_a and y_a, respectively. The constraint (4) needs to be enforced at each time step. Hence, for the i-th static obstacle, we define the constraint

$$\boldsymbol{h}_{s_i}(\boldsymbol{w}) = \begin{bmatrix} h_o(x_1, y_1, x_{c,i}, y_{c,i}, x_{a,i}, y_{a,i}, \alpha_i) \\ h_o(x_2, y_2, x_{c,i}, y_{c,i}, x_{a,i}, y_{a,i}, \alpha_i) \\ \vdots \\ h_o(x_{N_p}, y_{N_p}, x_{c,i}, y_{c,i}, x_{a,i}, y_{a,i}, \alpha_i) \end{bmatrix} \leq \mathbf{0}. \qquad (30)$$

Moving obstacles are handled in a similar fashion, but letting the ellipsis center position and angle be time varying. Obstacles in stand-on situations should, however, not be included in the constraints, since the mid-level algorithm is supposed to stand on in such situations. Moreover, if an obstacle has entered an emergency state, the obstacle is so close and behaving unpredictably that the mid-level algorithm should disregard it and leave it for the short-term layer. Hence, for the i-th moving obstacle not in a stand-on or an emergency situation, we define the constraint

$$\boldsymbol{h}_{m_i}(\boldsymbol{w}) = \begin{bmatrix} h_o(x_1, y_1, x_{c,i,1}, y_{c,i,1}, x_{a,i}, y_{a,i}, \alpha_{i,1}) \\ \vdots \\ h_o(x_{N_p}, y_{N_p}, x_{c,i,N_p}, y_{c,i,N_p}, x_{a,i}, y_{a,i}, \alpha_{i,N_p}) \end{bmatrix} \leq \mathbf{0}, \quad (31)$$

where $x_{c,i,k}, y_{c,i,k}$, and $\alpha_{i,k}$ denote the position and course of the i-th moving obstacle at time step k.

Given D static obstacles and M obstacles not in stand-on or emergency situations, we define the constraint

$$\boldsymbol{h}_o(\boldsymbol{w}, \boldsymbol{\xi}) = \begin{bmatrix} \boldsymbol{h}_{s_1}(\boldsymbol{w}) \\ \vdots \\ \boldsymbol{h}_{s_D}(\boldsymbol{w}) \\ \boldsymbol{h}_{m_1}(\boldsymbol{w}) \\ \vdots \\ \boldsymbol{h}_{m_M}(\boldsymbol{w}) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\xi} \end{bmatrix}, \qquad (32)$$

where we include slack variables $\boldsymbol{\xi} \geq \mathbf{0}$ on the moving obstacle constraints as part of a homotopy scheme. The reason for using

homotopy is that NLP solvers in general only finds local minima, and can have issues with moving an initial guess "through" obstacles. Normally, this is not an issue, but for the mid-level algorithm the optimal solution can change drastically from one iteration to another. This can for instance happen if an obstacle enters a head-on or give-way state, where the solution can be trapped on the wrong side of an obstacle. In general, homotopy describes introducing an extra parameter which is iteratively adjusted in order to iteratively move a local solution toward a global solution (Deuflhard, 2011). In our homotopy scheme, we introduce slack variables on the moving obstacle constraints, which will allow solutions to travel through obstacles at the cost of a homotopy cost (29) scaled by the homotopy parameter K_ξ. Initially, this is selected as a low value to have a high amount of slack on the moving obstacles, while it is iteratively increased toward $K_\xi \to \infty$, which results in $\boldsymbol{\xi} = \mathbf{0}$ and hence no slack on moving obstacles. Currently, we only introduce slack on moving obstacles, but slack should also be introduced to static obstacles if they are small enough for the algorithm to be able to pass on both sides, like e.g., rocks, navigational marks, etc.

Similarly as in Eriksen and Breivik (2017b) and Bitar et al. (2019a), we ensure steady-state feasible trajectories at each time step through a constraint $\boldsymbol{h}_{x_{r,k}}(\boldsymbol{x}_{r,k}) \leq \mathbf{0} \in \mathbb{R}^4$, which captures the state constraint $\boldsymbol{x}_r \in \mathcal{X}_r$ at time step k. To ensure stead-state feasibility for the entire prediction horizon, we define the constraint

$$\boldsymbol{h}_{x_r}(\boldsymbol{w}) = \begin{bmatrix} \boldsymbol{h}_{x_{r,k}}(\boldsymbol{x}_{r,0}) \\ \boldsymbol{h}_{x_{r,k}}(\boldsymbol{x}_{r,1}) \\ \vdots \\ \boldsymbol{h}_{x_{r,k}}(\boldsymbol{x}_{r,N_p-1}) \end{bmatrix} \leq \mathbf{0}. \qquad (33)$$

Finally, the inequality constraints are combined as.

$$\boldsymbol{h}(\boldsymbol{w}, \boldsymbol{\xi}) = \begin{bmatrix} \boldsymbol{h}_o(\boldsymbol{w}, \boldsymbol{\xi}) \\ \boldsymbol{h}_{x_r}(\boldsymbol{w}) \end{bmatrix} \in \mathbb{R}^{(M+D+4)N_p}. \qquad (34)$$

5. SHORT-TERM COLAV

For the short-term layer, the branching-course model predictive control (BC-MPC) algorithm is used, which is a sample-based MPC algorithm intended for short-term ASV COLAV. The BC-MPC algorithm was initially developed in Eriksen et al. (2019), extended to also consider static obstacles in Eriksen and Breivik (2019) and is experimentally validated in several full-scale experiments using a radar-based system for detecting and tracking obstacles. The algorithm complies with COLREGs rules 8, 13, and the second part of Rule 17, while favoring maneuvers complying with the maneuvering aspects of rules 14 and 15. Notice that Rule 17 allows a ship to ignore the maneuvering aspects of rules 14 and 15 in situations where the give-way vessel does not maneuver. The obstacle clearance will be larger if the algorithm ignores the maneuvering aspects of rules 14 and 15, like e.g., passing in front of an obstacle in a crossing situation where the ownship is the give-way vessel. Moving obstacles are in general handled by the mid-level algorithm, making this applicable only in emergency situations and for

obstacles detected so late that the mid-level algorithm is unable to avoid them.

The algorithm constructs a search space consisting of a finite number of trajectories, which each contain a sequence of maneuvers. The maneuvers are constructed using a dynamic model of the ownship and a set of acceleration motion primitives, resulting in feasible trajectories being specified to the vessel controller. For each maneuver, a discrete set of SOG and course accelerations are created as

$$
\begin{aligned}
\dot{U}_{\text{samples}} &= \left\{ \dot{U}_1, \dot{U}_2, \ldots, \dot{U}_{N_U} \right\} \\
\ddot{\chi}_{\text{samples}} &= \left\{ \ddot{\chi}_1, \ddot{\chi}_2, \ldots, \ddot{\chi}_{N_\chi} \right\},
\end{aligned}
\tag{35}
$$

where $\dot{U}_i, i \in [1, N_U]$ and $\ddot{\chi}_i, i \in [1, N_\chi]$ denote $N_U \in \mathbb{N}$ and $N_\chi \in \mathbb{N}$ vessel-feasible speed and course accelerations. Given the acceleration samples (35) and motion primitives for each maneuver in a trajectory, we create a set of desired SOG and course trajectories $\mathcal{U}_d$. These trajectories have continuous acceleration, and is designed in an open-loop fashion by using the current reference tracked by the vessel controller for initialization, rather than the current vessel SOG and course. The reason for this is that the reference to the vessel controller should be continuous in order to avoid jumps in the actuator commands. To include feedback in the trajectory prediction, a set of feedback-corrected SOG and course trajectories $\bar{\mathcal{U}}_d$ is predicted using a simplified error model of the vessel and vessel controller. Finally, the feedback-corrected SOG and course trajectories are used to compute a set of feedback-corrected pose trajectories:

$$
\bar{\mathcal{H}} = \left\{ \bar{\boldsymbol{\eta}}(\cdot) \middle| (\bar{U}(\cdot), \bar{\chi}(\cdot)) \in \bar{\mathcal{U}} \right\},
\tag{36}
$$

where $\bar{\boldsymbol{\eta}}(\cdot)$ denotes a kinematic simulation procedure that given SOG and course trajectories, $\bar{U}(\cdot)$ and $\bar{\chi}(\cdot)$, in $\bar{\mathcal{U}}_d$ computes the vessel pose. See Eriksen and Breivik (2019) and Eriksen et al. (2019) for more details on the trajectory generation procedure.

In order to converge toward the trajectory specified by the mid-level algorithm, a desired acceleration is computed based on a line-of-sight guidance scheme. In Eriksen and Breivik (2019) and Eriksen et al. (2019), the samples closest to the desired acceleration in (35) are replaced with the desired acceleration, given that this is vessel-feasible. A problem with this, is that when operating at high speeds, the possible acceleration may not be symmetric, resulting in that zero acceleration (hence keeping a constant speed and course), may not be part of the search space. This can cause undesirable behavior, since the BC-MPC algorithm will be unable to keep the speed and course constant, which can cause oscillatory behavior. In this paper, we therefore propose to move the acceleration samples closest to zero, and adding the desired acceleration as a separate sample, given that it is vessel feasible. This will make sure that keeping a constant speed and course, as well as a trajectory converging toward the desired trajectory is included in the search space.

Given the predicted trajectories, the algorithm finds the optimal desired SOG and course trajectory for the vessel controller $\boldsymbol{u}_d^*(\cdot) = [U_d(\cdot)^*, \chi_d(\cdot)^*]$ as

$$
\boldsymbol{u}_d^*(\cdot) = \underset{(\bar{\boldsymbol{\eta}}_k(\cdot), \boldsymbol{u}_{d,k}(\cdot)) \in (\bar{\mathcal{H}}, \bar{\mathcal{U}}_d)}{\text{argmin}} G(\bar{\boldsymbol{\eta}}_k(\cdot), \boldsymbol{u}_{d,k}(\cdot); \boldsymbol{p}_d^{\text{mid}}(\cdot)),
\tag{37}
$$

where the objective function is given as

$$
\begin{aligned}
G(\bar{\boldsymbol{\eta}}(\cdot), \boldsymbol{u}_d(\cdot); \boldsymbol{p}_d^{\text{mid}}(\cdot)) ={}& w_{\text{al}}\text{align}(\bar{\boldsymbol{\eta}}(\cdot); \boldsymbol{p}_d^{\text{mid}}(\cdot)) \\
&+ w_{\text{av,m}}\text{avoid}_{\text{m}}(\bar{\boldsymbol{\eta}}(\cdot)) + w_{\text{av,s}}\text{avoid}_{\text{s}}(\bar{\boldsymbol{\eta}}(\cdot)) \\
&+ w_{\text{t},U}\text{tran}_U(\boldsymbol{u}_d(\cdot)) + w_{\text{t},\chi}\text{tran}_\chi(\boldsymbol{u}_d(\cdot)).
\end{aligned}
\tag{38}
$$

The variables $w_{\text{al}}, w_{\text{av,m}}, w_{\text{av,s}}, w_{\text{t},U}, w_{\text{t},\chi} > 0$ are tuning parameters, while $\text{align}(\bar{\boldsymbol{\eta}}(\cdot); \boldsymbol{p}_d^{\text{mid}}(\cdot))$ measures the alignment between a candidate trajectory $\bar{\boldsymbol{\eta}}(\cdot)$ and the desired trajectory from the mid-level algorithm $\boldsymbol{p}_d^{\text{mid}}(\cdot)$. The function $\text{avoid}_{\text{m}}(\bar{\boldsymbol{\eta}}(\cdot))$ ensures COLAV of moving obstacles by penalizing trajectories close to obstacles, using a non-symmetric obstacle ship domain designed with the COLREGs in mind. The function $\text{avoid}_{\text{s}}(\bar{\boldsymbol{\eta}}(\cdot))$ ensures COLAV of static obstacles by introducing an occupancy grid, while $\text{tran}_U(\boldsymbol{u}_d(\cdot))$ and $\text{tran}_\chi(\boldsymbol{u}_d(\cdot))$ introduces transitional costs to avoid shattering. The transitional terms penalize deviations from the planned trajectory of the previous iteration, unless changing to the trajectory corresponding by the desired acceleration. See Eriksen and Breivik (2019) and Eriksen et al. (2019) for more details and descriptions of the terms.

6. SIMULATION RESULTS

The hybrid COLAV system is verified through simulations, which are present in this section. The simulations include ocean current and both static and moving obstacles. We include moving obstacles both acting in compliance with the COLREGs, and violating the COLREGs.

6.1. Simulation Setup

The simulations are performed in MATLAB on a computer with an 2.8 GHz Intel Core i7 processor running macOS Mojave, using CasADi (Andersson et al., 2019) and IPOPT (Wächter and Biegler, 2005) for implementing the high-level and mid-level algorithms. The simulator is built upon the mathematical model of the Telemetron ASV described in section 2, and the model-based speed and course controller in Eriksen and Breivik (2018) is used as the vessel controller.

The parameters of the high-level algorithm are listed in **Table 1**. The number of prediction steps N_{hi} is chosen to achieve a time step length $h = {}^{t_{\max}}\!/_{N_{\text{hi}}} < 1.5$ s, which seems to be a good compromise between capturing the relevant system dynamics and having a feasible computational requirement.

The mid-level algorithm is implemented using the parameters in **Table 2**.

The slack variable cost K_ξ has five elements, implying that we use five steps in our homotopy scheme. The mid-level NLP is initially warm started with the solution from the previous iteration, while each step in the homotopy scheme is warm started with the solution from the previous step of the homotopy scheme, converging toward the solution without slack on the constraints. To reduce the computational load and increase the predictability of the mid-level algorithm, we utilize six steps of each planned mid-level trajectory, only running the mid-level algorithm every 60 s. This implies that six steps of the predicted

TABLE 1 | Tuning parameters for the high-level algorithm.

Param.	Value	Comment
$t_{\max}$		Maximum trajectory time
Scenario 1	1420 s	
Scenario 2	1420 s	
Scenario 3	725 s	
N_{hi}	1000	Number of prediction steps
K_e	1.0 s^3/m	Energy penalty gain
K_δ	1.0	Quadratic yaw control penalty gain
L_m	4.0 m	Length between control origin and outboard motor

TABLE 2 | Tuning parameters for the mid-level algorithm.

Param.	Value	Comment
$\bar{d}_{CPA}^{enter}$	900 m	State machine d_{CPA} entry criteria
$\underline{d}_{CPA}^{exit}$	2000 m	State machine d_{CPA} exit criteria
$[\underline{t}_{CPA}^{enter}, \bar{t}_{CPA}^{enter}]$	[0, 270] s	State machine t_{CPA} entry criteria
$[\underline{t}_{CPA}^{exit}, \bar{t}_{CPA}^{exit}]$	[-20, 290] s	State machine t_{CPA} exit criteria
$t_{crit}^{EM,enter}$	20 s	Emergency state t_{crit} entry criteria
$t_{crit}^{EM,exit}$	25 s	Emergency state t_{crit} exit criteria
h	10 s	Step size
N_p	36	Number of prediction steps
K_p	0.02	Position error scaling
σ	1	Huber loss function threshold
$K_{\dot{U}}$	0.3	SOG-derivative penalty term scaling
$K_{\dot{\chi}}$	2.5	Course-derivative penalty term scaling
K_{HO}	40	Head-on potential function scaling
$[\alpha_{x,HO}, \alpha_{y,HO}]$	[1/500, 1/400]	Head-on potential function steepness parameters
$x_{0,HO}$	1000 m	Head-on potential function attenuation parameter
K_{GW}	40	Give-way potential function scaling
$[\alpha_{x,GW}, \alpha_{y,GW}]$	[1/400, 1/500]	Give-way potential function steepness parameters
$y_{0,GW}$	−500 m	Give-way potential function attenuation parameter
K_{SO}	3	Stand-on function scaling
K_{EM}	3	Emergency function scaling
K_ξ	[0.1, 1, 10, 100, ∞]	Iterative slack variable cost
x_a	600 m	Moving obstacle ellipsis major axis size
y_a	225 m	Moving obstacle ellipsis minor axis size

solution will be implemented before computing a new solution, which further implies that the state machine is also only run every 60 s. If the mid-level algorithm fails in finding a feasible solution, the algorithm will re-use the solution from the last iteration. This may for instance happen if the algorithm tries to compute a solution while being inside a moving obstacle ellipse, which sometimes can be the case when an obstacle is exiting an emergency or stand-on state. The BC-MPC algorithm is run every 5 s, with parameters as described in Eriksen and

Breivik (2019). An update rate of 5 s is considered sufficient due the typically large maneuvering margins at sea. It is also worth noting that the detection and tracking system can represent a significant time delay, especially for radar-based systems (Eriksen et al., 2019). For confined and congested areas the BC-MPC algorithm may need to be run at a higher rate, which also imposes requirements for high-bandwidth obstacle estimates. Static obstacles are padded with a safety margin of 150 m for the high-level and mid-level algorithms, while the BC-MPC algorithm uses a safety margin of 100 m for static obstacles. The reason for having a smaller static obstacle safety margin for the BC-MPC algorithm is that it tends to struggle with following trajectories on the static obstacle boundaries. The BC-MPC algorithm would hence not be able to follow the nominal trajectory if the static obstacle safety margin was the same as for the mid-level and high-level algorithms.

The simulations are performed without any noise on the obstacle estimates, providing the algorithms with exact information about the obstacles position, course, and speed. The BC-MPC algorithm has previously been shown to perform well with noisy and uncertain obstacle estimates in full-scale experiments using radar-based detection and tracking of obstacles (Eriksen and Breivik, 2019; Eriksen et al., 2019). The mid-level algorithm is likely to have a larger requirement to low noise levels on the obstacle estimates, since the state machine in the mid-level algorithm depends on logic and discrete switching. However, the algorithm is also run less frequently, reducing the required bandwidth of the obstacle estimates, possibly allowing using smoothing or tracking filters with a lower process noise if necessary. It may also be feasible to make the mid-level algorithm depend on data from the automatic identification system, which typically have much lower noise levels than radar-based tracking systems, while being subject to robustness issues (Harati-Mokhtari et al., 2007).

We present three scenarios, which demonstrate different important properties of the hybrid COLAV system:
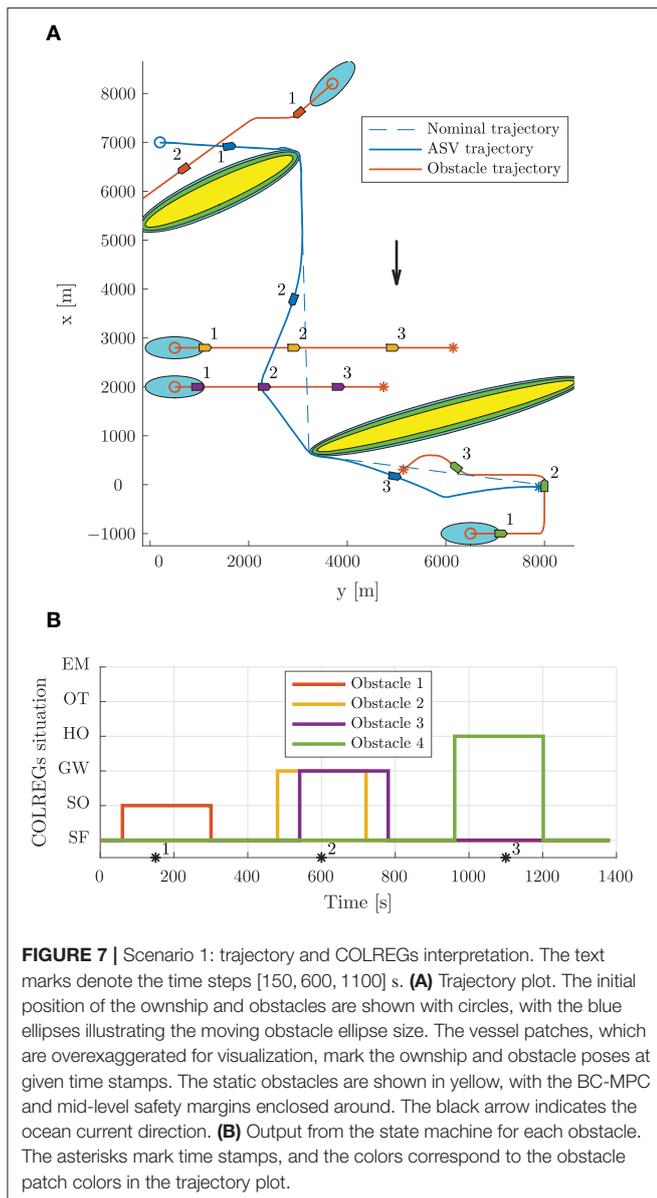
Scenario 1 This scenario contains two static obstacles, and four moving obstacles of which all comply with the COLREGs. The moving obstacles demonstrate stand-on, give-way and head-on situations.

Scenario 2 This scenario contains one static and five moving obstacles. The moving obstacles demonstrate stand-on with an obstacle ignoring the COLREGs, an overtaking and a simultaneous head-on, give-way and stand-on situation with obstacles complying with the COLREGs.

Scenario 3 This scenario contains two moving obstacles, which suddenly perform dangerous maneuvers close to the ownship, displaying the use of the emergency state.

6.2. Scenario 1

Scenario 1 contains two static obstacles, four moving obstacles, an ocean current of $[-2, 0]^\top$ m/s and is shown in **Figure 7**. The high-level planner plans a nominal trajectory between the

FIGURE 7 | Scenario 1: trajectory and COLREGs interpretation. The text marks denote the time steps [150, 600, 1100] s. **(A)** Trajectory plot. The initial position of the ownship and obstacles are shown with circles, with the blue ellipses illustrating the moving obstacle ellipse size. The vessel patches, which are overexaggerated for visualization, mark the ownship and obstacle poses at given time stamps. The static obstacles are shown in yellow, with the BC-MPC and mid-level safety margins enclosed around. The black arrow indicates the ocean current direction. **(B)** Output from the state machine for each obstacle. The asterisks mark time stamps, and the colors correspond to the obstacle patch colors in the trajectory plot.

FIGURE 8 | Scenario 1: speed and angular trajectories. The asterisks mark the same time samples as in **Figure 7**. **(A)** Speed trajectories. **(B)** Angular trajectories.

initial and goal positions at $[7000, 200]^\top$ m and $[0, 7900]^\top$ m, respectively. The first obstacle is in a stand-on situation, where it is required to maneuver in order to avoid collision with the ownship, which is required to stand on. As shown in **Figure 7B**, the first obstacle is quickly considered as a stand-on situation, at which the mid-level algorithm disregards the obstacle and continues with the current speed and course. Following this, the obstacle maneuvers in accordance to the COLREGs, and we avoid collision. After the first static obstacle, we encounter two crossing vessels where the ownship is deemed the give-way vessel. In accordance with the COLREGs, we maneuver to starboard in order to pass behind both obstacles. Notice that the second give-way obstacle is detected as a give-way situation later than the first, since the entry criteria in the state machine includes the time to CPA, which is higher for the second give-way obstacle.
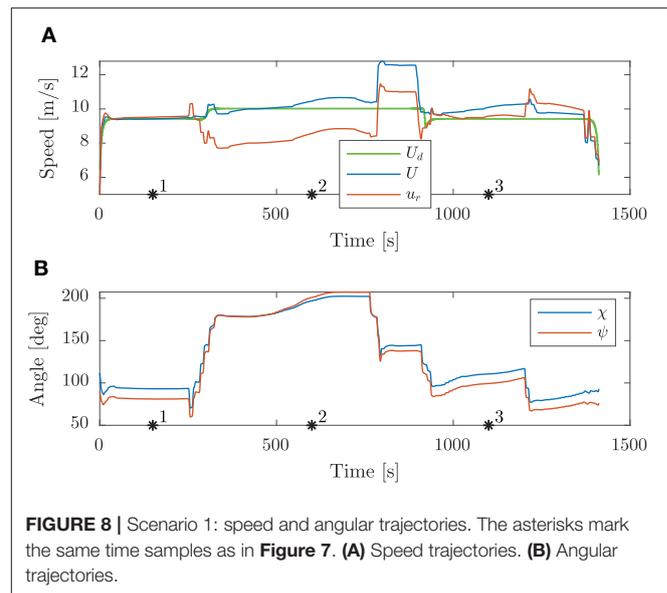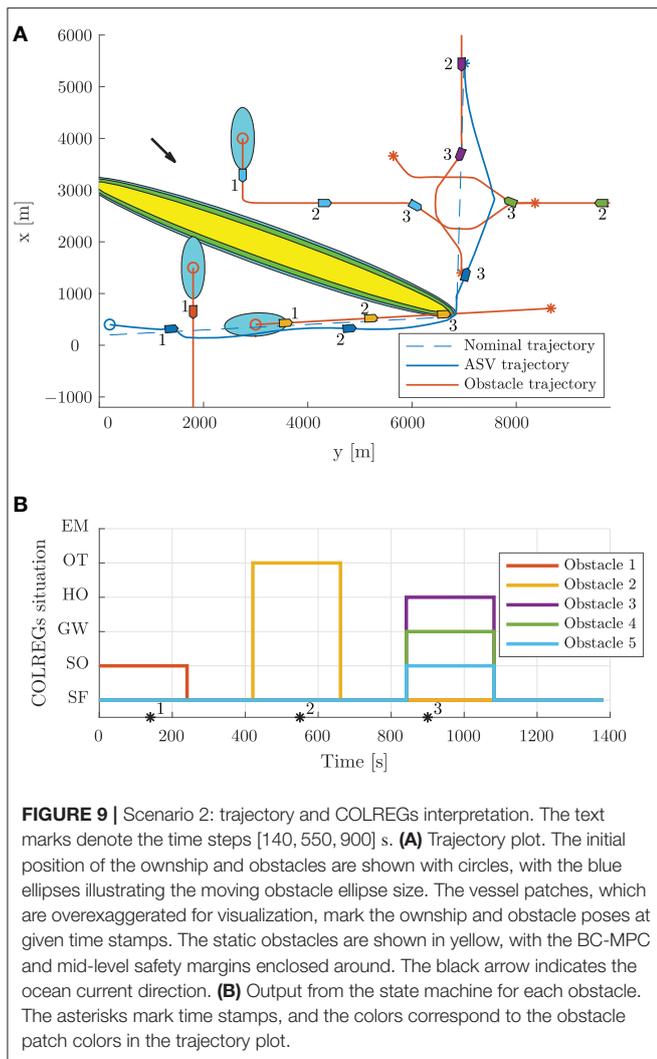
After avoiding the two give-way obstacles, we converge toward the nominal trajectory and encounter a head-on situation. This is correctly identified by the state machine as head on, and we maneuver to starboard in order to avoid collision. Notice that even though the obstacle maneuvers, we keep the obstacle in the head-on state until we have passed it.
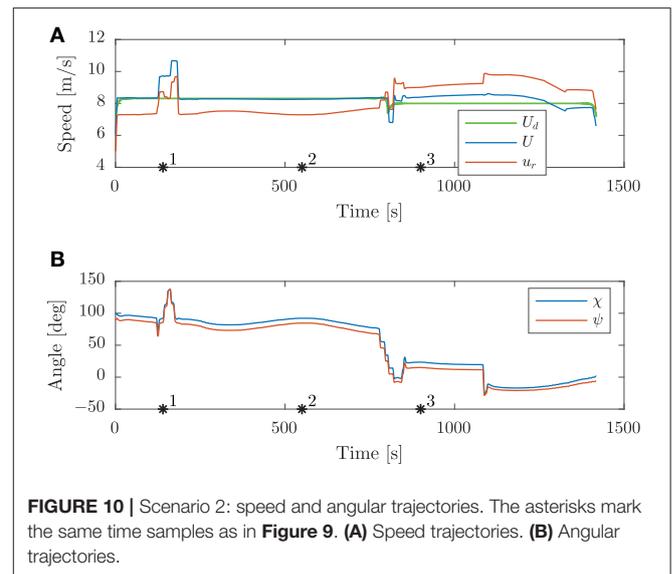
Figure 8 shows the speed and angular trajectories during Scenario 1, where the desired speed is calculated as the nominal speed at the closest point on the nominal trajectory given the ownship position. From this, we see that the mid-level and BC-MPC algorithms manage to track the desired nominal speed before and after the first static obstacle, where no obstacles require maneuvering away from the nominal trajectory. Notice that when encountering the two crossing obstacles, the mid-level algorithm chooses to slowly change the course, which is due to the attenuation of the give-way potential function and the large distance between the vessels. It would be better to make a clear course change, which is a subject of tuning. After passing the two crossing obstacles, the mid-level algorithm increases the speed in order to get back to the nominal trajectory, which is due to the algorithm attempting to keep the speed projected on the nominal trajectory equal as the desired nominal speed. Furthermore, notice that the mid-level algorithm actively controls the relative surge speed in order achieve the desired SOG, which is clearly seen when passing the first static obstacle.

6.3. Scenario 2
Scenario 2, shown in **Figure 9**, is more complex than Scenario 1, with a total of five moving obstacles, and has an ocean current of $[-1, 1]^\top$ m/s. The high-level planner plans a nominal trajectory between the initial and goal positions at $[200, 200]^\top$ m and $[5500, 7000]^\top$ m, respectively. The first obstacle is a crossing vessel, which similarly as in Scenario 1 is deemed to give way for the ownship, which should keep the current speed and course. However, in this scenario, the obstacle violates the COLREGs by

FIGURE 9 | Scenario 2: trajectory and COLREGs interpretation. The text marks denote the time steps [140, 550, 900] s. **(A)** Trajectory plot. The initial position of the ownship and obstacles are shown with circles, with the blue ellipses illustrating the moving obstacle ellipse size. The vessel patches, which are overexaggerated for visualization, mark the ownship and obstacle poses at given time stamps. The static obstacles are shown in yellow, with the BC-MPC and mid-level safety margins enclosed around. The black arrow indicates the ocean current direction. **(B)** Output from the state machine for each obstacle. The asterisks mark time stamps, and the colors correspond to the obstacle patch colors in the trajectory plot.

FIGURE 10 | Scenario 2: speed and angular trajectories. The asterisks mark the same time samples as in **Figure 9**. **(A)** Speed trajectories. **(B)** Angular trajectories.
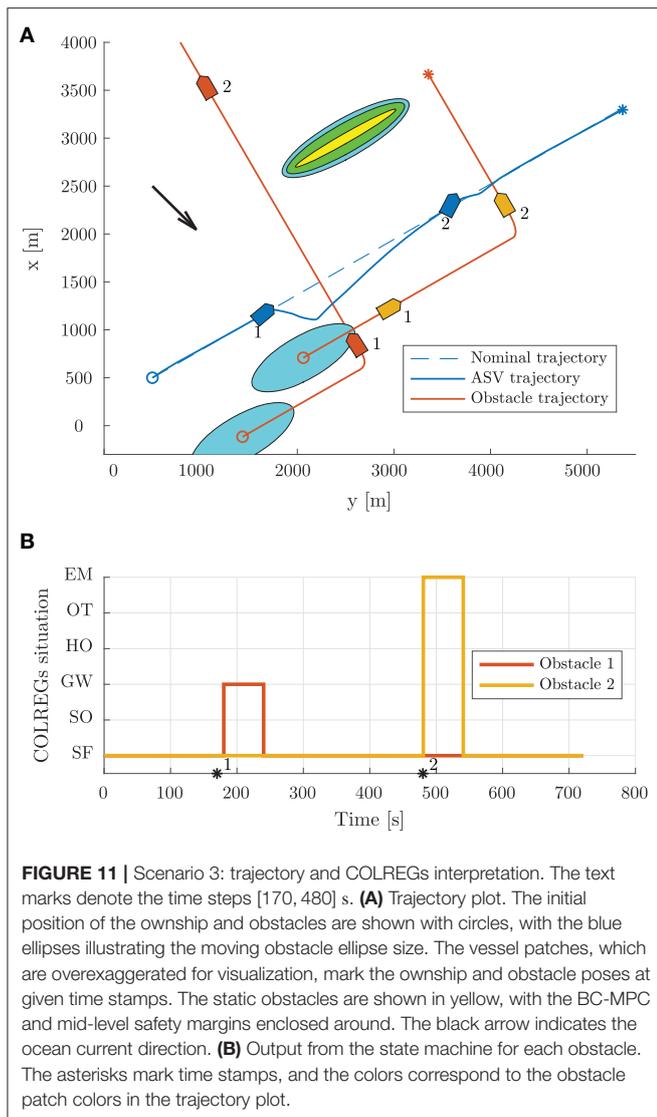
vessel crossing from starboard. The mid-level algorithm solves this situation with the desirable behavior, and converges toward the nominal trajectory after the situation is resolved. As shown in **Figure 9B**, the state machine interprets the situations correctly.

From the speed trajectory in **Figure 10** it is clear that the mid-level algorithm follows the desired nominal speed also when overtaking the second obstacle.

6.4. Scenario 3

Scenario 3, shown in **Figure 11**, contains two moving obstacles on parallel courses with the ownship, and has an ocean current of $[-1, 1]^\top$ m/s. The high-level planner plans a nominal trajectory between the initial and goal positions at $[500, 500]^\top$ m and $[3328, 5399]^\top$ m, respectively, which results in a straight line trajectory with a course angle of 60°. The first obstacle travels at a higher speed than the ownship, while the second one travels at a lower speed and will be overtaken by the ownship. Since the obstacles are on parallel paths with the obstacle, the time to CPA is sufficiently high such that the obstacles are in the safe state, even though the vessels are quite close. However, both obstacles make sudden maneuvers to port dangerously close to the ownship and enters on a crossing course with the ownship. With respect to the COLREGs, the ownship is required to give way to both obstacles since they are crossing from the ownship's starboard side. One can, however, argue that the maneuvers displayed by the obstacles are dangerous and displays poor seamanship, such that the ownship should not be held accountable if a collision occurred. Nevertheless, the hybrid COLAV system manages to avoid both obstacles. As seen in **Figure 11B**, the first obstacle is sufficiently far away from the ownship to be considered as a give-way situation when the state machine interprets the situation, and the mid-level algorithm plans a trajectory passing behind the first obstacle. The second obstacle maneuvers to port even closer to the ownship, resulting in the distance to the critical point being within the threshold

not maneuvering in order to avoid collision. Therefore, the BC-MPC algorithm maneuvers to avoid collision when the obstacle gets so close that the safety margins of the BC-MPC algorithms is violated. The BC-MPC algorithm maneuvers to port, as advised by COLREGs Rule 17 for crossing situations where the stand-on vessel has to maneuver, and safely avoid the first obstacle. The second obstacle is overtaken by the ownship, and correctly considered as an overtaking situation by the state machine. For such an situation, there is no requirement on how the ownship should maneuver, except keeping clear from the overtaken vessel. After passing the second obstacle, we encounter a complex situation with simultaneous head-on, give-way and stand-on obligations. In this situation, each vessel, including the ownship, finds itself in a situation where a head-on and a give-way situation require starboard maneuvers, while a stand-on situation requires the vessel to keep the current speed and course. However, head-on and give-way obligations should be prioritized higher than stand-on situations, and the situation is quite easily solved by each vessel maneuvering to starboard and passing behind the

FIGURE 12 | Scenario 3: speed and angular trajectories. The asterisks mark the same time samples as in **Figure 11**. **(A)** Speed trajectories. **(B)** Angular trajectories.

TABLE 3 | Minimum distance to static and moving obstacles for the simulation scenarios.

Scenario	Minimum distance to static obstacles (m)	Minimum distance to moving obstacle number (m)				
		1	**2**	**3**	**4**	**5**
Scenario 1	93.7	634.3	596.3	522.7	726.8	–
Scenario 2	118.2	185.5	228.3	1,097.2	575.6	842.3
Scenario 3	1123.8	326.4	106.6	–	–	–

FIGURE 11 | Scenario 3: trajectory and COLREGs interpretation. The text marks denote the time steps [170, 480] s. **(A)** Trajectory plot. The initial position of the ownship and obstacles are shown with circles, with the blue ellipses illustrating the moving obstacle ellipse size. The vessel patches, which are overexaggerated for visualization, mark the ownship and obstacle poses at given time stamps. The static obstacles are shown in yellow, with the BC-MPC and mid-level safety margins enclosed around. The black arrow indicates the ocean current direction. **(B)** Output from the state machine for each obstacle. The asterisks mark time stamps, and the colors correspond to the obstacle patch colors in the trajectory plot.

for entering the emergency situation when the state machine interprets the situation. In this situation, the mid-level algorithm disregards the obstacle and leaves it to the BC-MPC algorithm to avoid collision.

As seen in **Figure 12**, the mid-level algorithm both reduces the speed and changes the course to avoid the first obstacle. When approaching the second obstacle, the BC-MPC algorithm initiates a speed reduction, and after some time also maneuver to starboard in order to pass behind the obstacle and resolve the situation.

6.5. Simulation Summary
The simulation results show that the hybrid COLAV system is able to handle a wide range of situations, while also behaving in an energy-optimal way when moving obstacles are not interfering with the ownship trajectory. **Table 3** shows the minimum distance to static and moving obstacles for the scenarios.
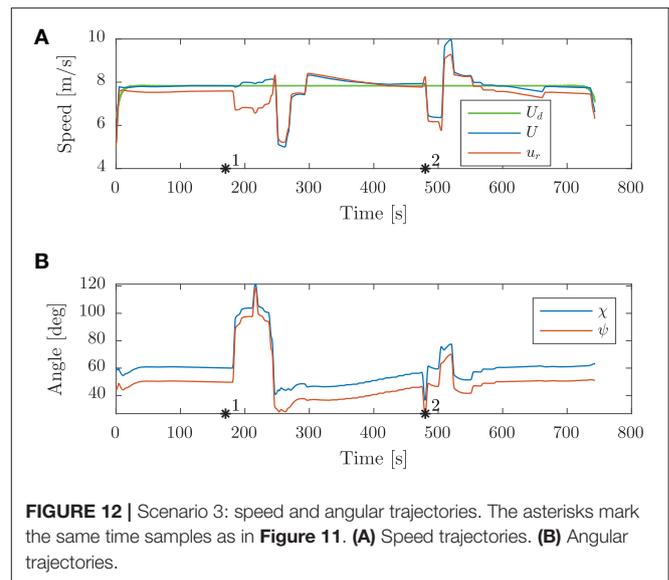
The minimum distance to static obstacles is in Scenario 1 below the safety region size of the BC-MPC algorithm, which is intentional and caused by the algorithm using a smooth penalty function for interpreting static obstacles. The penalty function value increases linearly when moving further into the safety region, see Eriksen and Breivik (2019) for more details. The minimum distance to moving obstacles is a bit difficult to interpret, since the obstacle ship domains are non-circular, implying that the required clearance depends on relative position of the ownship with respect to the moving obstacles. However, we see that we have a larger clearance in head-on, give-way and stand-on situations where the obstacles comply with the COLREGs, and do not perform dangerous maneuvers (as in Scenario 3), compared to overtaking situations. The reason for this is that when overtaking (obstacle 2 in Scenario 2), we pass the obstacle on a parallel course, resulting in the minor axis of the moving obstacle ellipse indicating the required clearance. Furthermore, we see that obstacle 1 in Scenario 2, which ignores its give-way obligation, comes significantly closer than other crossing obstacles except for those in Scenario 3. The reason for this is that the BC-MPC algorithm, which handles this situation, has a lower clearance requirement than the mid-level algorithm, which still should be considered as safe. In Scenario 3, the two obstacles display poor seamanship, and behave dangerously. Obstacle 1 is handled by the mid-level algorithm and passed

with a clearance lower than the major axis of the mid-level algorithm, which is caused by the BC-MPC algorithm "cutting the corner." The clearance should still be considered safe since we are behind the obstacle, and the clearance requirements of the BC-MPC algorithm is enforced. Obstacle 2, which is placed in the emergency state and handled by the BC-MPC algorithm, is passed with a clearance of only 106.6 m. This is lower than the clearance to Obstacle 1 in Scenario 2 (which violated its stand-on requirement), and is due to the BC-MPC algorithm having a non-symmetric obstacle ship domain function allowing for a smaller clearance when passing behind an obstacle than in front.

For the three scenarios, the high-level planner used an average of 67 s with a maximum of 93 s to compute the solution. Since the high-level planner is intended to be run off-line, this is well within reasonable limits. The mid-level algorithm used 0.60 s on average, and a maximum of 2.1 s, which we consider to be computationally feasible since the mid-level algorithm only is run every 60 s. The BC-MPC algorithm used 0.29 s on average, and a maximum of 0.63 s, which we also consider to be real-time feasible when the BC-MPC algorithm is run every 5 s. The BC-MPC algorithm is highly parallelizable, which could reduce the BC-MPC runtime by a large magnitude if required. The mid- and high-level algorithms may not return solutions as they are non-convex optimization problems, but the BC-MPC algorithm makes the hybrid COLAV system real-time feasible since it always will find a (potentially sub-optimal) solution.

7. CONCLUSION

In this paper, we have presented a three-layered hybrid COLAV system, compliant with COLREGs rules 8 and 13–17. As part of this, we have further developed the MPC-based mid-level COLAV algorithm in Eriksen and Breivik (2017b) and Bitar et al. (2019a) to comply with COLREGs rules 13–16 and parts of Rule 17, which includes developing a state machine for COLREGs interpretation. The hybrid COLAV system has a well-defined division of labor, including an inherent understanding of COLREGs Rule 17, where the mid-level algorithm obeys stand-on situations, while the BC-MPC algorithm handles situations where give-way vessels do not maneuver.

The hybrid COLAV system is verified through simulations, where we in three scenarios challenge the system with a number of different situations. The scenarios include multi-obstacle situations with multiple simultaneously active COLREGs rules, and situations where obstacles violate the COLREGs. Collision is avoided in all the scenarios, and we show that the

ownship follows an energy-optimized trajectory generated by the high-level planner when moving obstacles do not interfere with this trajectory.

For further work, we suggest to:

- Investigate if using situation-dependent entry and exit criteria parameters in the state machine improves the performance.
- Expand the state machine with the possibility of transitioning from head-on, give-way and overtaking states to the emergency state for situations where obstacles behave dangerously or hostile.
- Develop a methodology for deciding tuning parameters.
- Perform simulations with noisy obstacle estimates to investigate how the state machine and mid-level algorithm respond to this.
- Explore the possibilities for integrating the COLREGs interpretation in the mid-level NLP, relaxing the assumption of the current COLREGs situation being valid for the entire prediction horizon.
- Investigate the possibility of including static obstacles from e.g., ENCs in the high- and mid-level algorithms.
- Simulate scenarios where multiple vessels running the hybrid COLAV system interact with each other.
- Validate the hybrid COLAV system in full-scale experiments.

DATA AVAILABILITY STATEMENT

The datasets generated for this study are available on request to the corresponding author.

AUTHOR CONTRIBUTIONS

The work in this article was the result of a collaboration between B-OE and GB, supervised by MB and AL. The contributions to the mid-level and short-term algorithms are made by B-OE, while the COLREGs interpreter was developed in collaboration between GB and B-OE. GB has implemented the high-level planner, and prepared this for integration with the developed simulator. B-OE has taken lead on writing the paper, in collaboration with GB. MB and AL have provided valuable feedback in the writing process.

FUNDING

REFERENCES

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M. (2019). CasADi: a software framework for nonlinear optimization and optimal control. *Math. Program. Comput.* 11, 1–36. doi: 10.1007/s12532-018-0139-4

Benjamin, M. R., Leonard, J. J., Curcio, J. A., and Newman, P. M. (2006). A method for protocol-based collision avoidance between autonomous marine surface craft. *J. Field Robot.* 23, 333–346. doi: 10.1002/rob.20121

Bitar, G., Breivik, M., and Lekkas, A. M. (2018). "Energy-optimized path planning for autonomous ferries," in *Proceedings of the 11th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles (CAMS)* (Opatija), 389–394.

Bitar, G., Eriksen, B.-O. H., Lekkas, A. M., and Breivik, M. (2019a). "Energy-optimized hybrid collision avoidance for ASVs," in *Proceedings of the 17th IEEE European Control Conference (ECC)* (Naples), 2522–2529.

Bitar, G., Vestad, V. N., Lekkas, A. M., and Breivik, M. (2019b). "Warm-started optimized trajectory planning for ASVs," in *Proceedings of the 12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles (CAMS)* (Daejeon).

Campbell, S., Abu-Tair, M., and Naeem, W. (2014). An automatic COLREGs-compliant obstacle avoidance system for an unmanned surface vehicle. *Proc. Inst. Mech. Eng. M J. Eng. Marit. Environ.* 228, 108–121. doi: 10.1177/1475090213498229

Casalino, G., Turetta, A., and Simetti, E. (2009). "A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields," in *Proceedings of the 2009 IEEE OCEANS-EUROPE Conference* (Bremen).

Chauvin, C. (2011). Human factors and maritime safety. *J. Navig.* 64, 625–632. doi: 10.1017/S0373463311000142

Cockcroft, A. N., and Lameijer, J. N. F. (2004). *A Guide to the Collision Avoidance Rules.* Oxford, UK: Elsevier.

Deuflhard, P. (2011). *Newton Methods for Nonlinear Problems.* Berlin: Springer.

Eriksen, B.-O. H., and Breivik, M. (2017a). *Modeling, Identification and Control of High-Speed ASVs: Theory and Experiments.* Cham: Springer International Publishing, 407–431.

Eriksen, B.-O. H., and Breivik, M. (2017b). "MPC-based mid-level collision avoidance for ASVs using nonlinear programming," in *Proceedings of the 1st IEEE Conference on Control Technology and Applications (CCTA)* (Kohala Coast, HI), 766–772.

Eriksen, B.-O. H., and Breivik, M. (2018). "A model-based speed and course controller for high-speed ASVs," in *Proceedings of the 11th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles (CAMS)* (Opatija), 317–322.

Eriksen, B.-O. H., and Breivik, M. (2019). Short-term ASV collision avoidance with static and moving obstacles. *Model. Identif. Control* 40, 177–187. doi: 10.4173/mic.2019.3.4

Eriksen, B.-O. H., Breivik, M., Wilthil, E. F., Flåten, A. L., and Brekke, E. F. (2019). The branching-course model predictive control algorithm for maritime collision avoidance. *J. Field Robot.* 36, 1222–1249. doi: 10.1002/rob.21900

Eriksen, B.-O. H., Wilthil, E. F., Flåten, A. L., Brekke, E. F., and Breivik, M. (2018). "Radar-based maritime collision avoidance using dynamic window," in *Proceedings of the 2018 IEEE Aerospace Conference* (Big Sky, MT), 1–9.

Hagen, I. B., Kufoalor, D. K. M., Brekke, E., and Johansen, T. A. (2018). "MPC-based collision avoidance strategy for existing marine vessel guidance systems," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)* (Brisbane, QLD), 7618–7623.

Harati-Mokhtari, A., Wall, A., Brooks, P., and Wang, J. (2007). Automatic identification system (AIS): data reliability and human error implications. *J. Navig.* 60, 373–389. doi: 10.1017/S0373463307004298

Kufoalor, D. K. M., Brekke, E. F., and Johansen, T. A. (2018). "Proactive collision avoidance for ASVs using a dynamic reciprocal velocity obstacles method," in *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Madrid), 2402–2409.

Kuwata, Y., Wolf, M. T., Zarzhitsky, D., and Huntsberger, T. L. (2014). Safe maritime autonomous navigation with COLREGS, using velocity obstacles. *IEEE J. Ocean. Eng.* 39, 110–119. doi: 10.1109/joe.2013.2254214

Levander, O. (2017). Autonomous ships on the high seas. *IEEE Spectr.* 54, 26–31. doi: 10.1109/MSPEC.2017.7833502

Loe, Ø. A. G. (2008). *Collision avoidance for unmanned surface vehicles* (Master's thesis), Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Švec, P., Shah, B. C., Bertaska, I. R., Alvarez, J., Sinisterra, A. J., von Ellenrieder, K., et al. (2013). "Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic," in *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Tokyo), 3871–3878.

Szlapczynski, R., and Szlapczynska, J. (2017). Review of ship safety domains: models and applications. *Ocean Eng.* 145, 277–289. doi: 10.1016/j.oceaneng.2017.09.020

Tam, C., and Bucknall, R. (2010). Collision risk assessment for ships. *J. Mar. Sci. Technol.* 15, 257–270. doi: 10.1007/s00773-010-0089-7

Wächter, A., and Biegler, L. T. (2005). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.* 106, 25–57. doi: 10.1007/s10107-004-0559-y

Wu, D. (2019). *Proactive maritime collision avoidance based on historical AIS data* (Master's thesis), Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Zhang, X., Liniger, A., Sakai, A., and Borrelli, F. (2018). "Autonomous parking using optimization-based collision avoidance," in *Proceedings of the 57th IEEE Conference on Decision and Control (CDC)* (Miami, FL), 4327–4332.