Short-term ASV Collision Avoidance with Static and Moving Obstacles

Bjørn-Olav H. Eriksen   Morten Breivik

Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), NO-7491 Trondheim, Norway. E-mail: {bjorn-olav.h.eriksen, morten.breivik}@ieee.org

Abstract

This article considers collision avoidance (COLAV) for both static and moving obstacles using the branching-course model predictive control (BC-MPC) algorithm, which is designed for use by autonomous surface vehicles (ASVs). The BC-MPC algorithm originally only considered COLAV of moving obstacles, so in order to make the algorithm also be able to avoid static obstacles, we introduce an extra term in the objective function based on an occupancy grid. In addition, other improvements are made to the algorithm resulting in trajectories with less wobbling. The modified algorithm is verified through full-scale experiments in the Trondheimsfjord in Norway with both virtual static obstacles and a physical moving obstacle. A radar-based tracking system is used to detect and track the moving obstacle, which enables the algorithm to avoid obstacles without depending on vessel-to-vessel communication. The experiments show that the algorithm is able to simultaneously avoid both static and moving obstacles, while providing clear and readily observable maneuvers. The BC-MPC algorithm is compliant with rules 8, 13 and 17 of the the International Regulations for Preventing Collisions at Sea (COLREGs), and favors maneuvers following rules 14 and 15.

Keywords: Autonomous surface vehicles, collision avoidance, model predictive control

1 Introduction

Most, if not all, parts of society are currently being automated at a rapid pace. One example is the development of autonomous cars, as exemplified by the development efforts made by e.g. Tesla, Google and Uber. Such a trend is also ongoing in the maritime domain, where autonomous technology presents opportunities for increased cost efficiency, in addition to reducing the environmental impact of goods and passenger transport. One example of this is the Yara Birkeland project in Norway, where an electrically-powered autonomous cargo ship will replace 40000 diesel-powered truckloads of fertilizer each year by 2022 (Paris, 2017). Furthermore, it is reported that in excess of 75% of maritime accidents are caused by human errors (Chauvin, 2011; Levander, 2017), which also reveals a potential for increased safety by introducing autonomous technology at sea. Employing ASVs in areas where other vessels are present does,

however, require a robust COLAV system in order to avoid collisions and operate safely.

There exist several algorithms for ASV COLAV, e.g. the velocity obstacle (VO) algorithm (Kuwata et al., 2014), the A* algorithm (Schuster et al., 2014) and algorithms based on model predictive control (MPC) and optimization (Benjamin et al., 2006; Švec et al., 2013; Abdelaal and Hahn, 2016; Hagen et al., 2018). These algorithms are, however, designed with the idea of "one size fits all", where the same algorithm is used to solve both situations requiring proactive and reactive behaviors. A challenge with this approach is that the algorithm must be able to solve problems of a wide range sufficiently well, which makes the algorithm difficult to design and tune. A different approach is to utilize a hybrid architecture (Loe, 2008; Casalino et al., 2009), where the complementary strengths of different algorithms can be combined in a layered architecture. An example of a hybrid architecture is shown in Figure 1, where the COLAV system
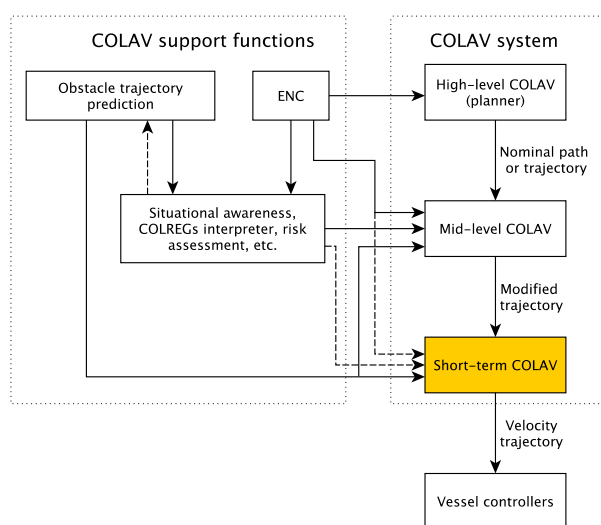
Figure 1: A hybrid architecture with three layers. The support functions provide relevant information for the COLAV algorithms, including prediction of obstacle trajectories, static obstacles from electronic nautical charts (ENC) and situational awareness in the form of COLREGs situations. Courtesy of (Eriksen et al., 2019b).

is divided into three layers, namely a high-level, mid-level and a short-term COLAV algorithm. The high-level planner performs long-term planning by finding a path or trajectory from an initial position to a goal position while avoiding static obstacles, satisfying time constraints and minimizing energy consumption. The mid-level algorithm attempts to follow the planned path or trajectory from the high-level planner, while making local modifications in order to avoid moving obstacles. This algorithm should be designed to comply with the maneuvering rules of the COLREGs, which dictate how vessels should behave in situations where there exists a risk of collision with other vessels (Cockcroft and Lameijer, 2004). The short-term COLAV algorithm inputs the modified trajectory from the mid-level algorithm, and should have low computational requirements ensuring that the COLAV system can react to sudden changes in the environment. This algorithm should also serve as a final safety barrier in situations where e.g. the mid-level algorithm fails to find a solution (Eriksen and Breivik, 2017b). In addition, the short-term algorithm should have a shorter planning horizon than the mid-level algorithm. This makes the short-term algorithm inherently capable of handling situations where moving obstacles do not comply with the COLREGs, and the maneuvering aspects of rules 14 and 15, which regulate the behavior in head-on and crossing situations, may need to be ignored to comply with the COLREGs. The algorithm should, however, maneuver in accordance with rules 14 and 15 when the situation allows it.

The authors have performed a significant amount of work on the hybrid architecture in Figure 1, concerning e.g. model-based vessel controllers (Eriksen and Breivik, 2017a, 2018), short-term COLAV (Eriksen et al., 2018, 2019b), mid-level COLAV (Eriksen and Breivik, 2017b) and a high-level planner interfaced to the mid-level algorithm (Bitar et al., 2019). In an upcoming article (Eriksen et al., 2019a), we populate the hybrid architecture with algorithms including the BC-MPC algorithm discussed in this article, and demonstrate COLAV compliant with COLREGs rules 8 and 13–17 in simulations. Work has also been performed on obstacle trajectory prediction (Hexeberg et al., 2017; Dalsnes et al., 2018). For the short-term COLAV layer, we initially focused on the dynamic window (DW) algorithm, using a radar-based tracking system for detecting and tracking obstacles (Wilthil et al., 2017). The reason for using exteroceptive sensors such as radars for detecting obstacles is that they do not depend on vessel-to-vessel communication or collaboration with other vessels, hence enabling avoidance of vessels which do not have or use automatic identification system (AIS) transponders. Another questionable aspect of AIS is that other vessels may provide incorrect information (Harati-Mokhtari et al., 2007), which can be difficult to detect and handle. However, there is a fair amount of noise on obstacle estimates originating from systems using exteroceptive sensors, which the DW algorithm was shown not to handle sufficiently well in full-scale experiments (Eriksen et al., 2018). We therefore developed the BC-MPC algorithm for short-term COLAV (Eriksen et al., 2019b), which is based on MPC and designed to be robust to obstacle estimate noise. This algorithm is shown to have good performance in full-scale experiments, but originally only accounts for moving obstacles.

In this article, we further develop the BC-MPC algorithm to also handle avoidance of static obstacles in addition to moving obstacles, as well as producing trajectories with less wobbling. The modified algorithm is verified in full-scale experiments in the Trondheimsfjord, Norway, showing good performance. The experiments are performed with virtual static obstacles, while a moving obstacle is detected and tracked using a radar, not depending on vessel-to-vessel communication.

The rest of this article is organized as follows: Section 2 presents the BC-MPC algorithm and the modifications done to it, Section 3 presents the experimental setup and results, while Section 4 concludes the article and points to possibilities for further work.

2 The BC-MPC algorithm

The BC-MPC algorithm (Eriksen et al., 2019b) is a COLAV algorithm designed using sample-based MPC, intended for short-term COLAV for ASVs. Sample-based MPC algorithms are based on computing an objective function over a

finite discrete search space and selecting the optimized solution, rather than utilizing search algorithms as in gradient-based algorithms. A benefit of sample-based algorithms is that they do not have problems with solving highly nonlinear and non-convex problems, which in general is difficult for gradient-based algorithms. This makes sample-based algorithms well suited for use in the short-term layer in Figure 1. Furthermore, the BC-MPC algorithm is designed to be robust with respect to noisy obstacle estimates, which is a significant source of disturbance when using exteroceptive sensors such as radars for detecting and tracking obstacles.

With respect to the COLREGs, the BC-MPC algorithm complies with rules 8, 13 and 17, and favors maneuvers following rules 14 and 15. In cases where the algorithm chooses to ignore the maneuvering aspects of rules 14 and 15, which can be required when rule 17 revokes a stand-on obligation, the maneuvers have an extended clearance to obstacles.

At each iteration, the algorithm computes a search space consisting of a finite number of possible trajectories, which each contains a sequence of maneuvers. Given this search space, an objective function is computed on the trajectories, and the optimized trajectory is selected and used as the reference to the vessel controllers which control the speed over ground (SOG) and course. The algorithm is based on MPC, hence only the first part of the optimized trajectory is used before a new solution is computed and implemented.

This section presents an overview of the BC-MPC algorithm. Interested readers are referred to Eriksen et al. (2019b) for more details on the algorithm. In addition, this section presents modifications enabling the algorithm to perform static obstacle avoidance and produce trajectories with less wobbling than the original algorithm.

2.1 Trajectory generation

At each iteration, a new finite search space of possible trajectories is generated. Every trajectory contains a number of sub-trajectories, each containing one maneuver. This naturally forms a tree structure, with the nodes representing vessel configurations and edges representing maneuvers. The initial condition is used as the root node, and the depth of the tree is equal to the number of maneuvers in each trajectory. Notice that it is not possible to guarantee optimality with this approach, since the search space is non-continuous. However, with a sufficient number of trajectories, good solutions can be found in an efficient manner even for highly nonlinear and non-convex problems.

The trajectory generation is performed by a repeatable maneuver-generation procedure, which when given a vessel configuration computes a set of sub-trajectories each containing one maneuver. Piecewise linear acceleration profiles in speed and course serve as a template for the maneuvers. An example of 5 motion primitives based on the acceler-
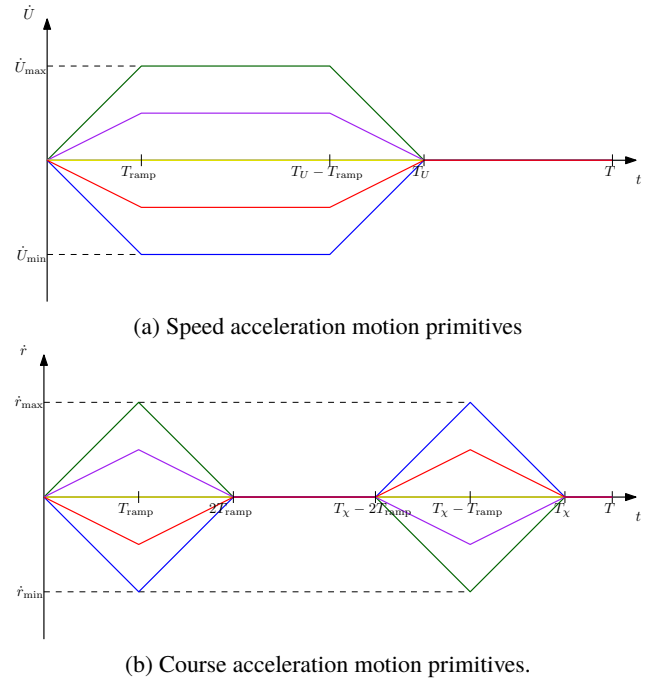

(a) Speed acceleration motion primitives

(b) Course acceleration motion primitives.

Figure 2: Acceleration motion primitives, where T is the step time, T_{ramp} denotes the ramp time, while T_U and T_χ are the SOG and course maneuver time lengths, respectively. The symbols $\dot{U}_{max}, \dot{U}_{min}, \dot{r}_{max}$ and $\dot{r}_{min}$ denote the acceleration limits of the vessel at the initial vessel state. Courtesy of (Eriksen et al., 2019b).

ation profiles in speed and course is shown in Figure 2. The acceleration profiles are dependent on the step time length (the maneuver time length) $T > 0$, the ramp time $T_{ramp} \in (0, \min(\frac{T_U}{2}, \frac{T_\chi}{4})]$ and the speed and course maneuver lengths, $T_U, T_\chi \in (0, T]$, respectively. Given a current vessel velocity, the maximum and minimum speed and course accelerations $\dot{U}_{max}, \dot{U}_{min}, \dot{r}_{max}$ and $\dot{r}_{min}$ are computed using a vessel model.

To improve the convergence properties of the algorithm, we employ a guidance function which can modify some of the trajectories in the search space. This is done by moving the closest acceleration sample in speed and course to a desired acceleration generated by the guidance function, if this is inside the feasible acceleration region.

Desired speed and course trajectories $U_d(t)$ and $\chi_d(t)$ are generated by analytically integrating the acceleration motion primitives. Numerical examples of 5 speed and 5 course trajectories are shown in figures 3 and 4. It should be noted that these trajectories are intended as reference trajectories for the vessel controllers, hence they are initiated in an open-loop fashion with the current desired speed and course in order to ensure continuous references for the vessel controllers. The desired speed and course trajectories are
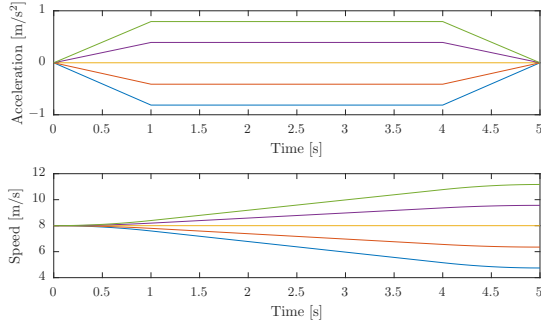
Figure 3: Example of 5 speed trajectories with ramp time $T_{\text{ramp}} = 1$ s, and maneuver and step time lengths $T_U = T = 5$ s. Acceleration is shown in the top plot, while speed is shown in the bottom plot. Courtesy of (Eriksen et al., 2019b).

joined together in a union set of desired velocity trajectories:

$$\mathcal{U}_d = \{U_{d,1}(t), U_{d,2}(t), \ldots, U_{d,N_U}(t)\}$$
$$\times \{\chi_{d,1}(t), \chi_{d,2}(t), \ldots, \chi_{d,N_\chi}(t)\}, \quad (1)$$

resulting in a total of $N_U \cdot N_\chi$ desired velocity trajectories where $N_U \in \mathbb{Z}^+$ and $N_\chi \in \mathbb{Z}^+$ are the number of speed and course motion primitives. To include feedback in the trajectory generation, we use an error model of the vessel to generate feedback-corrected speed and course trajectories $\bar{U}_d(t)$ and $\bar{\chi}_d(t)$, which similarly as in (1) is combined in a set $\bar{\mathcal{U}}_d$. The feedback-corrected speed and course trajectories are used to generate feedback-corrected predicted pose trajectories:

$$\bar{\mathcal{H}} = \left\{ \bar{\boldsymbol{\eta}}(t; \bar{U}(t), \bar{\chi}(t)) \big| (\bar{U}(t), \bar{\chi}(t)) \in \bar{\mathcal{U}} \right\}, \quad (2)$$

where $\bar{\boldsymbol{\eta}}(t; \bar{U}(t), \bar{\chi}(t))$ denotes a kinematic simulation procedure to obtain the vessel pose.

A full trajectory search space is created by first generating a set of sub-trajectories by using the maneuver-generation procedure initialized with the initial vehicle pose. At this stage, the prediction tree has a depth of one with the initial vessel pose as the root node and a set of leaf nodes each reached by one maneuver. Following this, we append the trajectories with another maneuver by repeating the maneuver-generation procedure, initialized on each of the leaf nodes, which increases the depth of the trajectory prediction tree with one level. This is repeated until the trajectory prediction tree has the desired depth, i.e. each trajectory has the desired number of maneuvers. This concept is illustrated in Figure 5. The acceleration profile parameters and number of speed and course motion primitives can be level-dependent, which allows for shaping the maneuvers differently and avoiding exponential growth with the number of levels. To reduce the complexity in tuning the algorithm, we use the same ramp time T_{ramp} and speed and course maneuver lengths T_U and T_χ throughout each level. For a desired
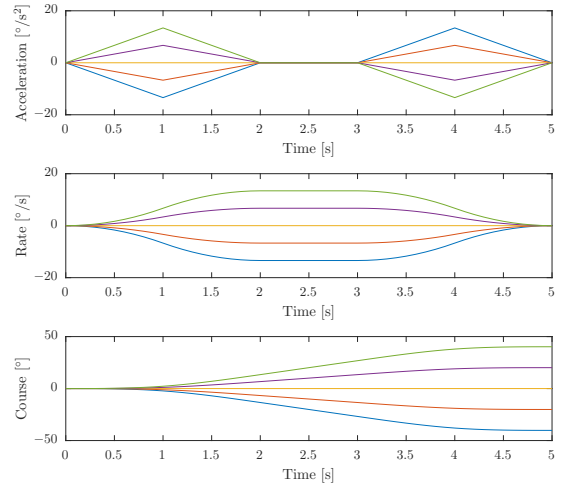

Figure 4: Example of 5 course trajectories with ramp time $T_{\text{ramp}} = 1$ s, and maneuver and step time lengths $T_\chi = T = 5$ s. Acceleration is shown in the top plot, rate in the middle plot and course in the bottom plot. Courtesy of (Eriksen et al., 2019b).

trajectory tree depth B (B maneuvers in each trajectory), this leaves us with deciding the step time lengths of each level $\boldsymbol{T} = [T_1, T_2, \ldots, T_B]$, and the number of speed and course maneuvers at each level $\boldsymbol{N}_U = [N_{U,1}, N_{U,2}, \ldots, N_{U,B}]$ and $\boldsymbol{N}_\chi = [N_{\chi,1}, N_{\chi,2}, \ldots, N_{\chi,B}]$.

A set of feedback-corrected predicted pose trajectories for a trajectory generation with $B = 3$ levels is shown in Figure 6. The ramp time is $T_{\text{ramp}} = 1$ s, and the speed and course maneuver lengths are $T_U = T_\chi = 5$ s. The step time lengths are $\boldsymbol{T} = [20, 30, 30]$ s, and the number of speed and course maneuvers are $\boldsymbol{N}_U = [1, 1, 1]$ and $\boldsymbol{N}_\chi = [5, 3, 3]$.

2.2 Selecting the optimized trajectory

Given a search space of vessel trajectories and a desired trajectory $\boldsymbol{p}_d(t) \in \mathbb{R}^2$, we solve an optimization problem to find the optimized desired velocity trajectory $\boldsymbol{u}_d^*(t) = \begin{bmatrix} U_d^*(t) & \chi_d^*(t) \end{bmatrix}^\top$ as:

$$\boldsymbol{u}_d^*(t) = \operatorname*{argmin}_{(\bar{\boldsymbol{\eta}}_k(t), \boldsymbol{u}_{d,k}(t)) \in (\bar{\mathcal{H}}, \mathcal{U}_d)} G(\bar{\boldsymbol{\eta}}_k(t), \boldsymbol{u}_{d,k}(t); \boldsymbol{p}_d(t)).$$
$$(3)$$

The objective function is given as:

$$G(\bar{\boldsymbol{\eta}}(t), \boldsymbol{u}_d(t); \boldsymbol{p}_d(t)) = w_{\text{al}} \text{align}(\bar{\boldsymbol{\eta}}(t); \boldsymbol{p}_d(t))$$
$$+ w_{\text{av,m}} \text{avoid}_{\text{m}}(\bar{\boldsymbol{\eta}}(t)) + w_{\text{av,s}} \text{avoid}_{\text{s}}(\bar{\boldsymbol{\eta}}(t))$$
$$+ w_{\text{t},U} \text{tran}_U(\boldsymbol{u}_d(t)) + w_{\text{t},\chi} \text{tran}_\chi(\boldsymbol{u}_d(t)), \quad (4)$$

where $w_{\text{al}}, w_{\text{av,m}}, w_{\text{av,s}}, w_{\text{t},U}, w_{\text{t},\chi} > 0$ are tuning parameters.

The align$(\cdot)$ function assigns a value to following the desired trajectory $\boldsymbol{p}_d(t)$. The avoid$_{\text{m}}(\cdot)$ function assigns a
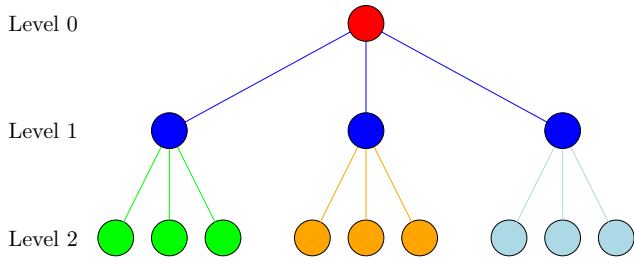
Figure 5: Illustration of a trajectory prediction tree with two levels. The red node is the root node containing the initial vessel configuration. Other colors group nodes and edges associated with each maneuver-generation procedure, which generate three maneuvers each time (given by combinations of N_U and N_χ satisfying $N_U \cdot N_\chi = 3$). The tree contains a total of nine trajectories, each consisting of two sub-trajectories.
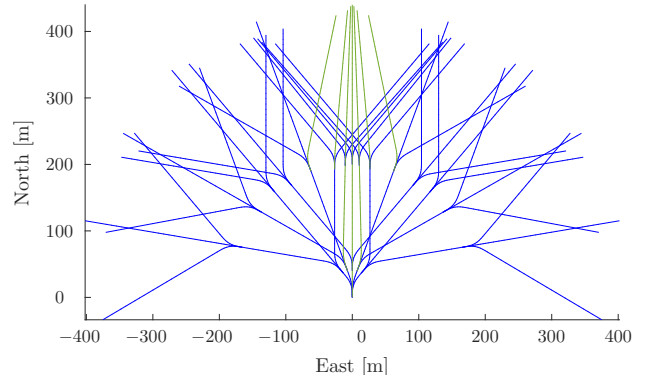

Figure 6: A set of predicted pose trajectories with three levels. Notice how the guidance function shifts some of the maneuvers, marked in dark green, to converge towards the desired trajectory, which is a straight-north trajectory from the initial pose (not shown in the figure). For illustration purposes, the trajectories only contain course maneuvers.

cost to traveling close to moving obstacles, which depends on the distance to an obstacle for each point on the predicted trajectories. The maneuvering rules in the COLREGs, rules 13–15, require the vessel to maneuver to starboard in head-on situations, and recommend to pass behind an obstacle if the obstacle approaches from the starboard side. To motivate the algorithm to follow these rules, while being free to ignore the specific maneuvering aspects if required in situations where the other vessel violates the COLREGs, we use the obstacle regions in Figure 7 when calculating this cost. The regions can be interpreted as follows: the margin region is allowable to enter, the safety region is not desirable to enter, while the collision region should not be entered. Notice that the algorithm will require a larger clearance in situations where the maneuvering rules in the COLREGs are ignored, e.g. if maneuvering to port in a head-on situation. See Eriksen et al. (2019b) for more details on the align($\cdot$) and avoid$_m(\cdot)$ terms.

In this article, we introduce the avoid$_s(\cdot)$, tran$_U(\cdot)$ and tran$_\chi(\cdot)$ terms. The avoid$_s(\cdot)$ term assigns a cost to avoiding static obstacles, while tran$_U(\cdot)$ and tran$_\chi(\cdot)$ are transitional cost terms increasing the robustness to noise. These terms will be discussed in detail in the following two sections.

2.3 Static obstacle avoidance

Static obstacles are modeled using an occupancy grid, which allows for easy representation of obstacles with arbitrary shapes like e.g. land and islands. In addition, static obstacles are padded with a decaying gradient to introduce some smoothness to the static obstacle avoidance function. Given an occupancy grid $O(\boldsymbol{p}) \in [0, 100]$ where $O(\boldsymbol{p}) = 100$ and $O(\boldsymbol{p}) = 0$ represents an occupied and empty cell, respec-
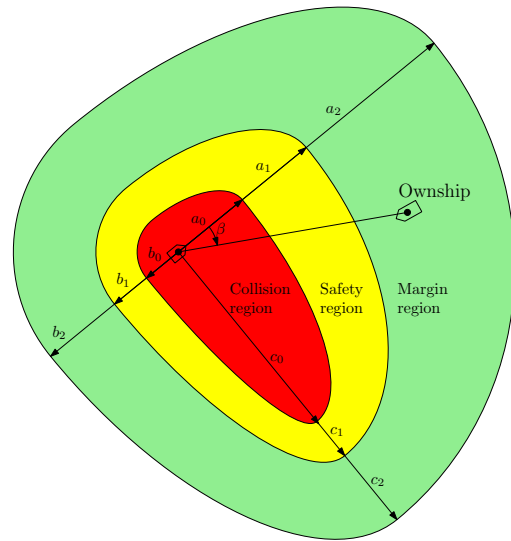

Figure 7: Avoidance cost regions centered at the moving obstacle, each constructed by one circular and three elliptical segments. The green, yellow and red regions are named the margin, safety and collision regions, respectively. The avoidance cost increases linearly with different gradients inside the green and yellow regions, while the cost is constant inside the red region. The variables a_i, b_i and $c_i, i \in \{1, 2, 3\}$ denote the region sizes, where $c_i = b_i + d_{\text{COLREGs}}$ with d_{COLREGs} controlling the COLREGs expansion. Courtesy of (Eriksen et al., 2019b).

tively, we define the static obstacle term as:

$$\text{avoid}_s(\bar{\boldsymbol{\eta}}(t)) = \int_{t_0}^{t_0+T} O(\bar{\boldsymbol{p}}(\gamma))\text{d}\gamma, \qquad (5)$$

where t_0 denotes the initial time and $\bar{\boldsymbol{\eta}}(t) = \begin{bmatrix} \bar{\boldsymbol{p}}(t)^\top & \bar{\psi}(t) \end{bmatrix}^\top$.

2.4 Speed and course transitional costs

In order to improve the robustness to noise on obstacle estimates, transitional cost is included in the objective function, which penalizes changing the planned trajectory from iteration to iteration. In Eriksen et al. (2019b), a single transitional cost term is used, which introduces a cost if one selects a different speed and/or course than the one closest to the one selected in the previous iteration. Note that the trajectory prediction is based on sampling the possible acceleration of the vessel in the current iteration, which implies that the exact trajectory selected in the previous iteration may not exist in the current search space.

Here, it is proposed to split the transitional cost term into separate speed and course terms. This motivates the algorithm to not alter the course if the speed is changed and vice versa, which would not be the case when using a single transitional cost term. The transitional cost terms are defined as:

$$\text{tran}_U(\boldsymbol{u}_d(t)) = \begin{cases} 1, & \int_{t_0}^{t_0+T_1} \left| U_d(\gamma) - U_d^-(\gamma) \right| \text{d}\gamma > e_{U,\min} \\ 0, & \text{else,} \end{cases} \qquad (6)$$

$$\text{tran}_\chi(\boldsymbol{u}_d(t)) = \begin{cases} 1, & \int_{t_0}^{t_0+T_1} \left| \chi_d(\gamma) - \chi_d^-(\gamma) \right| \text{d}\gamma > e_{\chi,\min} \\ 0, & \text{else,} \end{cases} \qquad (7)$$

with $\boldsymbol{u}_d(t) = \begin{bmatrix} U_d(t) & \chi_d(t) \end{bmatrix}^\top$. The variables $U_d^-(t)$ and $\chi_d^-(t)$ denote the current desired velocity trajectory tracked by the vessel controllers, and T_1 is the step time of the first trajectory maneuver. The variables $e_{U,\min}$ and $e_{\chi,\min}$ denote the minimum difference between the current desired velocity trajectory and the candidates:

$$e_{U,\min} = \min_{\boldsymbol{u}_d(t) \in \mathcal{U}_d} \int_{t_0}^{t_0+T_1} \left| U_d(\gamma) - U_d^-(\gamma) \right| \text{d}\gamma$$

$$e_{\chi,\min} = \min_{\boldsymbol{u}_d(t) \in \mathcal{U}_d} \int_{t_0}^{t_0+T_1} \left| \chi_d(\gamma) - \chi_d^-(\gamma) \right| \text{d}\gamma. \qquad (8)$$

3 Experimental results

The modified BC-MPC algorithm was tested in full-scale experiments in the Trondheimsfjord in Norway on the 27th of September 2018. This section describes the experimental setup and presents results from the experiments.

Figure 8: The Telemetron ASV, owned and operated by Maritime Robotics. Courtesy of Maritime Robotics.

Figure 9: The Kongsberg Seatex Ocean Space Drone 2, which is identical to the Ocean Space Drone 1 (OSD1). Courtesy of Kongsberg Seatex.

3.1 Experimental setup

The experimental setup was similar to the setup reported in Eriksen et al. (2019b), using the Telemetron ASV from Maritime Robotics as the ownship and the Ocean Space Drone 1 (OSD1) from Kongsberg Seatex as the moving obstacle. In addition, virtual static obstacles, expanded with a padding radius, were used to emulate static obstacles. The padding radius was selected as 150 m in most of the experiments. Notice that this padding radius only relates to static obstacles and that safety margins for moving obstacles are enforced by the obstacle regions in Figure 7. The Telemetron ASV, shown in Figure 8, is a 26-foot high-speed ASV capable of speeds up to 18 m/s and equipped for both manned and unmanned operations. The OSD1, shown in Figure 9, is a modified offshore lifeboat with a length of 12 m, and was steered at a constant speed of 5 knots during the experiments. The OSD1 played the role of a moving obstacle in the experiments, and was detected and tracked using a radar-based tracking system, which is discussed in detail in Wilthil et al. (2017) and Wilthil (2019). Both the BC-MPC algorithm and the radar tracking system was implemented using the Robot Operating System (ROS), and

Table 1: Telemetron ASV specifications.

Component	Description
Vessel hull	Polarcirkel Sport 845
Length	8.45 m
Width	2.71 m
Weight	1675 kg
Propulsion system	Yamaha 225 HP outboard engine
Motor control	Electro-mechanical actuation of throttle valve
Rudder control	Hydraulic actuation of outboard engine angle with proportional-derivative (PD) feedback control
Navigation system	Kongsberg Seatex Seapath 330+
Radar	Simrad Broadband 4GâĎć Radar
Processing platform	IntelÂő i7 3.4 GHz CPU, running Ubuntu 16.04 Linux

Table 2: BC-MPC algorithm parameters.

Parameter	Value	Description
B	3	Trajectory prediction tree depth
T	[20, 30, 30] s	Step time lengths
N_U	[5, 1, 1]	Number of SOG maneuvers
N_χ	[5, 3, 3]	Number of course maneuvers
T_{ramp}	1 s	Ramp time
T_U	5 s	SOG maneuver length
T_χ	5 s	Course maneuver length
w_{al}	1.5	Align weight
$w_{\mathrm{av,m}}$	6000	Moving obstacle avoid weight
$w_{\mathrm{av,s}}$	30	Static obstacle avoid weight
$w_{\mathrm{t},U}$	2100	SOG transitional cost weight
$w_{\mathrm{t},\chi}$	1050	Course transitional cost weight
a_0	50 m	Collision region major axis
a_1	150 m	Safety region major axis
a_2	250 m	Margin region major axis
b_0	25 m	Collision region minor axis
b_1	75 m	Safety region minor axis
b_2	125 m	Margin region minor axis
d_{COLREGs}	100 m	COLREGs expansion

was run on a processing platform with an IntelÂő i7 3.4 GHz CPU running Ubuntu 16.04 Linux onboard the Telemetron ASV. See Table 1 for specifications on the Telemetron ASV and the sensor system.

The BC-MPC algorithm was run at a rate of 0.2 Hz with the parameters in Table 2. At sea, vessels typically maneuver with large margins, making it safe to run the BC-MPC algorithm at this rate. Furthermore, the sample time of the radar is 2.5 s, which together with the dynamics of the tracking system algorithms results in the closed-loop time delay being dominated by the obstacle detection and tracking system. With the given tuning parameters, the BC-MPC algorithm has a runtime of approximately 0.4 s (including interfacing the radar tracking system), allowing for a higher rate if sensors providing faster updates are available. The tuning parameters are quite similar to the ones used in the original algorithm, with the exception of the first step time length, which is selected as 20 s instead of 5 s in Eriksen et al. (2019b). With this tuning, the algorithm plans for making one maneuver of 5 s at the current time and keeping a constant course until 20 s have passed, rather than planning to do a new maneuver after only 5 s. This represents a more "maritime" way to maneuver compared to performing rapid consecutive maneuvers, and the transitional cost terms will motivate the algorithm to keep a constant course rather than selecting a new planned maneuver. Notice, however, that the algorithm is still free to choose a new maneuver every 5 s, but the transitional cost terms will favor keeping constant speed and course. To avoid that the vessel controller limited the performance of the COLAV system, we used a model-based speed and course controller shown to have high performance for high-speed ASVs (Eriksen and Breivik, 2018).

During the experiments, we tested four different scenarios:

1. A static-only scenario with two static obstacles.

2. A head-on situation with the OSD1 and four static obstacles.

3. A crossing situation with the OSD1 and one static obstacle.

4. An overtaking situation with the OSD1 and one static obstacle.

The desired speed of the Telemetron ASV was 5 m/s in all the scenarios, except the overtaking scenario where the desired speed was 8 m/s.

3.2 Scenario 1

Scenario 1 is shown in Figure 10. Here, two static obstacles block the desired trajectory, requiring the BC-MPC algorithm to circumvent the obstacles. This scenario may seem a bit unrealistic, since the high-level planner and mid-level COLAV algorithm should plan paths which avoid static obstacles. However, the BC-MPC algorithm must be able to avoid static obstacles in order to stay safe in situations where we deviate from the desired trajectory, e.g. when avoiding moving obstacles or in situations where the mid-level algorithm is unable to produce a solution. The ownship converges to the desired trajectory before avoiding the first static obstacle by maneuvering to starboard. It would probably have been better to maneuver to port, since this would avoid having to pass through the narrow channel between the first and the second obstacle. The BC-MPC algorithm does, however, have a limited planning horizon of 80 s with the current tuning parameters, which makes it unaware of the
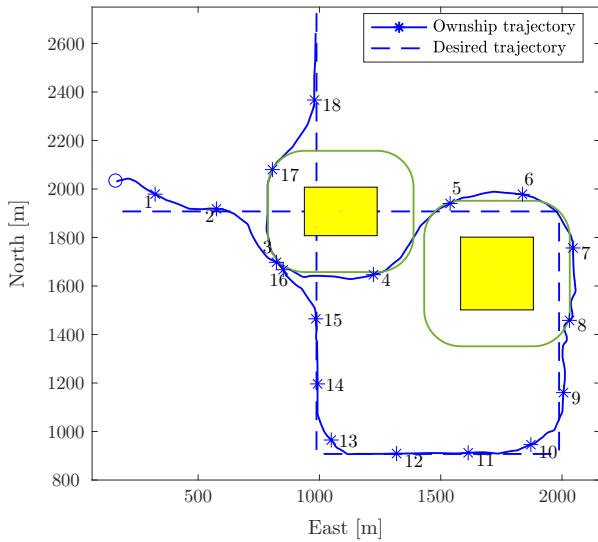
Figure 10: Scenario 1: Static-only scenario. The desired trajectory intersects with two obstacles, which the ownship successfully avoids. The blue circle denotes the initial position, while the numbers and asterisks mark each 60 s of the experiment. The yellow patches show the static obstacles, while the dark green contour lines show the padding regions.
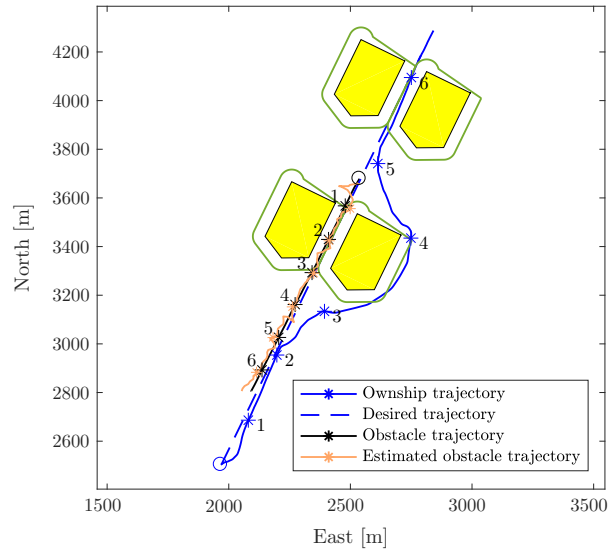

Figure 11: Scenario 2: Head-on situation. The desired trajectory passes through a narrow channel, which is blocked by the OSD1. The circles denote the initial positions, while the numbers and asterisks mark each 60 s of the experiment. The yellow patches show the static obstacles, while the dark green contour lines show the padding regions.

narrow channel when making the decision of maneuvering to starboard. Subsequently, the ownship converges towards the desired trajectory and passes the second obstacle by having a small distance to the desired trajectory, which resides slightly inside the padding region of the static obstacle. After passing the second obstacle, the ownship converges to the desired trajectory, before avoiding the first obstacle once again.

3.3 Scenario 2

Scenario 2 is a head-on situation where the desired trajectory goes through a narrow channel composed by two static obstacles, and the channel entry is blocked by the OSD1. In this scenario, the padding distance was selected as 50 m in order to create the narrow channel between the obstacles. As shown in Figure 11, the ownship avoids the OSD1 by maneuvering to starboard and hence complying with the COLREGs. Following this turn, the first static obstacle is passed on the east side. The ownship returns to the desired trajectory and travels through the channel composed by the two last static obstacles.

3.4 Scenario 3

Scenario 3, shown in Figure 12, is a crossing situation where the OSD1 approaches from the ownship's starboard side, re-

quiring the ownship to give way to avoid collision according to the COLREGs. In addition, there is a static obstacle on the starboard side of the ownship, blocking the ownship from maneuvering to starboard early. In compliance with the COLREGs, the ownship performs a starboard maneuver in order to pass behind the OSD1, while passing close to the boundary of the static obstacle. When the OSD1 has been passed, the ownship slowly converges towards the desired trajectory. The reason for the slow convergence is that the cost that the transitional cost terms introduces is just too large for the algorithm to change to a trajectory with a faster convergence. This is sometimes observed, but does not compromise safety and is a subject of tuning the transitional cost weights $w_{t,U}$ and $w_{t,\chi}$.

3.5 Scenario 4

Scenario 4 is an overtaking situation where the ownship approaches the OSD1 from behind. To allow the vessel being overtaken to maneuver to starboard if it finds itself in a separate collision situation, the BC-MPC algorithm is designed to favor a port turn in overtaking situations. However, as shown in Figure 13, a static obstacle is blocking the port side of the obstacle, which makes the ownship pass the obstacle on its starboard side. As mentioned, the BC-MPC algorithm is designed to pass with a larger clearance if passing on the port side rather than the starboard side, which
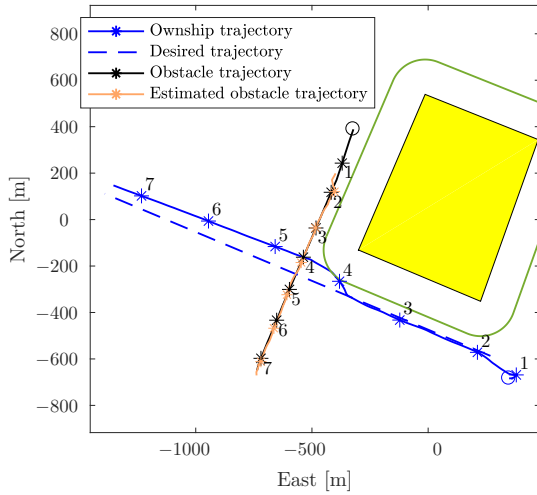
Figure 12: Scenario 3: Crossing situation. The desired trajectory intersects with the OSD1, which approaches from starboard. The static obstacle encloses Munkholmen, which is a small island located in the Trondheimsfjord. The circles denote the initial positions, while the numbers and asterisks mark each 60 s of the experiment. The yellow patch shows the static obstacle, while the dark green contour line shows the padding region.

can be seen by comparing this scenario with Experiment 3 in (Eriksen et al., 2019b).

3.6 Experiment summary

The BC-MPC algorithm is able to avoid collisions in all the scenarios, while converging to the desired trajectory when it is not obstructed by obstacles. The resulting ownship trajectories are clear and generally show the intension of the BC-MPC algorithm. The ownship trajectories are, however, a bit wobbly when the algorithm traverses alongside static obstacles. The reason for this behavior is the trajectory search space consisting of a finite number of trajectories, of which none may traverse exactly parallel to the static obstacle. This causes the algorithm sometimes to "zig-zag" along static obstacles, as seen in Scenario 1. In the usual case where the mid-level algorithm would recompute a collision-free trajectory circumventing the obstacles, the BC-MPC algorithm would however be able to traverse smoothly along the obstacles by following the desired trajectory. Also, due to algae growth on the hull, the vessel dynamics had changed quite a bit since the model-based vessel controller was tuned, which also contributed to wobbling in the form of course overshoots.

As seen in Table 3, the ownship travels inside the padding region of the static obstacles. This is to be expected, since the objective function is only sensitive to the static obstacles when the trajectory resides inside of the padding region.
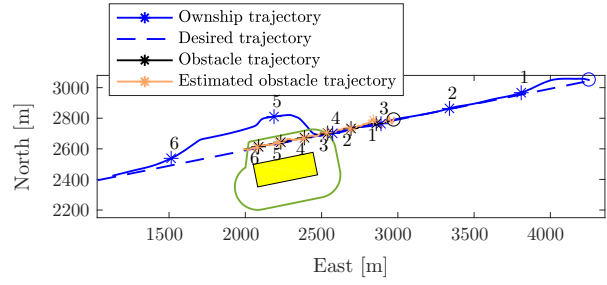

Figure 13: Scenario 4: Overtaking situation. The ownship overtakes the OSD1 by passing on the starboard side, while avoiding the static obstacle. The circles denote the initial positions, while the numbers and asterisks mark each 60 s of the experiment. The yellow patch shows the static obstacle, while the dark green contour line shows the padding region.

Table 3: Minimum distance to obstacles. *The padding distance in Scenario 2 is 50 m.

Scenario number	Minimum distance to static obstacles	Minimum distance to moving obstacle
Scenario 1	130.4 m	–
Scenario 2	31.3 m*	167.1 m
Scenario 3	148.6 m	76.1 m
Scenario 4	115.9 m	145.3 m

Hence, the padding region and static avoidance weight $w_{av,s}$ should be selected such that a sufficient safety margin is achieved. A formulation with multiple regions with different gradients, as for moving obstacles, could make it easier to tune the algorithm to obtain a desired safety margin to static obstacles. The required distance to the moving obstacle is a bit more complex to discuss, since the obstacle regions sizes depend on the relative bearing. The ownship does, however, stay outside of the safety region in the head-on and crossing scenarios (scenarios 2 and 3), while we slightly enter the safety region in the overtaking scenario (Scenario 4).

4 Conclusion and further work

In this article, we have presented two modifications to the BC-MPC algorithm for ASV COLAV. The first modification allows the algorithm to avoid static obstacles in the form of an occupancy grid. The second modification concerns improved transitional cost terms by introducing transitional cost in speed and course separately, motivating the algorithm to not change the course if the speed is changed and vice versa. In addition, the algorithm tuning has been changed in order to obtain more "maritime" maneuvers and better utilize the transitional cost terms. The modified BC-

MPC algorithm is tested in full-scale experiments in the Trondheimsfjord in Norway. A moving obstacle is detected and tracked using a radar-based system, while virtual static obstacles are added in the COLAV system. Four different scenarios were tested in experiments, all of which provided good results.

In Eriksen et al. (2019a), the authors have used the BC-MPC algorithm described in this article in a hybrid architecture, demonstrating COLAV compliant with COLREGs rules 8 and 13–17 in simulations. In the future, we would like to perform an extensive simulation study of the BC-MPC algorithm, in order to analyze the algorithm's performance in greater detail.

Acknowledgments

References

Abdelaal, M. and Hahn, A. NMPC-based trajectory tracking and collision avoidance of unmanned surface vessels with rule-based COLREGs confinement. In *Proc. of the 2016 IEEE Conference on Systems, Process and Control (ICSPC)*. Melaka, Malaysia, pages 23–28, 2016. doi:10.1109/SPC.2016.7920697.

Benjamin, M. R., Leonard, J. J., Curcio, J. A., and Newman, P. M. A method for protocol-based collision avoidance between autonomous marine surface craft. *Journal of Field Robotics*, 2006. 23(5):333–346. doi:10.1002/rob.20121.

Bitar, G., Eriksen, B.-O. H., Lekkas, A. M., and Breivik, M. Energy-optimized hybrid collision avoidance for ASVs. In *Proc. of the 17th IEEE European Control Conference (ECC)*. Naples, Italy, pages 2522–2529, 2019. doi:10.23919/ECC.2019.8795645.

Casalino, G., Turetta, A., and Simetti, E. A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields. In *Proc. of the 2009 IEEE OCEANS-EUROPE Conference*. Bremen, Germany, 2009. doi:10.1109/oceanse.2009.5278104.

Chauvin, C. Human factors and maritime safety. *Journal of Navigation*, 2011. 64(4):625–632. doi:10.1017/S0373463311000142.

Cockcroft, A. N. and Lameijer, J. N. F. *A Guide to the Collision Avoidance Rules*. Elsevier, 2004.

Dalsnes, B. R., Hexeberg, S., Flåten, A. L., Eriksen, B.-O. H., and Brekke, E. F. The neighbor course distribution method with Gaussian mixture models for AIS-based vessel trajectory prediction. In *Proc. of the 21st IEEE International Conference on Information Fusion (FUSION)*. Cambridge, UK, pages 580–587, 2018. doi:10.23919/ICIF.2018.8455607.

Eriksen, B.-O. H., Bitar, G., Breivik, M., and Lekkas, A. M. Hybrid collision avoidance for ASVs compliant with COLREGs rules 8 and 13–17. 2019a. Submitted to Frontiers in Robotics and AI, preprint available at https://arxiv.org/abs/1907.00198.

Eriksen, B.-O. H. and Breivik, M. *Modeling, Identification and Control of High-Speed ASVs: Theory and Experiments*, pages 407–431. Springer International Publishing, 2017a. doi:10.1007/978-3-319-55372-6_19.

Eriksen, B.-O. H. and Breivik, M. MPC-based mid-level collision avoidance for ASVs using nonlinear programming. In *Proc. of the 1st IEEE Conference on Control Technology and Applications (CCTA)*. Mauna Lani, Hawai'i, USA, pages 766–772, 2017b. doi:10.1109/CCTA.2017.8062554.

Eriksen, B.-O. H. and Breivik, M. A model-based speed and course controller for high-speed ASVs. In *Proc. of the 11th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles (CAMS)*. Opatija, Croatia, pages 317–322, 2018. doi:10.1016/j.ifacol.2018.09.504.

Eriksen, B.-O. H., Breivik, M., Wilthil, E. F., Flåten, A. L., and Brekke, E. F. The branching-course model predictive control algorithm for maritime collision avoidance. *Journal of Field Robotics*, 2019b. 36(7):1222–1249. doi:10.1002/rob.21900.

Eriksen, B.-O. H., Wilthil, E. F., Flåten, A. L., Brekke, E. F., and Breivik, M. Radar-based maritime collision avoidance using dynamic window. In *Proc. of the 2018 IEEE Aerospace Conference*. Big Sky, Montana, USA, pages 1–9, 2018. doi:10.1109/AERO.2018.8396666.

Hagen, I. B., Kufoalor, D. K. M., Brekke, E. F., and Johansen, T. A. MPC-based collision avoidance strategy for existing marine vessel guidance systems. In *Proc. of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia, pages 7618–7623, 2018. doi:10.1109/ICRA.2018.8463182.

Harati-Mokhtari, A., Wall, A., Brooks, P., and Wang, J. Automatic identification system (AIS): Data reliability and human error implications. *Journal of Navigation*, 2007. 60(3):373–389. doi:10.1017/S0373463307004298.

Hexeberg, S., Flåten, A. L., Eriksen, B.-O. H., and Brekke, E. F. AIS-based vessel trajectory prediction. In *Proc. of the 20th IEEE International Conference on Information Fusion (FUSION)*. Xi'an, China, pages 1–8, 2017. doi:10.23919/ICIF.2017.8009762.

Kuwata, Y., Wolf, M. T., Zarzhitsky, D., and Huntsberger, T. L. Safe maritime autonomous navigation with COLREGS, using velocity obstacles. *IEEE Journal of Oceanic Engineering*, 2014. 39(1):110–119. doi:10.1109/joe.2013.2254214.

Levander, O. Autonomous ships on the high seas. *IEEE Spectrum*, 2017. 54(2):26–31. doi:10.1109/MSPEC.2017.7833502.

Loe, Ø. A. G. *Collision Avoidance for Unmanned Surface Vehicles*. Master's thesis, Norwegian University of Science and Technology (NTNU), 2008.

Paris, C. Norway takes lead in race to build autonomous cargo ships. https://www.wsj.com/articles/norway-takes-lead-in-race-to-build-autonomous-cargo-ships-1500721202, 2017. Accessed: 2019-05-22.

Schuster, M., Blaich, M., and Reuter, J. Collision avoidance for vessels using a low-cost radar sensor. In *Proc. of the 19th IFAC World Congress*. pages 9673–9678, 2014. doi:10.3182/20140824-6-ZA-1003.01872.

Švec, P., Shah, B. C., Bertaska, I. R., Alvarez, J., Sinisterra, A. J., von Ellenrieder, K., Dhanak, M., and Gupta, S. K. Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic. In *Proc. of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Tokyo, Japan, pages 3871–3878, 2013. doi:10.1109/IROS.2013.6696910.

Wilthil, E. F. *Maritime Target Tracking with Varying Sensor Performance*. Ph.D. thesis, Norwegian University of Science and Technology (NTNU), 2019.

Wilthil, E. F., Flåten, A. L., and Brekke, E. F. *A Target Tracking System for ASV Collision Avoidance Based on the PDAF*, pages 269–288. Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-55372-6_13.