MPC-based Mid-level Collision Avoidance for ASVs using Nonlinear Programming

Bjørn-Olav H. Eriksen, Morten Breivik

***Abstract*— In this paper, we present a mid-level collision avoidance algorithm for autonomous surface vehicles (ASVs) based on model predictive control (MPC) using nonlinear programming. The algorithm enables avoidance of both static and moving obstacles, and following of a desired nominal trajectory if there is no danger of collision. We compare two alternative objective functions, where one is a quadratic function and the other is a nonlinear function designed to produce maneuvers observable for other vessels in compliance with rule 8 of the International Regulations for Preventing Collisions at Sea (COLREGS). The algorithm is implemented in the CASADI framework and uses the IPOPT solver. The performance of the algorithm is evaluated through simulations which show promising results. Furthermore, the algorithm is considered computationally feasible to run in real time. This algorithm serves as a base algorithm for further development in order to ensure full COLREGS compliance.**

I. Introduction

The development and use of autonomous technology in both industry and research is continuously moving forward. In particular, the automotive industry has had a leading role, while the development in the maritime domain has not received similar focus, even though the potential benefits from developing and utilizing autonomous marine technology is great. Employing autonomous surface vehicles (ASVs) for marine operations such as oceanography, bathymetry, passenger and goods transport, marine patrolling, etc. can result in increased safety, widened operational window and reduced costs.

A requirement for employing ASVs in the marine domain is a collision avoidance (COLAV) system. Such a system must be able to plan observable long-term maneuvers and at the same time respond to rapid changes in the environment, such as a nearby high-speed vessel suddenly changing course. Employing observable maneuvers is especially important to establish implicit communication between vessels based on their maneuvers. Performing observable maneuvers is also a requirement of the International Regulations for Preventing Collisions at Sea (COLREGS), which acts as "rules of the road" for marine surface vessels, both manned and unmanned.

COLAV algorithms can in general be divided into reactive and deliberate algorithms. Reactive algorithms utilize currently available sensor data, and employ a minimum of computations. This often produces sub-optimal solutions, but the low computational requirement makes the algorithms capable of responding to rapid changes in the environment. Examples of reactive algorithms include velocity obstacles (VO) [1, 2] and the dynamic window (DW) approach [3–5]. Deliberate algorithms, on the other hand, utilize apriori information (often in the form of aggregated sensor information, maps, etc.) and can encode more sophisticated criteria for the behavior, for instance generating observable maneuvers which are optimal in a global sense. Examples of deliberate algorithms include rapidly-exploring random trees (RRT) [6], graph search algorithms such as A* and D* [7, 8] and constrained nonlinear optimization [9, 10]. A downside of deliberate algorithms is their computational complexity, which results in high computational requirements and possibly challenges for real-time implementation. The solution to this issue is to employ a hybrid architecture, merging reactive and deliberate algorithms [9, 11]. An example of a three-layered hybrid architecture is shown in Fig. 1. In such an architecture, the top layer can perform long-term planning from the start position to the goal position, only considering static obstacles (land, reefs, etc.) while performing optimal planning with respect to arrival time, energy consumption, etc. This would typically be performed offline, but may also be computed online if required. The second layer inputs the desired nominal path or trajectory, and makes local adaptations if necessary. The planning horizon of this layer

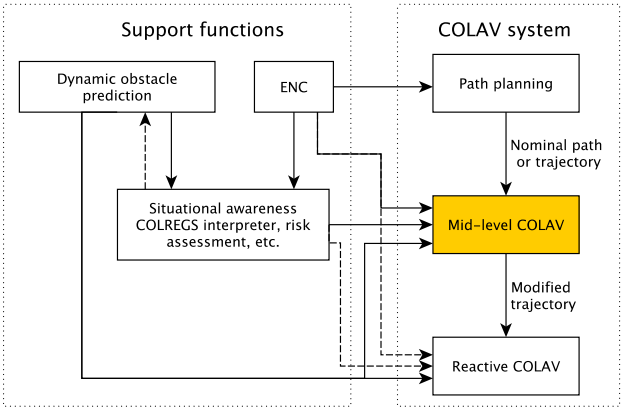

Fig. 1: Example of a hybrid COLAV architecture with three layers. The middle and top layers are typically deliberate algorithms, while a reactive algorithm is used in the bottom layer. In this architecture, electronic nautical charts (ENC), prediction of moving obstacles and interpretation of COLREGS are handled by separate support functions.

Bjørn-Olav H. Eriksen and Morten Breivik are with the Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), NO-7491 Trondheim, Norway. Email: {bjorn-olav.h.eriksen,morten.breivik}@ieee.org

should be long enough to plan avoidance maneuvers, but short enough to ensure computational feasibility. The reactive layer then tries to follow the trajectory specified by the mid-level algorithm, and makes local adaptations if necessary.

In this paper, we focus our attention to deliberate mid-layer COLAV formulated using nonlinear programming (NLP). Constrained optimization in the form of model predictive control (MPC) has already been applied for automotive COLAV [12, 13]. The use of MPC for ASV COLAV has been investigated in [10], however only using brute-force techniques by discretization of the search space in a finite number of control inputs. In this paper, we present an MPC algorithm for mid-level COLAV, implemented using NLP and solved using the IPOPT [14] solver in the CASADI [15] framework. The algorithm is tested with two objective functions, where one generates maneuvers which are compliant with rule 8 of COLREGS.

The paper is structured as follows: Section II presents an ASV model and kinematic constraints. An overview of COLREGS, and some discussion of the most applicable rules to ASVs is given in Section III. In Section IV, we define the mid-level COLAV problem as an optimization problem, while simulation results are shown in Section V. Finally, concluding remarks and possibilities for further work are given in Section VI.

II. ASV MODELING

In the scope of deliberate mid-level COLAV, we propose to use a purely kinematic model. This simplifies the design, since no vehicle kinetic model is required. For underactuated ASVs, a kinematic model can be formulated as [16]:

$$\dot{\boldsymbol{\eta}} = \begin{bmatrix} \cos(\psi) & 0 \\ \sin(\psi) & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{u}, \tag{1}$$

where $\boldsymbol{\eta} = \begin{bmatrix} N & E & \psi \end{bmatrix}^T \in \mathbb{R}^2 \times S$ is the vehicle pose with N and E representing the north and east position and ψ representing the yaw angle (heading). The vector $\boldsymbol{u} = \begin{bmatrix} U & r \end{bmatrix}^T \in \mathbb{R}^2$ is the vehicle velocity with U and r representing the vehicle speed over ground (SOG) and yaw rate (ROT), respectively. Note that the vehicle is assumed to have zero side-slip, hence the vehicle heading and course are assumed to be aligned. In addition, note that the ocean current is not included in (1).

The argument for neglecting the vehicle kinetics and the ocean current is that the reactive layer will take these into account when following the mid-level trajectory. However, some criteria should be employed to ensure a degree of feasibility with respect to the vessel capabilities. Underactuated ASVs typically employ a rudder for controlling ROT, and uses the forward thrust from the propeller to control SOG. For such a configuration, the actuation moment in yaw is highly dependent on the SOG since the rudder force is dependent on the fluid velocity over the rudder.

In [16], a nonlinear kinetic model for the Maritime Robotics Telemetron ASV, shown in Fig. 2, is identified. The model is of the form:

Fig. 2: The Telemetron ASV. Courtesy of Maritime Robotics.

$$\boldsymbol{M}(\boldsymbol{x})\dot{\boldsymbol{x}} + \boldsymbol{\sigma}(\boldsymbol{x}) = \boldsymbol{\tau}, \tag{2}$$

where $\boldsymbol{x} = \begin{bmatrix} U & r \end{bmatrix}^T$, $\boldsymbol{M}(\boldsymbol{x})$ is a velocity-dependent inertia matrix, $\boldsymbol{\sigma}(\boldsymbol{x})$ is a damping vector and $\boldsymbol{\tau}$ is a normalized control input. The inertia matrix and damping vector are formed by a number of polynomial and asymptotic terms [16]. Keeping in mind that the possible SOG and ROT are dependent on each other, we use the model to develop the kinematic limitations:

$$\boldsymbol{x} \in V_s = \Big\{ \begin{bmatrix} U & r \end{bmatrix}^T \in \mathbb{R}^2 | U_{\min}(r) \leq U \leq U_{\max}(r) \\ \wedge\, r_{\min} \leq r \leq r_{\max} \Big\}, \tag{3}$$

where $r_{\min} < 0$ and $r_{\max} > 0$ are constants, while $U_{\min}(r)$ and $U_{\max}(r)$ are functions of r. We handle the ROT limitation as constant, while letting the SOG limitation capture the dependency between the SOG and ROT limitations.

III. THE INTERNATIONAL REGULATIONS FOR PREVENTING COLLISIONS AT SEA (COLREGS)

COLREGS intends to provide a set of rules which when followed should avoid ship-to-ship collisions. The rules have been revised a number of times, to cope with the advances in technology and the increasing use of the seaways for different activities.

COLREGS is divided in 5 parts, with a total of 38 rules [17]. Part B (Steering and Sailing Rules) considers the rules most relevant for implementing COLAV algorithms. The rules most commonly discussed in the litterature on autonomous COLAV, see e.g. [2], are rules 13-15, which describe the overtaking, head-on, and crossing situations:

- Rule 13 Overtaking situation: An overtaking situation occurs when a vessel is approaching another vessel of more than 22.5° abaft her beam. A vessel overtaking is required to keep clear of the vessel being overtaken, such that risk of collision is avoided.
- Rule 14 Head on: A head-on situation occurs when two vessels are approaching at reciprocal (or nearly reciprocal) courses. This is usually interpreted as a relative bearing of $180^\circ \pm 6^\circ$. In such a situation, both vessels are required to do a port turn to avoid collision.

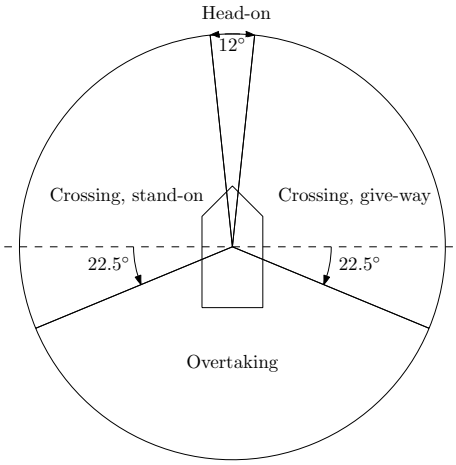

Fig. 3: Graphical interpretation of COLREGS situations.

Rule 15 Crossing: A crossing situation occurs when a vessel is approaching with a relative bearing of between 25° abaft her beam and 6° abaft her bow. In such a situation, the vessel having the other vessel on her starboard side is required to give way to avoid collision. If possible, one should avoid passing in front of the other vessel.

A graphical interpretation of the COLREGS situations is shown in Fig. 3. Formally, the vessel required to avoid collision is named the give-way vessel while the other vessel is named the stand-on vessel.

In addition to rules 13-15, rules 8, 16 and 17 are quite relevant for autonomous COLAV:

Rule 8 Action to avoid collision: This rule requires that actions taken to avoid collision should be large enough to be observable for other ships. An implication for this is that a series of small alterations of course and/or speed should not be applied. If there is sufficient space available, alterations of course should be the preferred action to avoid collision.

Rule 16 Action by give-way vessel: This rule dictates that the give-way vessel should, if possible, take early and substantial action to keep well clear of the stand-on vessel.

Rule 17 Action by stand-on vessel: This rule dictates that the stand-on vessel should keep her speed and course. However, if it is apparent that the give-way vessel is not taking action to avoid collision, the stand-on vessel is required to apply actions to best avoid collision.

Rule 8 is considered in [10] by limiting the possible actions, resulting in distinct observable actions. These rules are, however, seldom considered in the literature. It should be noted that rule 17 actually requires the stand-on vessel to not attempt to avoid collision, before it is apparent that the give-way vessel is not acting to avoid collision. By applying a hybrid architecture, rule 17 can be handled implicitly by implementing the latter part in the reactive algorithm.

IV. MPC-BASED MID-LEVEL COLAV

This section describes the parameterization and implementation of the mid-level COLAV algorithm.

A. Control objective

For the mid-level COLAV algorithm, we want to input a desired nominal path or trajectory which should be followed if there is no danger of collision. A trajectory can for example be a sequence of waypoints together with a desired speed. By assuming that we wish to follow straight lines between the waypoints, we can define a desired nominal trajectory:

$$\boldsymbol{p}_d(t) = \begin{bmatrix} N_d(t) \\ E_d(t) \end{bmatrix}. \tag{4}$$

Consequently, we can formulate the control objective as that the vessel trajectory $\boldsymbol{p}(t) = \begin{bmatrix} N(t) & E(t) \end{bmatrix}^T$ should converge towards $\boldsymbol{p}_d(t)$, while avoiding collisions and obeying COLREGS.

B. Obstacle modeling

A common simplification is to model both moving and static obstacles as circles. We define a static obstacle using a center position and a radius as $O_{s_i} = (\boldsymbol{p}_{s_i}, r_{s_i}) \in \mathbb{R}^2 \times \mathbb{R}^+$. A set of S static obstacles can then be defined as $\mathcal{O}_s = \{O_{s_1}, O_{s_2}, \dots O_{s_S}\}$.

Similarly, a moving obstacle can be defined by a time-varying center position and a radius as $O_{m_i}(t) = (\boldsymbol{p}_{m_i}(t), r_{m_i}) \in \mathbb{R}^2 \times \mathbb{R}^+$, where $\boldsymbol{p}_{m_i} : \mathbb{R} \to \mathbb{R}^2$. A set of M moving obstacles is then defined as $\mathcal{O}_m = \{O_{m_1}(t), O_{m_2}(t), \dots O_{m_M}(t)\}$.

C. Optimization problem construction

To solve the mid-level COLAV problem in Section IV-A as an optimal control problem (OCP), we first define the general OCP:

$$\begin{aligned} \text{minimize} \quad & \phi(\boldsymbol{\eta}(t), \boldsymbol{u}(t)) \\ \text{subject to} \quad & \dot{\boldsymbol{\eta}}(t) = \boldsymbol{F}(\boldsymbol{\eta}(t), \boldsymbol{u}(t)) \\ & \boldsymbol{h}(\boldsymbol{\eta}(t), \boldsymbol{u}(t)) \leq \boldsymbol{0}, \\ & \boldsymbol{\eta}(t_0) = \bar{\boldsymbol{\eta}}_0, \end{aligned} \tag{5}$$

where $\phi : (\mathbb{R}^2 \times S) \times \mathbb{R}^2 \to \mathbb{R}$ is the objective function, $\boldsymbol{\eta}(t)$ is the vehicle pose trajectory, $\boldsymbol{u}(t)$ is the control input trajectory and $\boldsymbol{F}(\boldsymbol{\eta}(t), \boldsymbol{u}(t))$ denotes the kinematic model (1). The function $\boldsymbol{h} : (\mathbb{R}^2 \times S) \times \mathbb{R}^2 \to \mathbb{R}^{n_h}$ forms n_h inequality constraints and $\bar{\boldsymbol{\eta}}_0 \in \mathbb{R}^2 \times S$ is the initial vehicle state.

Even though the continuous OCP in some cases is possible to solve e.g. by using Pontryagin's maximum principle, it is generally practical to define a nonlinear program (NLP) by discretizing (5). The discretized OCP can be formulated using a number of techniques. We choose to use direct multiple shooting, where both the state and control inputs are explicitly defined as decision variables. The NLP with Np prediction steps is:

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \phi(\boldsymbol{w}) \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{w}) \leq \boldsymbol{0}, \end{aligned} \tag{6}$$

where $\boldsymbol{w} = \begin{bmatrix}\boldsymbol{\eta}_0^T & \boldsymbol{u}_0^T & \dots & \boldsymbol{\eta}_{Np-1}^T & \boldsymbol{u}_{Np-1}^T & \boldsymbol{\eta}_{Np}^T\end{bmatrix}^T \in \mathbb{R}^{5Np+3}$ is a vector of $5Np+3$ decision variables, and $\boldsymbol{g}(\boldsymbol{w}) \in \mathbb{R}^{n_g}$ is a vector of n_g equality constraints.

We define a quadratic objective function:

$$\phi_1(\boldsymbol{w}, \boldsymbol{p}_{d_{1:Np}}) = \sum_{k=0}^{Np-1} \Big(K_p \left\| \boldsymbol{p}_{k+1} - \boldsymbol{p}_{d_{k+1}} \right\|_2^2 + K_U (U_k - U_d)^2 + K_r (r_k - r_d)^2 \Big), \quad (7)$$

where $K_p, K_U, K_r > 0$ are tuning parameters. The desired SOG U_d and ROT r_d can be derived from the desired nominal path, given by the sequence of desired positions $\boldsymbol{p}_{d_{1:Np}} = \begin{bmatrix}\boldsymbol{p}_{d_1} & \boldsymbol{p}_{d_2} & \dots & \boldsymbol{p}_{d_{Np}}\end{bmatrix}$.

When using multiple shooting, one must employ shooting constraints to ensure that the control input and vehicle states satisfy the kinematic model (1). We define an integrator function $\boldsymbol{f}(\boldsymbol{\eta}_k, \boldsymbol{u}_k)$ using Runge-Kutta of order 4:

$$\begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{F}(\boldsymbol{\eta}_k, \boldsymbol{u}_k) \\ \boldsymbol{k}_2 &= \boldsymbol{F}(\boldsymbol{\eta}_k + \frac{h}{2}\boldsymbol{k}_1, \boldsymbol{u}_k) \\ \boldsymbol{k}_3 &= \boldsymbol{F}(\boldsymbol{\eta}_k + \frac{h}{2}\boldsymbol{k}_2, \boldsymbol{u}_k) \\ \boldsymbol{k}_4 &= \boldsymbol{F}(\boldsymbol{\eta}_k + h\boldsymbol{k}_3, \boldsymbol{u}_k) \\ \boldsymbol{f}(\boldsymbol{\eta}_k, \boldsymbol{u}_k) &= \boldsymbol{\eta}_k + \frac{h}{6}\left(\boldsymbol{k}_1 + \mathbf{2}k_2 + \mathbf{2}k_3 + \boldsymbol{k}_4\right), \end{aligned} \quad (8)$$

where h is the discretization time step. Given a state and control input $\boldsymbol{\eta}_k$ and $\boldsymbol{u}_k$, we can now define the vessel state at the next iteration as $\boldsymbol{\eta}_{k+1} = \boldsymbol{f}(\boldsymbol{\eta}_k, \boldsymbol{u}_k)$. We include the shooting constraints in the equality constraints $\boldsymbol{g}(\boldsymbol{w})$ as:

$$\boldsymbol{g}(\boldsymbol{w}) = \begin{bmatrix} \bar{\boldsymbol{\eta}}_0 - \boldsymbol{\eta}_0 \\ \boldsymbol{f}(\boldsymbol{\eta}_0, \boldsymbol{u}_0) - \boldsymbol{\eta}_1 \\ \boldsymbol{f}(\boldsymbol{\eta}_1, \boldsymbol{u}_1) - \boldsymbol{\eta}_2 \\ \vdots \\ \boldsymbol{f}(\boldsymbol{\eta}_{Np-1}, \boldsymbol{u}_{Np-1}) - \boldsymbol{\eta}_{Np} \end{bmatrix}, \quad (9)$$

resulting in $n_g = 3(Np+1)$ equality constraints.

To guarantee that the resulting vehicle trajectory $\boldsymbol{p}(t)$ avoids collisions, we must make sure that the vehicle trajectory $\boldsymbol{p}(t)$ does not intersect the obstacles in the sets $\mathcal{O}_s$ and $\mathcal{O}_m$. In the set $\mathcal{O}_s$, we have S static obstacles. For each obstacle $O_{s_i} = (\boldsymbol{p}_{s_i}, r_{s_i}) \in \mathcal{O}_s$ we define the function:

$$\boldsymbol{h}_{s_i}(\boldsymbol{p}_{1:Np}) = \begin{bmatrix} r_{s_i}^2 - \left\|\boldsymbol{p}_1 - \boldsymbol{p}_{s_i}\right\|_2^2 \\ r_{s_i}^2 - \left\|\boldsymbol{p}_2 - \boldsymbol{p}_{s_i}\right\|_2^2 \\ \vdots \\ r_{s_i}^2 - \left\|\boldsymbol{p}_{Np} - \boldsymbol{p}_{s_i}\right\|_2^2 \end{bmatrix} \in \mathbb{R}^{Np}. \quad (10)$$

Similarly, for each moving obstacle $O_{m_i} = (\boldsymbol{p}_{m_i}(t), r_{m_i}) \in \mathcal{O}_m$ we define the function:

$$\boldsymbol{h}_{m_i}(\boldsymbol{p}_{1:Np}, \boldsymbol{t}_{1:Np}) = \begin{bmatrix} r_{m_i}^2 - \left\|\boldsymbol{p}_1 - \boldsymbol{p}_{m_i}(t_1)\right\|_2^2 \\ r_{m_i}^2 - \left\|\boldsymbol{p}_2 - \boldsymbol{p}_{m_i}(t_2)\right\|_2^2 \\ \vdots \\ r_{m_i}^2 - \left\|\boldsymbol{p}_{Np} - \boldsymbol{p}_{m_i}(t_{Np})\right\|_2^2 \end{bmatrix} \in \mathbb{R}^{Np}, \quad (11)$$

where $\boldsymbol{t}_{1:Np} = \begin{bmatrix}t_1 & t_2 & \dots & t_{Np}\end{bmatrix}^T$ contains the time related to each NLP step. We then define the inequality constraints:

$$\boldsymbol{h}_o(\boldsymbol{w}, \boldsymbol{t}_{1:Np}) = \begin{bmatrix} \boldsymbol{h}_{s_1}(\boldsymbol{p}_{1:Np}) \\ \vdots \\ \boldsymbol{h}_{s_S}(\boldsymbol{p}_{1:Np}) \\ \boldsymbol{h}_{m_1}(\boldsymbol{p}_{1:Np}, \boldsymbol{t}_{1:Np}) \\ \vdots \\ \boldsymbol{h}_{m_M}(\boldsymbol{p}_{1:Np}, \boldsymbol{t}_{1:Np}) \end{bmatrix} \in \mathbb{R}^{(S+M)Np}, \quad (12)$$

which ensure that the resulting trajectory avoids obstacles.

To ensure compliance with the velocity limitations in (3), we can for each control input $\boldsymbol{u}_i$ define the function:

$$\boldsymbol{h}_{u_i}(\boldsymbol{u}_i) = \begin{bmatrix} U_{\min}(r_i) - U_i \\ -(U_{\max}(r_i) - U_i) \\ r_{\min} - r_i \\ -(r_{\max} - r_i) \end{bmatrix} \in \mathbb{R}^4, \quad (13)$$

and form the inequality constraints:

$$\boldsymbol{h}_u(\boldsymbol{w}) = \begin{bmatrix} \boldsymbol{h}_{u_0}(\boldsymbol{u}_0) \\ \boldsymbol{h}_{u_1}(\boldsymbol{u}_1) \\ \vdots \\ \boldsymbol{h}_{u_{Np-1}}(\boldsymbol{u}_{Np-1}) \end{bmatrix} \in \mathbb{R}^{4Np}, \quad (14)$$

which ensure that all control inputs satisfy (3).

Finally, (12) and (14) is combined:

$$\boldsymbol{h}(\boldsymbol{w}, \boldsymbol{t}_{1:Np}) = \begin{bmatrix} \boldsymbol{h}_o(\boldsymbol{w}, \boldsymbol{t}_{1:Np}) \\ \boldsymbol{h}_u(\boldsymbol{w}) \end{bmatrix}, \quad (15)$$

resulting in $n_h = (4 + S + M)\,Np$ inequality constraints. By requiring the functions $U_{\min}(r)$ and $U_{\max}(r)$ to be C^2 in r, we see that $\boldsymbol{h}(\boldsymbol{w}, \boldsymbol{t}_{1:Np})$ is C^2 in $\boldsymbol{w}$.

D. COLREGS compliance

The NLP formulation (6), (7), (9) and (15) will be able to avoid collision with both moving and static obstacles, but will not necessarily obey the COLREGS rules formulated in Section III.

Implementing these rules in an NLP problem is a challenging task, and in this paper we will only investigate the possibility of obeying rule 8, which requires that the vessel maneuvers should be large enough to be observable to other vessels. This implies that applying a sequence of small alterations in course or speed should be avoided. Naturally, there are a number of ways to avoid this, including:

- Constraining the ROT and SOG derivatives to only allow values with magnitude very close to zero or greater than some threshold.
- Including penalty terms in the objective function which penalize course and speed alteration depending on the time to perform it. Slow alterations would then be penalized more than quick alterations.

Initially, it may seem like a good solution to constrain the ROT and SOG derivatives to only allow maneuvers of a given magnitude, avoiding small alterations. However,

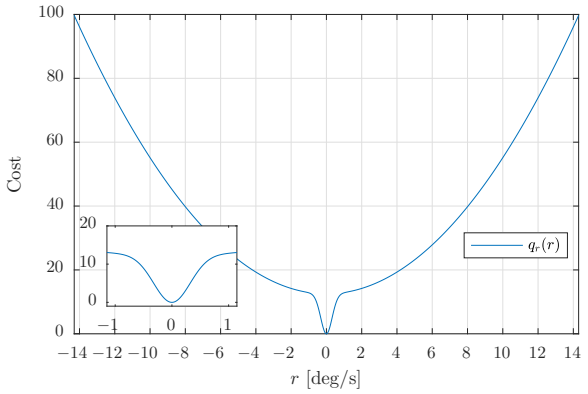

Fig. 4: ROT penalty function with $a_r = 112$, $b_r = 6.25{\cdot}10^{-5}$ and $r_{\max} = 0.25$ rad/s ≈ 14.3 deg/s. The function is non-convex close to the origin, but remains smooth as the zoomed insert shows.

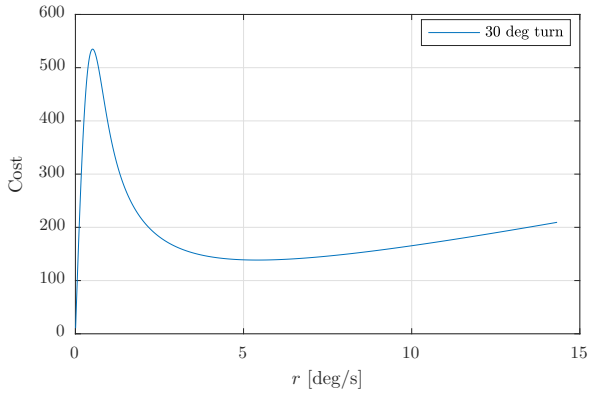

Fig. 5: Cost of performing a 30 deg course alteration, using different ROTs. The peak at $r \approx 0.51$ deg/s is the highest cost, while $r = 5.42$ deg/s is the minimum after the peak.

this would result in a highly non-convex (and even non-connected) search space. Solving an NLP of this form is difficult, and highly dependent on initialization. We therefore investigate the second option with introducing penalty terms in the objective function. Such penalty terms can either be defined on the change of course and SOG $\Delta\psi_k = \psi_{k+1} - \psi_k$ and $\Delta U_k = U_{k+1} - U_k$, for $k = 0 \dots Np-1$, or the ROT and SOG-derivative. Defining the terms on the ROT and SOG-derivative allows for changing the time step length h, without changing the behavior of the penalty terms. Since the SOG-derivative is not part of the NLP (6), we approximate it as $\dot{U}_k \approx \frac{U_{k+1}-U_k}{h}$.

We wish to ensure that the objective function is C^2, hence special considerations must be made when designing the penalty terms. Inspired by the normal distribution, we propose to combine a square and an exponential term as:

$$q(\zeta; a, b) = a\zeta^2 + (1 - \mathrm{e}^{-\frac{\zeta^2}{b}}), \tag{16}$$

where $a, b > 0$ control the shape of the function. We define two functions:

$$\begin{aligned} q_r(r; r_{\max}) &= \frac{100}{q(r_{\max}; a_r, b_r)} q(r; a_r, b_r) \\ q_{\dot{U}}(\dot{U}; \dot{U}_{\max}) &= \frac{100}{q(\dot{U}_{\max}; a_{\dot{U}}, b_{\dot{U}})} q(\dot{U}; a_{\dot{U}}, b_{\dot{U}}), \end{aligned} \tag{17}$$

where the parameters a_r and b_r define the shape of the ROT penalty function $q_r(r)$, and the parameters $a_{\dot{U}}$ and $b_{\dot{U}}$ define the shape of the SOG-derivative penalty term. The parameters $r_{\max} > 0$ and $\dot{U}_{\max} > 0$ are the maximum expected values of r and $\dot{U}$, and are used to scale the functions such that $q_r(r), q_{\dot{U}}(\dot{U}) \in [0, 100]$ for $r \in [-r_{\max}, r_{\max}]$ and $\dot{U} \in [-\dot{U}_{\max}, \dot{U}_{\max}]$.

The ROT penalty function $q_r(r)$ with $a_r = 112$, $b_r = 6.25 \cdot 10^{-5}$ and $r_{\max} = 0.25$ rad/s is shown in Fig. 4. It is obvious that the function is non-convex, but it is C^∞.

Assuming that a course alteration is performed with a constant ROT, the cost of performing a course alteration $\Delta\psi$ using the penalty term can be approximated as $q_r(r)\frac{\Delta\psi}{r}$, where $\frac{\Delta\psi}{r}$ is the time to execute the turn. From Fig. 5 it is clear that for a course alteration of 30 deg, the penalty term motivates choosing a ROT either significantly higher than 0.51 deg/s, or using very long time to execute the turn. Notice that this generalizes to an arbitrary turn, since the cost scales linearly with $\Delta\psi$.

To avoid that the position error in (7) dominates at large errors, we introduce the pseudo-Huber cost function:

$$C(x; \delta) = 2\delta^2 \left(\sqrt{1 + \left(\frac{x}{\delta}\right)^2} - 1 \right), \tag{18}$$

which for small x approximates a quadratic function x^2, and for large x approximates a linear function with slope 2δ [18]. The parameter $\delta > 0$ controls where the function changes from being quadratic to being linear.

Replacing the squared 2-norm position error with the pseudo-Huber cost function and the quadratic ROT and SOG cost with the new penalty terms we get the modified objective function:

$$\begin{aligned} \phi_2(\boldsymbol{w}, \boldsymbol{p}_{d_{1:Np}}) = \sum_{k=0}^{Np-1} \Big(& \frac{K_p}{2} C\left(\left\| \boldsymbol{p}_{k+1} - \boldsymbol{p}_{d_{k+1}} \right\|_2 ; \delta \right) \\ & + K_{\dot{U}} q_{\dot{U}}(\dot{U}_k) + K_r q_r(r_k) \Big). \end{aligned} \tag{19}$$

Notice that the desired SOG and ROT are implicitly included in the position error term. We select $\delta = 1$ and scale the position error by $1/2$ to get a slope of 1 for large position errors. This can, however, also be treated as a tuning parameter in the problem.

V. Simulation Results

We simulate a scenario with two static and one moving obstacle to show the resulting behavior of the mid-level COLAV algorithm, given the two objective functions (7) and (19). The mid-level COLAV algorithm is implemented as an MPC algorithm, where only the first step of the optimal solution is implemented.

To define and solve the NLP (6) with (9), (15), and the quadratic and modified objective functions (7) and (19), we

TABLE I: Simulation and tuning parameters.

Param.	Value	Comment
Ns	55	Number of simulation steps
h	10 s	Step size
Np	24	Prediction steps
$U_{\min}$	0 m/s	Minimum SOG
$U_{\max}$	17 m/s	Maximum SOG
Quadratic objective function ϕ_1:		
K_p:	$2/3 \cdot 10^{-4}$ 1/m^2	Position error scaling
K_U:	$1 \cdot 10^{-1}$ s^2/m^2	SOG error scaling
K_r:	$3 \cdot 10^{2}$ 1/rad^2	ROT error scaling
Modified objective function ϕ_2:		
K_p	$4 \cdot 10^{-2}$	Position error scaling
$K_{\dot{U}}$	$6 \cdot 10^{-1}$	SOG-derivative penalty term scaling
K_r:	$5 \cdot 10^{-1}$	ROT penalty term scaling
$[a_{\dot{U}}, b_{\dot{U}}]$	$[8, 2.5 \cdot 10^{-4}]$	SOG-derivative penalty term parameters
$[a_r, b_r]$	$[112, 6.25 \cdot 10^{-5}]$	ROT penalty term parameters

used the CASADI framework (v3.1.0-rc1 for MATLAB) [15] with the IPOPT solver [14]. From the parameterization in Section IV is is clear that the NLP problem is non-convex. IPOPT is however able to handle non-convex optimization.

Simulation and tuning parameters are shown in Table I. The desired nominal trajectory $\boldsymbol{p}_d(t)$ is generated by a set of waypoints and a desired speed along the path $U_d = 10$ m/s. We start the simulation with $\boldsymbol{\eta}_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, and initialize the first iteration of the MPC with the desired nominal trajectory, and SOG and ROT as zero. Hence:

$$\boldsymbol{w}_0 = \begin{bmatrix} \boldsymbol{\eta}_0^T & \boldsymbol{u}_0^T & \boldsymbol{p}_{d_1}^T & \dots & \boldsymbol{u}_0^T & \boldsymbol{p}_{d_{Np}}^T \end{bmatrix}^T, \tag{20}$$

with $\boldsymbol{u}_0 = \begin{bmatrix} U_d & 0 \end{bmatrix}^T$. Notice that the initial guess is infeasible if the desired nominal trajectory intersects with obstacles. However, IPOPT does not require a feasible initial guess. Later iterations of the MPC is initialized with the solution of the previous iteration.

The simulations are run in MATLAB R2016b, on a 2.8 GHz Intel Core i7 processor. The first iteration of the MPC with the quadratic and modified objective functions takes approximately 250 ms and 750 ms to solve, respectively. Later iterations with warm start take approximately 15–25 ms for the quadratic objective function, and 20–150 ms for the modified objective function. Hence, with a time step of 10 s, we consider the algorithm with the modified objective function as implementable in real time. The NLP with the modified objective function has more local minima than the NLP with the quadratic objective function, which is why the runtime with the modified objective function is less consistent than when using the quadratic objective function. Guaranteeing a maximum computational time for NLPs is difficult, but in the event that the optimization problem is not solved within the required time, the reactive COLAV layer would still keep the vessel on a collision-free path, although possibly getting stuck in a local minima.

Figures 6 and 7 show the resulting vessel trajectory using the quadratic and modified objective functions, respectively. The vessel trajectory is shown in red, while the colored

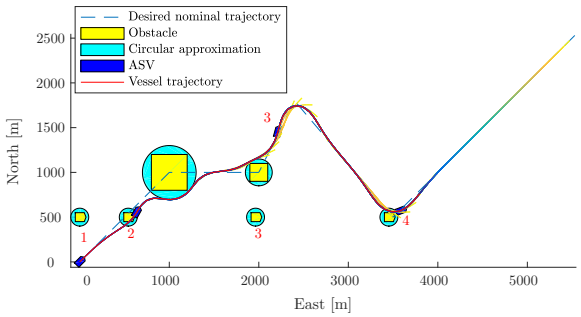

Fig. 6: Simulation result using the quadratic objective function ϕ_1. Note that the size of the ownship (in blue) is overexaggerated for visibility. Hence, what appears to be a collision with the moving obstacle is in fact not a collision.

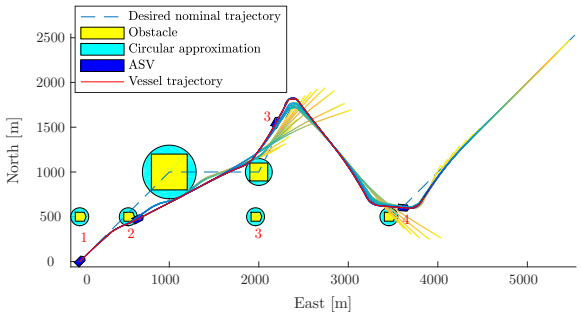

Fig. 7: Simulation result using the modified objective function ϕ_2. Note that the size of the ownship (in blue) is overexaggerated for visibility. Hence, what appears to be a collision with the moving obstacle is in fact not a collision.

lines show the predicted trajectory in each iteration with blue and yellow in the start and end of the predicted trajectory, respectively. Furthermore, the numbers 1–4 mark time instants, where $t_1 = 0$ s, $t_2 = 80$ s, $t_3 = 290$ s and $t_4 = 510$ s. We clearly see that the modified objective function results in a trajectory with clear and observable course changes in contrast to the quadratic objective function. This is also confirmed from the ROT in Fig. 8 which shows the input sequences. We also see that the modified objective function produces a more observable SOG trajectory than the quadratic objective function.

Both objective functions produce collision-free trajectories, and avoid the first encounter with the moving obstacle by passing in front of it at $t \approx 80$ s. An interesting observation is that with respect to COLREGS, our vessel should actually keep the course and speed constant here, and not attempt to avoid collision at an early stage, since our vessel is deemed the stand-on vessel in this situation. In the second encounter with the moving obstacle at $t \approx 500$ s, both algorithms turn to port to avoid collision.

VI. Conclusion and Further Work

We have designed and implemented an NLP-based mid-level COLAV algorithm, enabling avoidance of both static and moving obstacles. The algorithm is simulated with two

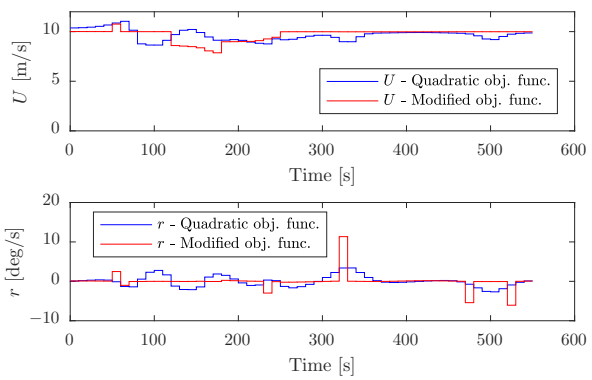

Fig. 8: SOG and ROT trajectories when using the quadratic and modified objective functions.

different objective functions, where one is designed to ensure compliance with rule 8 of COLREGS, thus providing observable course and speed changes. The simulation results are promising and motivate for further development to include other essential COLREGS rules. Based on the simulations, the NLP is considered to be real-time implementable with respect to computational requirements.

Possibilities for further work include:

- Designing objective function terms able to capture rules 13-15 and 17 of COLREGS. These terms can be aided by a support function, see Fig. 1, to encode which rules are applicable to each vessel in a given situation.
- Combining the developed mid-level COLAV algorithm with a reactive algorithm, e.g. continuing the work on the dynamic window algorithm in [5].

ACKNOWLEDGMENT

This work was supported by the Research Council of Norway through project number 244116 and the Centres of Excellence funding scheme with project number 223254.

REFERENCES

[1] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles", *International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

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

[3] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance", *IEEE Robotics & Automation Magazine*, vol. 4, pp. 23–33, 1997.

[4] P. Ögren and N. Leonard, "A convergent dynamic window approach to obstacle avoidance", *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 188–195, 2005.

[5] B.-O. H. Eriksen, M. Breivik, K. Y. Pettersen, and M. S. Wiig, "A modified dynamic window algorithm for horizontal collision avoidance for AUVs", in *Proc. of IEEE CCA*, Buenos Aires, Argentina, 2016.

[6] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning", Computer Science Dept., Iowa State University, Tech. Rep., 1998.

[7] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths", *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[8] A. Stentz, "The focussed D* algorithm for real-time replanning", in *Proc. of the International Joint Conference on Artificial Intelligence*, Pittsburgh, Pennsylvania, USA, 1995.

[9] Ø. A. G. Loe, "Collision avoidance for unmanned surface vehicles", Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2008.

[10] T. A. Johansen, T. Perez, and A. Cristofaro, "Ship collision avoidance and COLREGS compliance using simulation-based control behavior selection with predictive hazard assessment", *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3407–3422, 2016.

[11] G. Casalino, A. Turetta, and E. Simetti, "A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields", in *Proc. of Oceans 2009-Europe*, Bremen, Germany, 2009.

[12] T. Shim, G. Adireddy, and H. Yuan, "Autonomous vehicle collision avoidance system using path planning and model-predictive-control-based active front steering and wheel torque control", *Proceedings of the Institution of Mechanical Engineers, part D: Journal of Automobile Engineering*, vol. 226, no. 6, pp. 767–778, 2012.

[13] B. Yi, S. Gottschling, J. Ferdinand, N. Simm, F. Bonarens, and C. Stiller, "Real time integrated vehicle dynamics control and trajectory planning with MPC for critical maneuvers", in *IEEE Intelligent Vehicles Symposium*, Gothenburg, Sweden, 2016.

[14] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming", *Mathematical Programming*, vol. 106, pp. 25–57, 2005.

[15] J. Andersson, "A general-purpose software framework for dynamic optimization", PhD thesis, Arenberg Doctoral School, KU Leuven, Heverlee, Belgium, 2013.

[16] B.-O. H. Eriksen and M. Breivik, "Modeling, identification and control of high-speed asvs: Theory and experiments", in *Sensing and control for autonomous vehicles: Applications to land, water and air vehicles*, T. I. Fossen, K. Y. Pettersen, and H. Nijmeijer, Eds. Cham: Springer International Publishing, 2017, pp. 407–431.

[17] A. N. Cockcroft and J. N. F. Lameijer, *A guide to the collision avoidance rules*. Elsevier, 2004.

[18] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, 2003.