IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 September 2022, accepted 14 October 2022, date of publication 19 October 2022, date of current version 26 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3215654

RESEARCH ARTICLE

Ship Collision Avoidance and Anti Grounding Using Parallelized Cost Evaluation in Probabilistic Scenario-Based Model Predictive Control

TRYM TENGESDAL[1,2], TOR ARNE JOHANSEN[1,2], (Senior Member, IEEE), TOM DANIEL GRANDE[3], AND SIMON BLINDHEIM[1,2]

[1]Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), 7034 Trondheim, Norway
[2]Center for Autonomous Marine Operations and Systems (AMOS), Norwegian University of Science and Technology (NTNU), 7034 Trondheim, Norway
[3]Department of IT, Analysis and Control, Norwegian Tax Administration, 0663 Oslo, Norway

Corresponding author: Trym Tengesdal (trym.tengesdal@ntnu.no)

This work was supported in part by the Research Council of Norway, through the Online Risk Management and Risk Control for Autonomous Ships (ORCAS), under Grant 280655; in part by the Centre of Excellence Funding Scheme, Centre for Autonomous Marine Operations and Systems (AMOS), under Project 223254; and in part by the Autoship Centre for Research-Based Innovation, under Project 309230.

ABSTRACT The ability to effectively process large amounts of information in reasonable time will be important for robust deliberative collision avoidance (COLAV) planning algorithms. Failure to do so can lead to collision, and can be compared to lack of proper supervision from officers on watch (OOW). The main contribution in this article is a parallelized implementation of the Probabilistic Scenario-Based Model Predictive Control (PSB-MPC) on a Graphical Processing Unit (GPU) platform which incorporates both dynamic obstacle avoidance and anti-grounding. Simulation results demonstrate that the COLAV planner can produce collision-free trajectories with respect to grounding hazards and nearby vessels at relatively low computational cost, and which also comply to the COLREGS when deemed possible. Corresponding run-time results show that the algorithm utilizing parallel processing performs better than the alternative for increasing numbers of own-ship control behaviours, nearby static and dynamic obstacles, and dynamic obstacle prediction scenarios considered.

INDEX TERMS Maritime collision avoidance, parallel processing, CUDA, autonomous ships, model predictive control.

I. INTRODUCTION

A. BACKGROUND

Autonomous ships will require a high level of data processing in order to have adequate situational awareness and to make deliberate decisions. This requires efficient and robust algorithms, and well chosen platforms to enable fast computation. When facing a hazardous situation in e.g. confined space with multiple static and dynamic obstacles, the need to evaluate a larger set of future control behaviours or trajectories for the autonomous ship and other obstacles will be necessary, such that a risk minimizing or collision-free trajectory is possible to find. Furthermore, the collision-free planned trajectory should comply to the Convention on the International Regulations for Preventing Collision at Sea (COLREGS) [1] when possible. However, evaluating the risk associated in any of these control behaviours can be computationally expensive. Thus, to meet run-time requirements, a collision avoidance (COLAV) planning algorithm which scales well in the evaluation of different control behaviours will be both beneficial and necessary in such cases. Increased robustness can then also result as a consequence of being able to evaluate more vessel behaviour scenarios and situational information in the system at run-time.

A Scenario-Based Model Predictive Control (SB-MPC) [2] approach is here a viable option that can incorporate most of the elements needed in a robust COLAV planning algorithm, such as anti-grounding, dynamic obstacle avoidance

The associate editor coordinating the review of this manuscript and approving it for publication was Shaohua Wan.

and multi-ship adherence to COLREGS when possible. This is because of its flexibility in the formulation of its optimization problem, with different control objectives and possible integration of constraints, and which has a rich theoretical foundation. The sampling-based method is also flexible in the prediction models used to generate own-ship and dynamic obstacle prediction scenarios. The problem with this approach however, and especially for the probabilistic version (PSB-MPC) [3], is that the optimization problem in the COLAV planning algorithm scales poorly with an increasing set of considered own-ship avoidance maneuvers, static obstacles and dynamic obstacles with their own alternative prediction scenarios. The prediction of the collision risk with respect to all dynamic and uncertain obstacles involved, and calculating distances to all static obstacles for anti-grounding purposes, has exponentially increasing computational cost as the optimization problem increases.

B. LITERATURE REVIEW

Many studies on maritime collision avoidance exists today, and are mainly summarized in review papers such as [4], [5], [6], and [7], whereas we here focus on deliberative COLAV planning methods having dynamic obstacle avoidance and COLREGS adherence in addition to anti-grounding in their algorithms. For a general overview on planning algorithms, see [8].

In this article, deliberative refers to the COLAV algorithm planning efficient trajectories that adheres to the COLREGS when deemed possible, and avoid collision well before risky situations occur. Following the COLREGS blindly in any type of situation will not be sufficient, as was shown in [9] and also discussed in [10] and [11]. Thus, the deliberate COLAV planning algorithm should in general also consider the intention uncertainties of nearby dynamic obstacles. Further note that with COLREGS compliance, we mean compliance with the COLREGS rules 8, 13 - 17 on taking early and apparent action, and the correct action in overtaking, head-on and crossing situations with either give-way or stand-on obligations, respectively. These are the rules most relevant and common to consider for automatic COLAV planning. However, a complete COLAV system should consider the full rule set.

A lattice-based trajectory planner using A* search for finding collision-free trajectories is introduced in [12], where non-adherence to the COLREGS, trajectory deviation and collision risk with respect to static and dynamic obstacles is penalized in the cost function. An intention based motion model is used for dynamic obstacles, which relies on learning the positional prediction uncertainty for a given scene when used in calculating collision probabilities. The details on this model is not given, and results on how the planner scales in run-time with increasing lattice grid density, dynamic and static obstacles are however not given.

The work in [13] introduces a hierarchical system with three levels. The top level trajectory planner uses lattice-based A* search combined with an Optimal Control Problem (OCP) method for generating collision-free trajectories with respect to static obstacles. A mid level MPC-based COLAV planning algorithm modifies this trajectory to adhere to the COLREGS and avoid collisions with respect to dynamic obstacles. Lastly, a low-level reactive COLAV sampling-based planning algorithm acts as a fail-safe in case the levels above can not handle the situation. The system does however assume straight line trajectories for dynamic obstacle predictions without uncertainty, which does not coincide with real-time vessel behaviour in hazardous situations. Furthermore, scalability and run-time properties with an increasingly complex situation is not discussed.

In [14], a field-test verified A-star search trajectory planner is developed, which attempts to find a COLREGS-compliant and collision-free trajectory with respect to dynamic and static obstacles in a lattice. To predict nearby dynamic obstacle trajectories, the planner employs Monte-Carlo (MC) simulation using fuzzy logic and the trajectory history of the obstacle to find a set of probable trajectories, where the most probable one is considered for collision avoidance. As in [13], the method does not consider the prediction uncertainty associated with dynamic obstacles. Furthermore, the computational efficiency of the planner only tested for a set of 500 possible own-ship trajectories and one expected dynamic obstacle trajectory, which can be inadequate in highly congested scenarios.

Candeloro et. al. [15] propose a global and local lattice-based trajectory planner which uses Voronoi Diagrams to generate a set of static obstacle collision-free waypoints, from where a continuous trajectory is generated using Fermat's Spiral. The method considers local replanning windows for taking detected dynamic and static obstacles into account, and predicts dynamic obstacle motion with the Constant Velocity (CV) model [16]. A convex hull representing the dynamic obstacle uncertainty up until time to Closest Point of Approach (CPA) is created from using the position estimates and error covariances from a Kalman filter, which is then regarded as an area to avoid in the planner. This may however be overly conservative, due to the unrealistic uncertainty growth in the CV model [17]. How the local replanning run-time scales with increasing windows size, dynamic and static obstacles is not considered.

Nonlinear MPC for static and dynamic obstacle collision avoidance with environmental disturbance rejection was proposed in [18]. A deterministic CV model was used for the dynamic obstacle prediction, which will not be the case in real-time hazardous maritime situations where ships will maneuver. Furthermore, how the MPC scales with static and dynamic obstacles was not considered.

Chiang and colleagues [19] introduces a sampling-based static and dynamic obstacle considerate trajectory planner with COLREGS-compliant COLAV planning algorithm based on Rapidly exploring Random Trees (RRT), where a joint simulator is used to predict both the own-ship and dynamic obstacle motion. Potential fields are used in the prediction to ensure that all the vessels have collision-free trajectories with respect to each other and static obstacles.

The method is shown to have beneficial run-times feasible for real-time. However, the underlying assumption in the prediction is however that ships will always perform deterministic COLREGS-compliant maneuvers if possible, which is not necessarily true in practice.

Collision avoidance within a distributed flocking control strategy based on MPC was considered in [20], with respect to nearby dynamic vehicles in the flock and static obstacles. The computational efficiency or scalability of the method was however not discussed, and the states of all vehicles involved are assumed deterministic.

C. CONTRIBUTIONS

In this paper, an implementation of the sampling-based PSB-MPC algorithm on a GPU platform which facilitates efficient anti-grounding and dynamic obstacle avoidance is introduced. The main contribution of the article compared to current state-of-the-art static and dynamic obstacle COLAV planning algorithms is the description of a parallelization algorithm for efficient cost evaluation of possible own-ship trajectories in the PSB-MPC, taking into account dynamic obstacle uncertainties and complex static obstacles in maritime hazardous situations. The algorithm is feasible for real-time, as the MPC cost function evaluation scales linearly with increasing numbers of dynamic obstacles with their own prediction scenarios and also static obstacles, due to the parallelization. Static obstacles are read in from Electronic Navigational Chart (ENC) data and processed into simplified polygons using the Ramer-Douglas-Peucker (RDP) algorithm [21]. The efficiency of the parallelized implementation makes it possible for the COLAV planning algorithm to consider more dynamic obstacle prediction scenarios and own-ship trajectories, and more complex static obstacle maps for elevated situational awareness and better trajectory planning. Furthermore, a side contribution of the article is that the dynamic obstacle prediction scheme in [3] is updated to use a kinematic model with incorporated Line-of-Sight (LOS) guidance for more realistic trajectories.

D. ARTICLE STRUCTURE

The article is organized as follows. Section II gives background information about the PSB-MPC, with prediction models, cost function structure and grounding hazard extraction and representation. An outline of a sequential implementation of the algorithm is also given. A parallelized implementation of the PSB-MPC is given in Section III. Finally, Section IV show simulation results with the PSB-MPC, and Section V concludes the work.

II. THE PSB-MPC COLAV PLANNING ALGORITHM

The Probabilistic Scenario-based Model Predictive Control (PSB-MPC) is an optimization-based COLAV planning method that samples a finite set of possible own-ship trajectories, represented by control behaviours. This is illustrated in Fig. 1, where we note that the control behaviours selected are arbitrary.

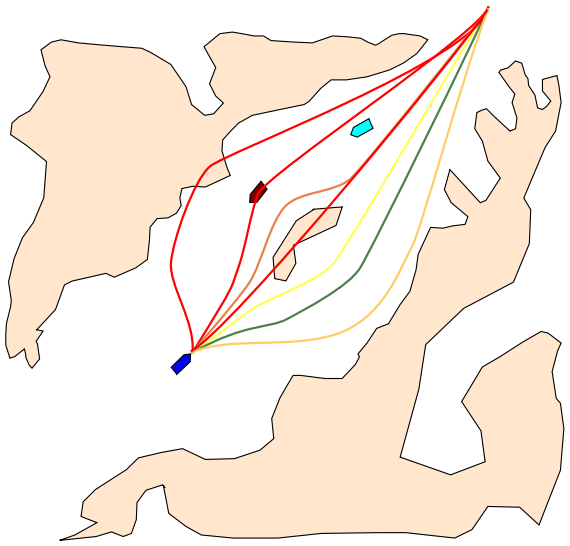


FIGURE 1. PSB-MPC illustration, with the own-ship running the algorithm in blue. Nearby dynamic obstacles are shown in cyan and brown. Grounding hazards are shown in beige. Candidate control behaviours predicted in the MPC are also shown, where the color from red to green represents their cost, with green being the lowest. Thus, the green candidate trajectory is the optimal one. The nominal trajectory goes straight north-east through the confined environment.

Formally, a control behaviour l in the PSB-MPC represents a sequence $[(U^l_{m,1}, \chi^l_{m,1}), \ldots, (U^l_{m,n_M}, \chi^l_{m,n_M})]$ consisting of speed multiplicative factors U_m and additive course angle offsets χ_m. The sequence represent n_M sequential avoidance maneuvers. The parameter n_M can in general be a variable, but will be considered fixed in this work. The sequence of speed and course modifications are applied to the autopilot references U_d and χ_d in speed and course angle at different time steps in the finite prediction horizon, which in turn generates a specific own-ship trajectory. Each control behaviour is evaluated by a cost function $\mathcal{H}^l(\cdot)$, which penalizes probabilistic collision risk, grounding risk, COLREGS violation and nominal trajectory deviation. The optimal one is selected as

$$l^*(t_0) = \arg\min_l \mathcal{H}^l(t_0) \tag{1}$$

where t_0 is the current time, and it is the first avoidance maneuver represented by $U^l_{m,1}$ and $\chi^l_{m,1}$ that is applied by the autopilot through the modified guidance references $U_c = U^{l^*}_{m,1} \cdot U_d$ and $\chi_c = \chi^{l^*}_{m,1} + \chi_d$. The open loop optimization based on (1) is repeated at regular intervals to account for more information in a moving horizon fashion as is common in MPC, and thus closes the loop. More discussion around feasibility and constraint satisfaction in the PSB-MPC can be found in [22].

A. PREDICTION MODELS

For deliberate COLAV planning algorithms, the own-ship prediction model can be selected to be complex or simple depending on how much vessel information one has.

The PSB-MPC can easily handle complex ship motion models in its framework. However, as the kinematic uncertainty associated with the ship motion prediction increases substantially with time, having a simple model to capture the approximate own-ship behaviour is often adequate for deliberative COLAV planning algorithms, where the low-level vessel control systems (autopilot) can compensate for model inaccuracies and disturbances. As predictions of vessel motions over longer time horizons are inherently uncertain, due to environmental disturbances, future maneuvering decisions and unforeseen events, especially in hazardous situations, we argue that there is limited gain in using an overly complex model. On the other hand, for reactive collision avoidance methods and lower level motion control with shorter prediction horizons, it will be more important to consider the ship dynamics accurately. Thus, as the PSB-MPC is flexible in the choice of the own-ship prediction model, it will not be described here but in the simulation study in Section IV.

Therefore, the following text will detail the model used for dynamic obstacles. To create trajectories simulating the ship motion for the own-ship and dynamic obstacles forward in time, we use Euler's method for numerical integration. Specifically, the integration is done over the prediction horizon with discrete predicted times $t_k \in D(t_0) = \{t_0, \ldots, t_0 + k\Delta_{mpc}, \ldots, T_{mpc}\}$, with Δ_{mpc} as the time step and T_{mpc} as the prediction horizon.

As one most often do not have information on the underlying dynamic obstacle vessel or object, their motion models should be simple. The preliminary PSB-MPC used the Ornstein-Uhlenbeck (OU) process [23] in order to predict the motion of dynamic obstacles, and allows for alternative obstacle prediction scenarios [24]. However, the trajectories only specify a single change in course, and are thus not necessarily realistic. A more realistic approach as shown in Fig. 2 is now used, where more avoidance like maneuvers are used.

The predicted obstacle motion is implemented using the following kinematic model

$$\begin{aligned} x^i_{k+1} &= x^i_k + U^i_k cos(\chi^i_k) \\ y^i_{k+1} &= y^i_k + U^i_k sin(\chi^i_k) \\ \chi^i_{k+1} &= \chi^i_k + \frac{1}{T_\chi}(\chi^i_{d,k} - \chi^i_k) \\ U^i_{k+1} &= U^i_k + \frac{1}{T_U}(U^i_{d,k} - U^i_k) \end{aligned} \tag{2}$$

where the superscript i is used for dynamic obstacles. The above kinematic model is combined with Line-of-Sight (LOS) guidance [25] to predict the following of a nominal obstacle path parameterized by n_{wps} waypoints WPS: $[\boldsymbol{p}_1, \ldots, \boldsymbol{p}_z, \ldots, \boldsymbol{p}_{n_{wps}}]$, where $\boldsymbol{p}_z = [x^{wp}_z, y^{wp}_z]^T$ is waypoint $z \in \{1, 2, \ldots, n_{wps}\}$. In the case of straight line paths, the LOS guidance method considers waypoint segments from $\boldsymbol{p}_z$ to $\boldsymbol{p}_{z+1}$, and finds the path tangential angle

$$\alpha_z = atan2(y^{wp}_{z+1} - y^{wp}_z, x^{wp}_{z+1} - x^{wp}_z) \tag{3}$$

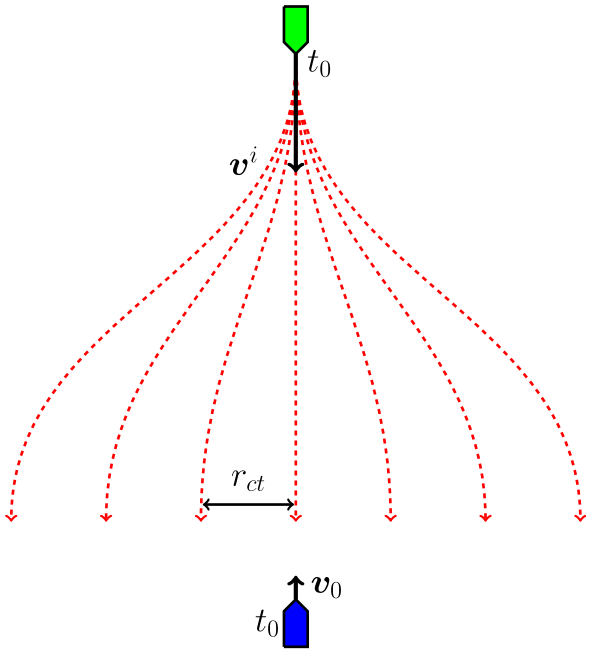


FIGURE 2. Head-on scenario with obstacle i in green and own-ship in blue. Their velocity vectors v^i and v_0, respectively, are also shown. The updated prediction scheme using LOS guidance allows for the obstacle to make realistic alternative maneuvers to port and starboard. The stationary time spacing between trajectories is determined by r_{ct}.

and path-fixed frame referenced path deviation $\boldsymbol{\epsilon}_k$ with rotation α_z as

$$\boldsymbol{\epsilon}^i_k = \boldsymbol{R}^T_{\alpha_z}(\boldsymbol{p}^i_k - \boldsymbol{p}_z) \tag{4}$$

where $\boldsymbol{p}^i_k = [x^i_k, y^i_k]^T$ is the position of obstacle i at time t_k. The rotation matrix $\boldsymbol{R}_{\alpha_z}$ is given by

$$\boldsymbol{R}_{\alpha_z} = \begin{bmatrix} cos(\alpha_z) & -sin(\alpha_z) \\ sin(\alpha_z) & cos(\alpha_z) \end{bmatrix} \tag{5}$$

The along-track error s^i_k and cross-track error e^i_k, see [26] for more details, makes up the path deviation $\boldsymbol{\epsilon}^i_k = [s^i_k, e^i_k]^T$, where the latter error is used with (3) to calculate the desired obstacle COG as

$$\chi^i_{d,k} = \alpha_z + arctan\left(-\frac{e^i_k}{\Delta^i}\right) \tag{6}$$

with Δ^i as the lookahead distance, dependent on the obstacle ship type. See [25] for illustrations and more information. The combination of a kinematic model used with LOS guidance allows for a lightweight prediction of alternative dynamic obstacle maneuvering scenarios. By also specifying a speed profile through a desired SOG $U^i_{d,k}$ in addition to the LOS guidance, one goes from path-generation to trajectory generation for the dynamic obstacles [26].

The PSB-MPC normally gets dynamic obstacle information from the tracking system, where their state estimates have an associated kinematic uncertainty, typically represented through a covariance matrix $\boldsymbol{P}^i(t_0)$. The obstacle kinematic uncertainty is here predicted forward in time

heuristically using an OU-process [17] with mean velocity taken as the current state estimated velocity:

$$\boldsymbol{P}_{k+1}^{i} = \boldsymbol{P}_{0}^{i} + \boldsymbol{\Sigma}_1 \circ \boldsymbol{\Sigma}_2(t_{k+1} - t_0) \tag{7}$$

with $\boldsymbol{P}_k^i$ as the predicted covariance and

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} \frac{\sigma_x^2}{\gamma_x^3} & \frac{\sigma_{xy}}{\gamma_x\gamma_y} & \frac{\sigma_x^2}{2\gamma_x^2} & \frac{2\sigma_{xy}}{\gamma_x} \\ \frac{\sigma_{xy}}{\gamma_x\gamma_y} & \frac{\sigma_y^2}{\gamma_y^3} & \frac{2\sigma_{xy}}{\gamma_y} & \frac{\sigma_y^2}{2\gamma_y^2} \\ \frac{\sigma_x^2}{2\gamma_x^2} & \frac{2\sigma_{xy}}{\gamma_y} & \frac{\sigma_x^2}{\gamma_x} & \frac{2\sigma_{xy}}{\gamma_x + \gamma_y} \\ \frac{2\sigma_{xy}}{\gamma_x} & \frac{\sigma_y^2}{2\gamma_y^2} & \frac{2\sigma_{xy}}{\gamma_x + \gamma_y} & \frac{\sigma_y^2}{\gamma_y} \end{bmatrix} \tag{8}$$

as the stationary process noise part of the process, with σ_x, σ_{xy} and σ_y as the OU model Wiener process noise parameters. The parameters γ_x and γ_y are the reversion strength parameters, which determines the convergence rate of the OU-process towards its mean velocity. The expression for $\boldsymbol{\Sigma}_2(t_{k+1} - t_k)$ can be found in [23]. The symbol $\circ$ in (7) denotes the Hadamard product. The reason behind the usage of the OU-process for uncertainty prediction is its more limited growth in covariance compared to e.g. using a CV model [23]. The 3σ positional uncertainty is heuristically bounded by r_{ct} in the prediction, such that each obstacle trajectory has a tube uncertainty with approximate radius r_{ct}. As for the own-ship, the parameters for the obstacle prediction are all dependent on the type of ship, ship control system, ship captain etc., and should be estimated using available data about the obstacle.

In this article we assume that the nominal obstacle trajectory is a straight line from its current course, and create waypoints on this line. Vessel to vessel communication or e.g. road map methods [27] may be used to predict the nominal obstacle trajectories in more confined spaces where the straight line trajectory assumption is restrictive. An illustration of the uncertainty prediction together with the dynamic obstacle trajectory is shown for a case with a non-straight line obstacle trajectory in Fig. 3.

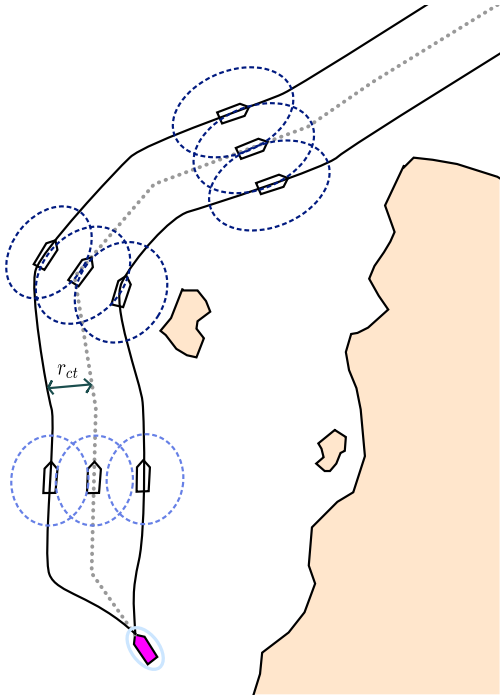


FIGURE 3. Dynamic obstacle prediction illustration with an obstacle in purple. Three prediction scenarios are shown, all starting at t_0, where the vessel is depicted in full purple with its tracked estimation error covariance represented around it as a 3σ probability ellipse in light blue. The nominal predicted obstacle trajectory is shown with the grey dotted line, whereas the alternative scenarios are spaced r_{ct} apart.

B. GROUNDING HAZARDS

The grounding hazards considered in the PSB-MPC are parameterized as two-dimensional polygons. In this article, polygons are read in from shapefiles using the C based library Shapefile C Library, which are generated using the Electronic Navigational Chart processing module in seacharts corresponding to the relevant map region considered [28]. If real-time sensor data is available, this can also be used to update the polygons used in the MPC.

Because electronic map data can have high accuracy, larger polygons extracted can have tens of thousands of vertices. However, for collision avoidance, this level of detail is not necessary, and the polygons should thus be simplified in order to save computation time in the algorithm. One of the earliest and most common curve simplification methods that can be used for polygon simplification is the RDP algorithm [21]. The method recursively simplifies a curve of points by consecutively considering its line segments, pruning away points which are further away from the considered line segment than a specified threshold ϵ_{rdp}. The distance tolerance parameter ϵ should be chosen as not to overly simplify the polygons, preserving as much structure as possible. The method is summarized in Algorithm 1. A graphical illustration of the algorithm can be found in https://en.wikipedia.org/wiki/Ramer-Douglas-Peucker_algorithm. The distance from the own-ship center to nearby polygons is used in the PSB-MPC grounding cost. It is obtained by using a point to polygon calculation method [29], using the ray intersection method for determining if the own-ship center point is inside the polygon, which is suitable for both convex and concave polygons.

C. COST FUNCTION REFORMULATION

We consider the following restructuring of the PSB-MPC cost function

$$\mathcal{H}^l(t_0) = \mathcal{H}_{do}^l + \mathcal{H}_{colregs}^l + \mathcal{H}_{so}^l + \mathcal{H}_{p}^l \tag{9}$$

for a control behaviour l, where the four terms are the cost associated with dynamic obstacles, COLREGS violation, static obstacles or grounding hazards and trajectory tracking,

Algorithm 1 The Ramer-Douglas-Peucker Curve Simplification Algorithm

function RDP($Points$, ϵ_{rdp})
  $d_{max} \leftarrow 0, j \leftarrow 1, end \leftarrow length(Points)$.
  for $i = 2, .., end$ **do**
    $d \leftarrow perpendicularDistance(Points[i], Points[1], Points[end])$
    if $d > d_{max}$ **then**
      $j \leftarrow i, d_{max} \leftarrow d$
    end if
  end for
  if $d_{max} > \epsilon_{rdp}$ **then**
    $rResults1$ =RDP($Points[1, .., j], \epsilon$)
    $rResults2$ =RDP($Points[j, .., end], \epsilon$)
    $newPoints = \{rResults1[1, .., length(rResults1) - 1], rResults2\}$
  else
    $newPoints = \{Points[1], Points[end]\}$
  end if
  return $newPoints$
end function

respectively. The dynamic obstacle related cost is here reformulated to

$$\mathcal{H}_{do}^{l} = \sum_{i=1}^{n_{do}} w^i \mathcal{H}_{do}^{l,i} \tag{10}$$

where w^i represent the weight of the cost from obstacle i, in general influenced by factors such as distance, bearing, nearby grounding hazards and vessel-vessel communication. If no prior information is used, it is set to $w^i = 1$. The dynamic obstacle i cost is given by

$$\mathcal{H}_{do}^{l,i} = \sum_{s=1}^{n_{ps}^i} \mathbb{P}_s^i C_s^{l,i} \tag{11}$$

where n_{ps}^i is the number of prediction scenarios for the obstacle, $\mathbb{P}_s^i$ represent the associated prediction scenario probabilities from an intention inference module [24], and $C_s^{l,i}$ is the cost involving prediction scenario s for obstacle i, given as

$$C_s^{l,i} = \max_k \zeta_i \mathcal{C}_{i,k}^{l,s} \hat{\mathbb{P}}_{c,k}^{l,i,s} exp(-t_k/T_d) \tag{12}$$

which is taken as the maximum of the probabilistic collision risk, involving the relative kinetic energy term $\mathcal{C}_i^{l,s}(t) = K_{coll}||\boldsymbol{v}_k^i - \boldsymbol{v}_k||^2$ between the obstacle i velocity $\boldsymbol{v}_k^i$ and own-ship velocity $\boldsymbol{v}_k$, with parameter K_{coll}, [2]. $\hat{\mathbb{P}}_{c,k}^{l,i,s}$ is the collision probability estimate calculated using the Cross-Entropy method, see [3] for more details. The track loss modifier ζ_i, [30] takes into account cases when dynamic obstacle tracks are lost for some time. Lastly, an exponential discounting term with time constant T_d gives lower weighting of collision events far ahead in the future.

The intention uncertainty of a dynamic obstacle is represented through the scenario probabilities $\mathbb{P}_s^i$ for each considered obstacle prediction scenario. Given a representable set of obstacle prediction scenarios, we are able to cover most anticipated obstacle maneuvering cases because we predict the uncertainty for each of the scenarios. These probabilities of different target ship plans or trajectories are typically inferred by an intention model as in [24] and [31], and can be used for having elevated situational awareness in the planner. Furthermore, the probabilities are an adequate way of taking into account intention information, as they are easy to interpret, can be used to define risk and leads to a natural way of weighting the collision risk associated with different decision candidates for an obstacle ship. The downside is that one needs a validated intention inference model, and a sufficient set of dynamic obstacle prediction scenarios in order to have meaningful estimates.

To favor COLREGS compliance in multi-ship situations, the COLREGS related cost is now separated into its own term in the PSB-MPC, and given as

$$\mathcal{H}_{colregs}^{l} = \kappa \sum_{i=1}^{n_{do}} w^i \mu^{l,i} \tag{13}$$

where κ is a tuning parameter and

$$\mu^{l,i} = \sum_{s=1}^{n_{ps}^i} \mathbb{P}_s^i \mu_s^{l,i} \tag{14}$$

with $\mu_s^{l,i} \in \{0, 1\}$ as the indicator of the own-ship following control behaviour l violating COLREGS with respect to obstacle i in prediction scenario s, calculated as in [2] for head-on, overtaking and crossing situations. The parameters w^i and dynamic obstacle scenario probabilities are again used for weighting purposes. The new formulation now penalizes COLREGS breaches with respect to all dynamic obstacles, and allows for better handling of compliance in multi-ship situations.

The static obstacle related cost or grounding cost is parameterized as

$$\mathcal{H}_{so}^{l} = \max_j \mathcal{H}_{so}^{l,j} \tag{15}$$

where

$$\mathcal{H}_{so}^{l,j} = \max_k \quad (G_1 + G_2 \phi_{j,k}^l V_w^2) \times exp(-(G_3|d_{0j,k}^l - d_{safe}| + G_4 t_k)) \tag{16}$$

inspired by [32], where G_1 to G_4 are tuning parameters, V_w the estimated wind speed, $\phi_{j,k}^l = \max(0, \boldsymbol{\omega}_j \cdot \boldsymbol{L}_{0j,k}^l)$ with $\boldsymbol{\omega}_j$ as the wind direction unit vector. $\boldsymbol{L}_{0j,k}^l$ is the unit vector pointing from the own-ship to the static obstacle j, and $d_{0j,k}^l$ the corresponding distance. d_{safe} is the circular own-ship safety zone.

The trajectory deviation cost is given as

$$\mathcal{H}_{p}^{l} = \frac{1}{n_M} \sum_{M=1}^{n_M} f(\cdot) + \frac{1}{n_M - 1} \sum_{M=2}^{n_M} h(\cdot) \tag{17}$$

with $f(\cdot)$ and $h(\cdot)$ as the control deviation and change cost, respectively. More details on the different terms involved in the cost function can be found in [2], [22], [24], [30], and [33].

D. STANDARD PSB-MPC IMPLEMENTATION

As the PSB-MPC is a finite set MPC, the solution to the non-convex Mixed Integer Nonlinear Program (MINLP) in (1) is parameterized by the chosen discrete set of own-ship control behaviours. The benefit of the finite-set MPC formulation is that by brute force iterating over the set of control behaviours we are able to find a global solution, which would be hard in the case if numerical optimization was used.

Implementing the cost evaluation in the PSB-MPC on a sequential computing platform will involve loops over the own-ship control behaviours, where loops over static and dynamic obstacles in their set of prediction scenarios are found within. This would look something like the method outlined in Algorithm 2. One can see that this implementation involves several nested for loops, especially the one over dynamic obstacles and their prediction scenarios. In addition, one must also loop over the number of discrete samples $t_k \in D(t_0)$ in the predicted trajectories. Thus, the MPC problem will scale poorly with increasing number of control behaviours, static and dynamic obstacles.

Algorithm 2 Standard PSB-MPC Cost Evaluation on a Sequential Processing Platform, Assuming All Obstacle Prediction Scenarios Are Generated Beforehand

Initialize optimal control behaviour to $l^* = 1$.
for $l = 1, .., n_{cbs}$ **do**
  Predict the own-ship trajectory following control behaviour l.
  Calculate the trajectory related cost $\mathcal{H}_p^l$ using (17).
  for $j = 1, .., n_{so}$ **do**
    Calculate the static obstacle j grounding cost $\mathcal{H}_{so}^{l,j}$ using (16).
  end for
  Calculate total grounding cost $\mathcal{H}_{so}^l$ using (15).
  for $i = 1, .., n_{do}$ **do**
    for $s = 1, .., n_{ps}^i$ **do**
      Calculate probabilistic collision cost $C_s^{l,i}$ from (12) and COLREGS indicator $\mu_s^{l,i}$ in (14).
    end for
    Calculate dynamic obstacle i cost $\mathcal{H}_{do}^{l,i}$ using (11).
  end for
  Calculate total dynamic obstacle cost $\mathcal{H}_{do}^l$ using (10).
  Calculate control behaviour cost $\mathcal{H}^l(t_0) = \mathcal{H}_{do}^l + \mathcal{H}_{colregs}^l + \mathcal{H}_{so}^l + \mathcal{H}_p^l$.
  if $\mathcal{H}^l(t_0) < \mathcal{H}^{l^*}(t_0)$ **then**
    Set $l^* = l$.
  end if
end for

III. PARALLELIZED PSB-MPC IMPLEMENTATION

The nature of the finite set MPC described in the above section makes it possible to independently evaluate the cost associated with the control behaviours, and thus apply parallelism in the main part of the algorithm. Furthermore, all obstacle prediction scenarios are assumed to be independent of the own-ship control behaviour and can be generated beforehand. This is deemed reasonable as we take into account maneuvering uncertainty in the obstacle prediction.

When considering large amounts of situational information and a dense set of possible own-ship trajectories, evaluating the cost of an own-ship control behaviour sequentially will not make the COLAV planning algorithm real-time feasible. Parallelizing the cost evaluation will allow for more refined own-ship decision making, as more own-ship trajectories can be considered. Also, more static obstacles and prediction scenarios for dynamic obstacles can then be considered, resulting in increased situational awareness for the own-ship. The limiting factor here will then be how many threads that can be scheduled on the parallel computation platform.

A naive way of cost function evaluation parallelization would be to schedule GPU threads to evaluate the cost (9). However, this is a big task for a single thread, as it among others involves going through all static and dynamic obstacles in all their prediction scenarios to find the total cost. This equates to a nested for loop over obstacles, prediction scenarios and discrete time samples in the code that implements the MPC as in Algorithm 2, and will scale poorly with an increase in the number of obstacles and number of prediction scenarios n_{ps}^i for dynamic obstacles. As GPU cores have limited processing power compared to CPU cores, their tasks should be as lightweight as possible.

Two of the main bottlenecks in the cost evaluation is calculating the distance to static obstacles and the estimation of collision probabilities. The first bottleneck is readily apparent when considering large polygons with tens of thousands of vertices. However, the RDP algorithm will reduce the number of vertices in a polygon and thus alleviate computational effort. Reducing the number of time steps to evaluate the grounding cost can also aid in fixing this problem.

For the second bottleneck, giving each thread the job of estimating collision probabilities associated with only a pair of trajectories will give higher throughput, at the cost of scheduling more threads on the GPU and therefore having higher memory demands. However, as GPU technology continue to improve with respect to single core processing power and device memory, this is deemed a worthy trade-off. Furthermore, the calculation efficiency using the Cross-Entropy method for collision probability estimation [3] is increased by estimating $\hat{\mathbb{P}}_c^{l,i,s} \approx 0$ when the predicted distance between the own-ship and an obstacle is larger than $d_{safe} + 4\sigma_{largest}^i$, where $\sigma_{largest}^i$ is the standard deviation along the axis where obstacle i has the largest predicted positional uncertainty.

Thus, a way to solve the bottlenecks in (1) utilizing parallel processing can be done in two steps: First schedule n_{cbs} threads to predict the own-ship trajectory and calculate the trajectory related cost (17) for each control behaviour $l = 1, 2, .., n_{cbs}$. Then, schedule

$$n_{ct} = n_{cbs} \cdot (n_{so} + \sum_{i=1}^{n_{obst}} n_{ps}^i) \tag{18}$$

threads that evaluates the cost (16), (12) and the COLREGS violation indicator in (14). The total cost (9) is finally stitched together afterwards on the CPU. This way, no GPU thread has run-times dependent on large nested for-loops, and the MPC-problem scales better with increasing number of obstacles and dynamic obstacle prediction scenarios. This approach of using parallelization to solving (1) can be summarized in Algorithm 3. Here, ''parfor'' denotes a parallel for loop.

Algorithm 3 Parallelized PSB-MPC Cost Evaluation, Assuming All Obstacle Prediction Scenarios Are Generated Beforehand

1: Schedule n_{cbs} GPU threads, transferring all the required data for own-ship trajectory prediction and calculating (17).
2: **parfor** $l = 1, .., n_{cbs}$ **do**
3: Predict the own-ship trajectory following control behaviour l, save trajectory in GPU memory for use by the subsequent processing.
4: Calculate the trajectory related cost $\mathcal{H}_p^l$ using (17).
5: Return the results to CPU memory.
6: **end parfor**
7: Schedule n_{ct} GPU threads, transferring all the required data needed for partial static and dynamic obstacle cost evaluation.
8: **parfor** $ct = 1, .., n_{ct}$ **do**
9: Extract control behaviour l, static obstacle j or dynamic obstacle i and prediction scenario s to consider.
10: Calculate the grounding cost $\mathcal{H}_{so}^{l,j}$ using (16) or $C_s^{l,i}$ using (12) and the indicator $\mu_s^{l,i}$, depending on if a static or dynamic obstacle is considered in the thread.
11: Return the results to CPU memory.
12: **end parfor**
13: Use all the calculated $\mathcal{H}_{so}^{l,j}$ to calculate $\mathcal{H}_{so}^l$ using (15).
14: Use all the calculated $C_s^{l,i}$ and $\mu_s^{l,i}$ plus other relevant data to calculate $\mathcal{H}_{do}^l$ using (10) and $\mathcal{H}_{colregs}^l$ using (13).
15: Finally, calculate (9) for all control behaviours using the previously calculated terms $\mathcal{H}_{do}^l$, $\mathcal{H}_{colregs}^l$, $\mathcal{H}_{so}^l$ and $\mathcal{H}_p^l$ of the cost function, and extract the optimal one l^* giving minimal cost.

Note that how the PSB-MPC algorithm is implemented both on the CPU and GPU will have big impacts on the run-time results obtained in this article. Hardware, programming language and software libraries used will be significant factors here. An alternative to the structure in algorithm 3 would be to have separate kernels to evaluate the static and dynamic obstacle partial costs. This could be better suiting for a setup with multiple GPUs, as the two kernels could then be run concurrently. Lastly, because of the extra latency overhead due to porting data from the host (CPU) to the device (GPU), as much memory as possible for the relevant data needed on the GPU should be pre-allocated.

IV. SIMULATION STUDY

A. OWN-SHIP MODEL

In this article we also use a kinematic model with LOS guidance [25] and a constant speed profile for the own-ship, as used for dynamic obstacles in Section II-A, to predict any of the candidate trajectories shown in Fig. 1. Specific to the own-ship, the model is restated as

$$\begin{aligned} x_{k+1} &= x_k + U_k cos(\chi_k) \\ y_{k+1} &= y_k + U_k sin(\chi_k) \\ \chi_{k+1} &= \chi_k + \frac{1}{T_\chi}(\chi_{d,k} - \chi_k) \\ U_{k+1} &= U_k + \frac{1}{T_U}(U_{d,k} - U_k) \end{aligned} \tag{19}$$

which describes the own-ship state $\boldsymbol{x}_k = [x_k,\ y_k,\ \chi_k,\ U_k]^T$ motion at time t_k. Again, the state consists of the vessel surface position in Cartesian coordinates, course over ground (COG) and speed over ground (SOG), respectively. The time constants T_U and T_χ in speed and course may be found by applying parameter identification methods using motion data from the considered vessel.

For each own-ship control behaviour, the speed modifications $u_{m,M}^l$ and course modifications $\chi_{m,M}^l$ for all n_M sequential maneuvers considered in the PSB-MPC are applied to the LOS guidance references for speed and course at maneuvering times $t_M, M = 1, 2, \ldots, n_M$, evenly spaced througout the horizon with a time spacing parameter t_{ts} for simplicity.

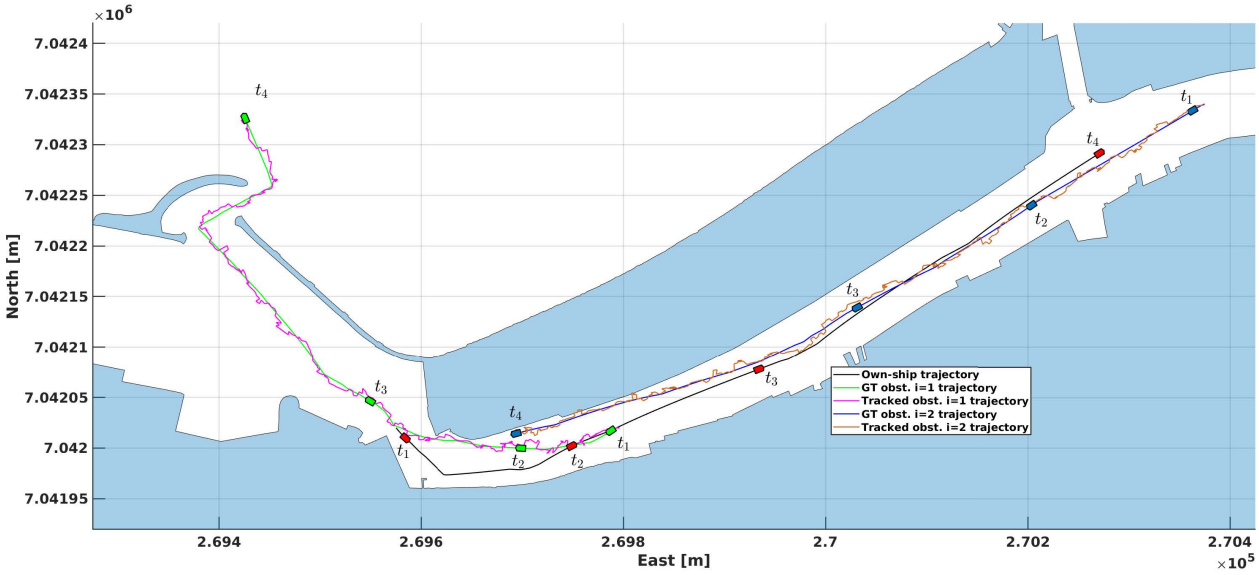
B. SETUP

The GPU-based PSB-MPC is tested in two situations to illustrate that the COLAV planning algorithm can tackle dynamic obstacles with uncertainties in addition to grounding hazards. The first river scenario is chosen to test how the COLAV planning method handles avoidance in confined spaces, whereas the second scenario aims to test the algorithm performance in a longer time horizon with multiple dynamic obstacles in a mix of an open sea area and a narrow channel. The setup with tracking system and parameters are similar to that in [3], where the obstacle tracker is deliberately tuned conservatively to test the MPC robustness against kinematic uncertainty. The situations are described below, with a number of $N_{MC} = 50$ Monte Carlo simulations used for each situation. A run-time analysis considering the first situation is performed, comparing the CPU and GPU implementations of the PSB-MPC, Algorithm 2 and 3, respectively. The CPU version evaluates the PSB-MPC cost for all own-ship control behaviours sequentially on CPU cores. The simulations are performed on a work station with an Intel(R) Core(TM) i9-10900K 3.70 GHz processor, with 32 GB RAM and an NVIDIA GeForce RTX 3090 GPU. C++ is used to implement the CPU version of the PSB-MPC, whereas C++ and CUDA is used for the GPU version.

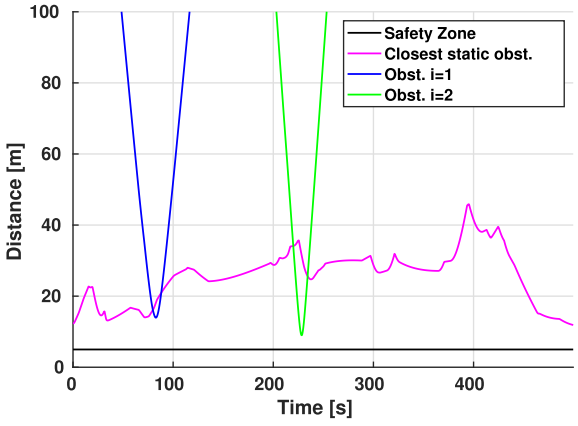
1) Head-on scenario in Nidelva in Trondheim, Norway. The own-ship travels upstream with constant

TABLE 1. Important PSB-MPC parameters for the Nidelva situation.

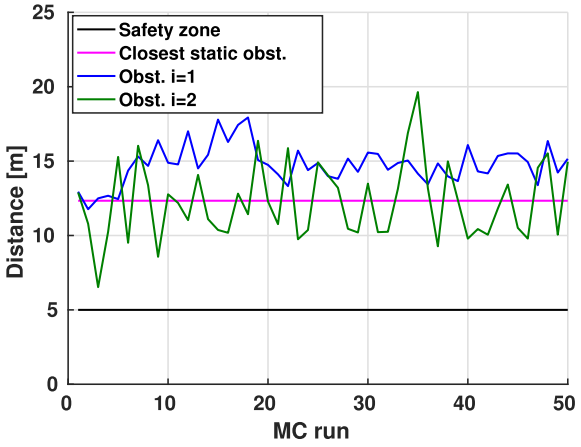
Parameter	Value	Comment
ϵ	2 m	RDP distance threshold
T_{MPC}	150 s	Prediction horizon
T_s	0.5 s	Prediction time step
T_d	100 s	Collision cost time discounting parameter
n_{ps}^{LOS}	5	Number of LOS prediction scenarios
r_{ct}	10.0 m	Prediction scenario spacing
d_{so}	200.0 m	Static obstacle consideration range
n_M	2	Number of sequential avoidance maneuvers
$u_{offsets,1}$	$\{1.0, 0.5, 0.0\}$	Surge offsets first maneuver
$u_{offsets,2}$	$\{1.0, 0.5\}$	Surge offsets second maneuver
$\chi_{offsets,1}$	$\{-60, -45, -30, -15, -10, -5, 0, 5, 10, 15, 30, 45, 60\}$	Course offsets first maneuver
$\chi_{offsets,2}$	$\{-60, -45, -30, -15, -10, -5, 0, 5, 10, 15, 30, 45, 60\}$	Course offsets second maneuver



(a) North east plot at multiple time instants for a sample run. Dynamic obstacles are shown in green ($i = 1$) and blue ($i = 2$). The own-ship is shown in red.



(b) Distance to static and dynamic obstacles for the sample run.



(c) Statistics on the minimum distance to the obstacles over all N_{MC} simulation runs.

FIGURE 4. Results for the situation in Nidelva with multiple obstacles.

speed 2 m/s, whereas two dynamic obstacles travels downstream with constant speed 2 m/s. Vessels of lengths 5 m are here considered, and an own-ship safety zone of $d_{safe} = 5$ m is used.

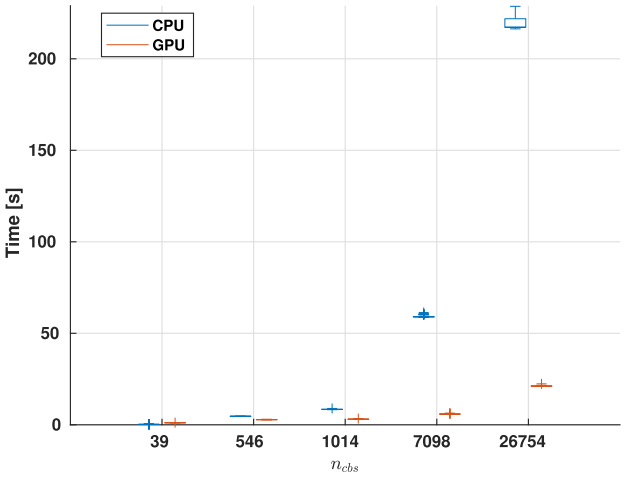


FIGURE 5. Box-plot representation of the runtime results with respect to increasing numbers of control behaviours n_{cbs}, when keeping the number of dynamic obstacle prediction scenarios constant at $n_{ps}^i = 1$.

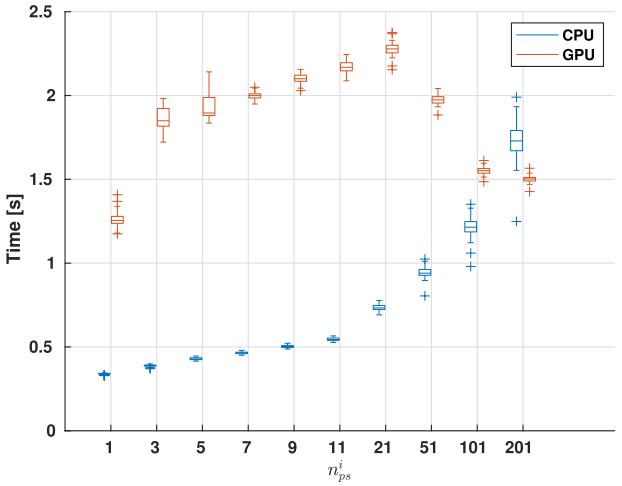


FIGURE 6. Box-plot representation of the runtime results with respect to increasing dynamic obstacle prediction scenarios n_{ps}^i, when keeping the number of own-ship avoidance maneuvers constant at $n_M = 1$ and a total number of control behaviours $n_{cbs} = 39$.

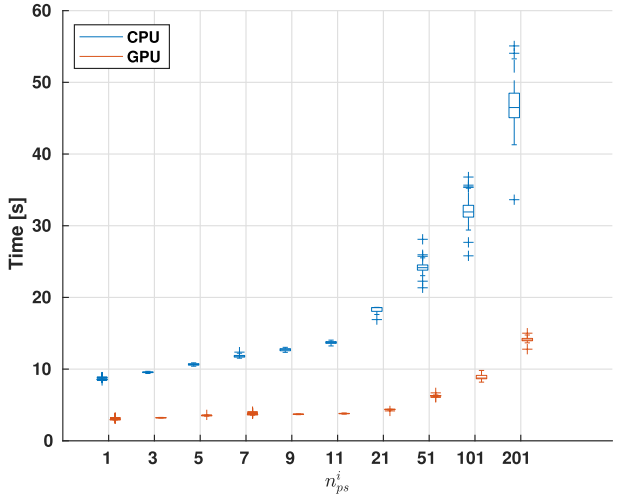


FIGURE 7. Box-plot representation of the runtime results with respect to increasing dynamic obstacle prediction scenarios n_{ps}^i, when keeping the number of own-ship avoidance maneuvers constant at $n_M = 2$ and a total number of control behaviours $n_{cbs} = 1014$.

2) Multi-ship situation with grounding hazards near Sakshaug, Trøndelag in Norway. Dynamic obstacle $i = 1$ is traveling from the south through Straumen with constant speed 5 m/s and ends up in an overtaking situation with respect to the own-ship, whereas dynamic obstacle $i = 2$ travels east-west through Straumen with constant speed 6 m/s and ends up in head-on situations with respect to the other vessels. Obstacle $i = 3$ travels with speed 7 m/s east-west from Straumen towards the own-ship in a head-on situation, and obstacle $i = 4$ just north-east of the own-ship travels south with speed 8 m/s. The own-ship travels with constant speed 7 m/s. Vessels of lengths 10 m are considered, and an own-ship safety zone of $d_{safe} = 10$ m is used. In addition to COLREGS adherence with respect to multiple ships, the challenge here is voyage through the narrow passage in Straumen, beneath the bridge which has two pylons that the vessels have to avoid.

For simplicity, a uniform set of scenario probabilities $\mathbb{P}_s^i$ are defined for the dynamic obstacles, which resembles a conservative case when no prior information from intent inference is available. For the grounding hazards, only polygons within a range d_{so} are considered, to reduce computation time. Waypoints for the own-ship are set in a way that a top level planner could generate, but with small margins to static obstacles, such that the anti-grounding part of the PSB-MPC becomes important. Furthermore, the waypoints are set such that a nominal collision-free trajectory does not exist for all vessels involved.

The MPC is tuned such that anti-grounding and collision avoidance is prioritized over adhering to COLREGS and following the nominal trajectory. Naturally, because river voyage is different from sea voyage, the PSB-MPC has a different tuning for the two situations. Important parameters for the first situation tuning are given in Table 1.

C. NIDELVA SITUATION

Results for the first situation are given in Fig. 4. The dynamic obstacles are here assumed to be self-governing, running their own PSB-MPC algorithm to simulate human behaviour. The conservative tracking system tuning will create an extra challenge for the COLAV planning algorithm, with higher kinematic obstacle uncertainty. Despite this and nearby grounding hazards, all vessels involved are able to avoid collision and grounding in addition to adhering to the COLREGS rules 8, 13 and 16 related to clear actions, head-on situation and actions for give-way vessels, respectively. The near constant minimum distance to the closest static obstacle in the statistics is because the own-ship is closest to a grounding hazard initially. Note that the map data for the river area do not include the piers at which boats are docked, which would be taken into account through e.g. LIDAR data in a real-time application.

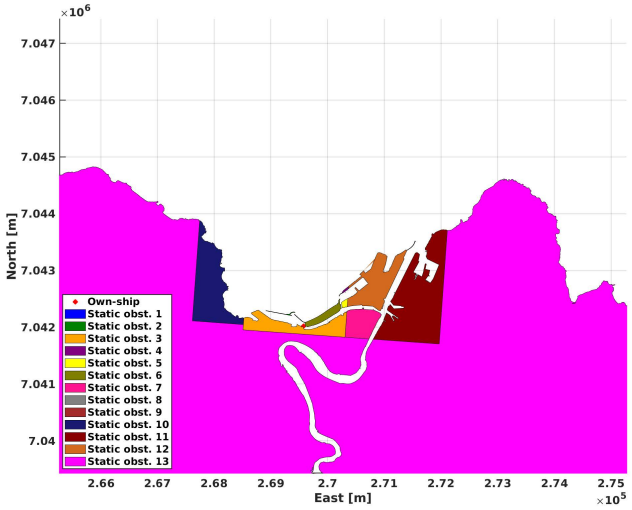


FIGURE 8. Map of the Trondheim region with Nidelva in the middle, with all relevant polygons labelled with different colors. The own-ship position is the small red dot in Nidelva in the middle.

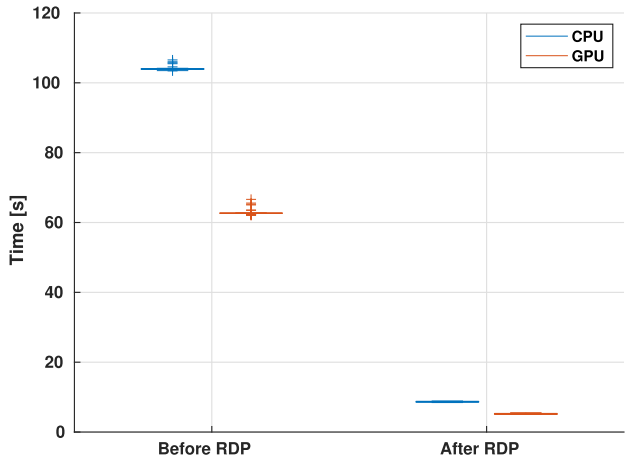


FIGURE 9. Box-plot representation of the runtime results with respect to the worst case polygon scenario before and after applying RDP, when keeping the number of own-ship avoidance maneuvers constant at $n_M = 2$ and a total number of control behaviours $n_{cbs} = 1014$.

For the situation in Nidelva, a run-time analysis was performed with respect to the number of control behaviours n_{cbs} for the MPC, and the number of dynamic obstacle prediction scenarios n_{ps}^i considered. The number of control behaviours is increased by increasing the number of sequential maneuvers n_M in the horizon, and by expanding the finite set of course and surge modifications. Both the CPU and GPU implementations were run for N_{MC} simulations for each parameter setting. Figs. 5, 6 and 7 show a box-plot representation of the results. The GPU-implementation of the PSB-MPC performs better than the CPU-version when the number of control behaviours increase beyond a thousand. With $n_{cbs} < 1000$ and a scheduled number of threads $n_{ct} < 5000$, the overhead of launching the GPU kernels becomes too large compared to the gain of parallelized cost evaluation. This makes the CPU-implementation feasible for cases where

TABLE 2. Polygon vertices before and after applying RDP on the Nidelva environment.

Polygon	Vertices before	Vertices after
1	8	3
2	207	27
3	649	95
4	322	33
5	140	18
6	890	53
7	207	48
8	8	3
9	8	5
10	1633	187
11	2110	162
12	2483	143
13	21961	1734

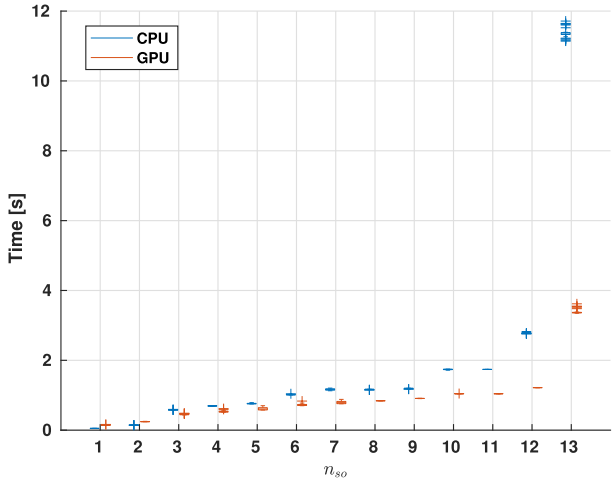


FIGURE 10. Box-plot representation of the runtime results with respect to increasing numbers of static obstacles n_{so}, when keeping the number of own-ship avoidance maneuvers constant at $n_M = 2$ and a total number of control behaviours $n_{cbs} = 1014$.

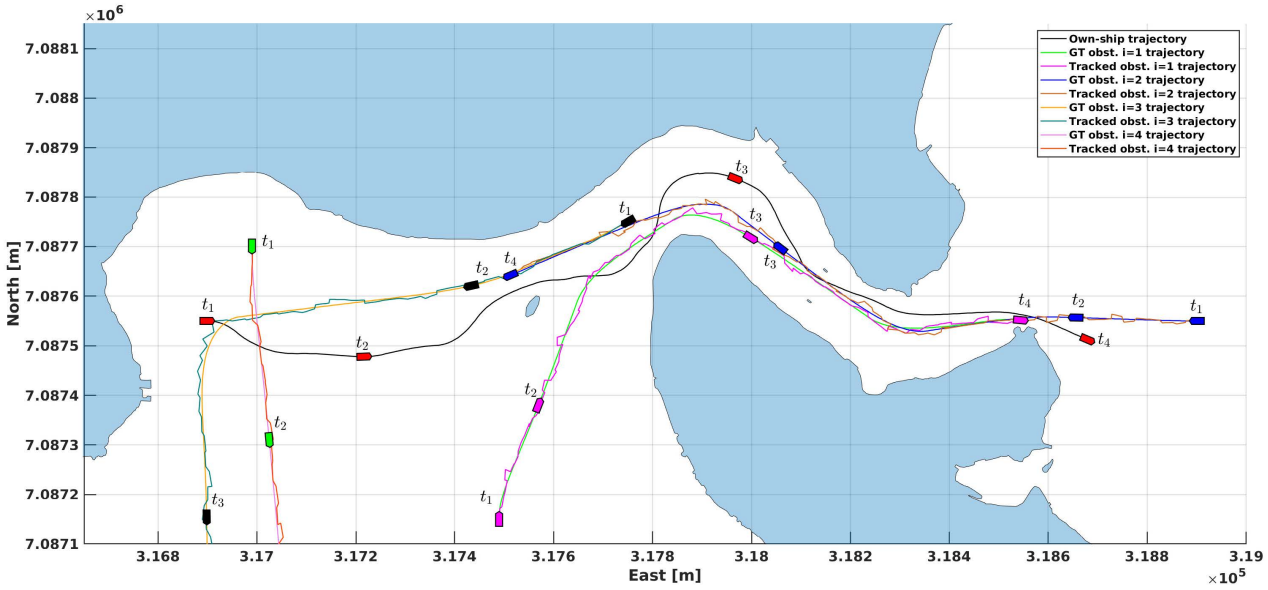
typically $n_M = 1$ and a small number of possible course and speed changes is enough, and only a small number of static and dynamic obstacles are considered.

Furthermore, one can see that the CPU implementation performs better than the GPU implementation when considering increasing numbers of prediction scenarios up until $n_{ps}^i = 101$ for dynamic obstacles, when using a low number of control behaviours $n_{cbs} = 39$. In this case, the GPU run-time is mainly caused by the overhead of porting data back and forth between the host and device side. The contrary result is the case when considering $n_{cbs} > 1000$. This is again because a CPU is optimized for fast sequential execution on fewer but more complex tasks, whereas a GPU is optimized for execution of many simple tasks in parallel. A similar result is obtained by increasing n_{obst} while keeping n_{ps}^i constant, but will not be reported here.

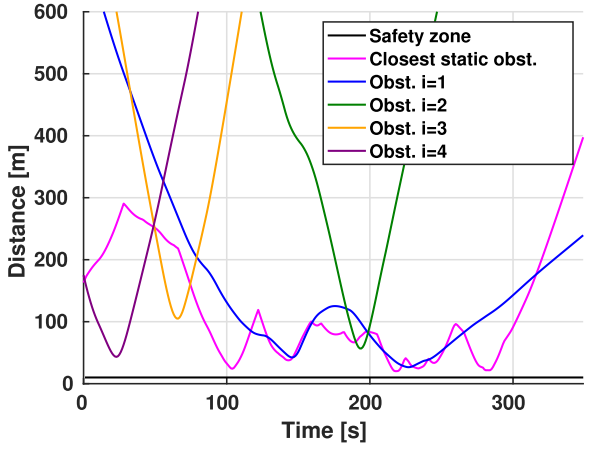
From Figs. 5 - 7, an approximate linear scaling of the MPC run-time complexity with increasing own-ship control behaviours, dynamic obstacle scenarios and static obstacles can be found. For static obstacles represented as polygons, one also have to take into account the added run-time complexity due to the number of vertices in the polygons.

TABLE 3. Important PSB-MPC parameters for the Sakshaug situation.

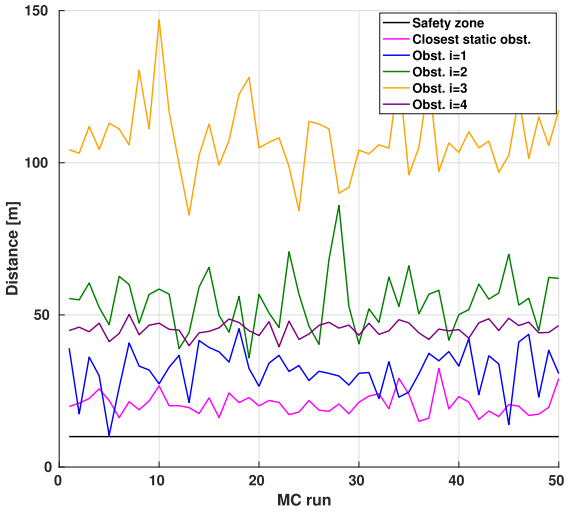
Parameter	Value	Comment
ϵ	2 m	RDP distance threshold
T_{MPC}	150 s	Prediction horizon
T_s	1.0 s	Prediction time step
T_d	100 s	Collision cost time discounting parameter
n_{ps}^{LOS}	5	Number of LOS prediction scenarios
r_{ct}	20.0 m	Prediction scenario spacing
d_{so}	800.0 m	Static obstacle consideration range
n_M	2	Number of sequential avoidance maneuvers
$u_{offsets,1}$	$\{1.0, 0.5, 0.0\}$	Surge offsets first maneuver
$u_{offsets,2}$	$\{1.0, 0.5\}$	Surge offsets second maneuver
$\chi_{offsets,1}$	$\{-90, -75, -60, -45, -30, -15, 0, 15, 30, 45, 60, 75, 90\}$	Course offsets first maneuver
$\chi_{offsets,2}$	$\{-90, -75, -60, -45, -30, -15, 0, 15, 30, 45, 60, 75, 90\}$	Course offsets second maneuver



(a) North east plot at multiple time instants for a sample run. Dynamic obstacles in purple ($i = 1$), blue ($i = 2$), black ($i = 3$) and green ($i = 4$). The own-ship is shown in red.

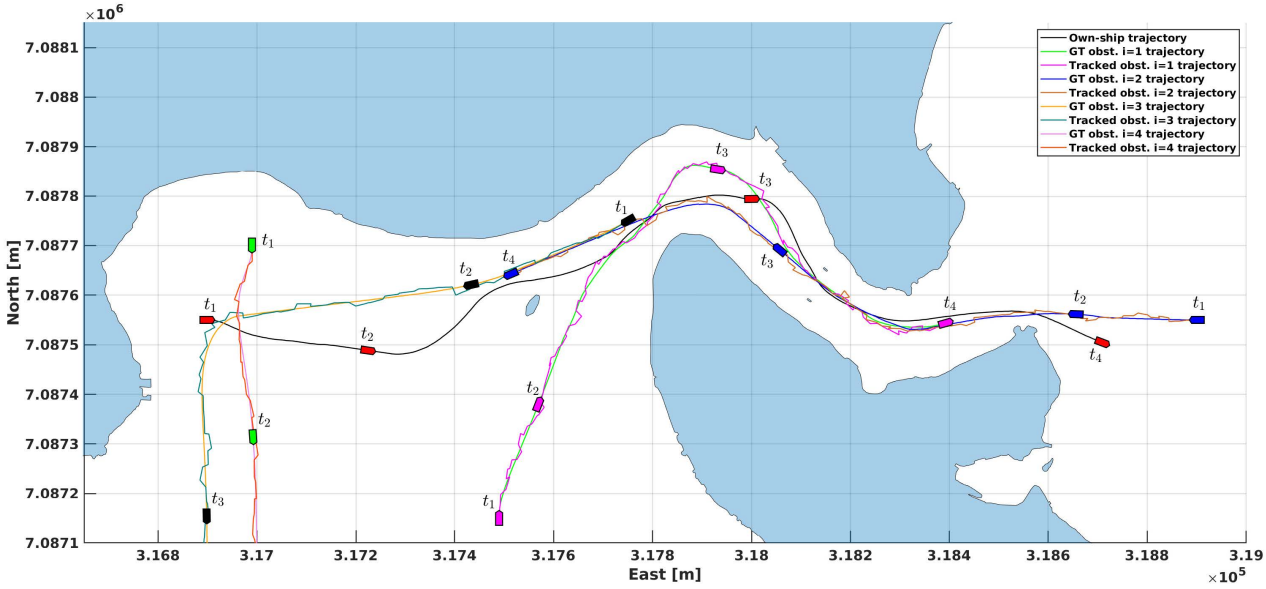


(b) Distance to static and dynamic obstacles for the sample run.

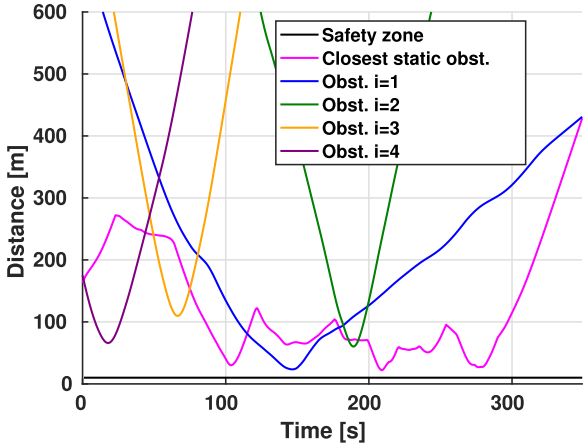


(c) Statistics on the minimum distance to the obstacles over all N_{MC} simulation runs.

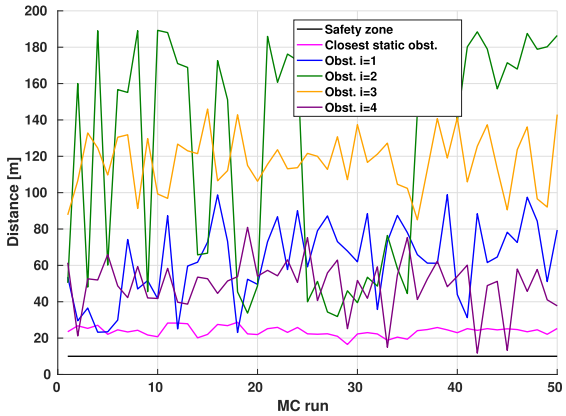
FIGURE 11. Results for the situation in Sakshaug with multiple obstacles in the first case.



(a) North east plot at multiple time instants for the sample run. Dynamic obstacles in purple ($i = 1$), blue ($i = 2$), black ($i = 3$) and green ($i = 4$). The own-ship is shown in red.



(b) Distance to static and dynamic obstacles for the sample run.



(c) Statistics on the minimum distance to the obstacles over all N_{MC} simulation runs.

FIGURE 12. **Results for the situation in Sakshaug with multiple obstacles in the second case.**

Also, tests to compare the run-time related to calculating predominantly the grounding cost in the MPC on a CPU and GPU platform was performed, when the own-ship is located in Nidelva standing still. No dynamic obstacles are considered, and thus the calculation of the distance to static obstacles will be the bottleneck. The largest static obstacle in the region is a polygon with 21962 vertices originally, and has 1734 vertices after application of the RDP algorithm. The map environment around Nidelva in Trondheim is illustrated in Fig. 8, where the static obstacle $j = 13$ is the largest one with 21962 vertices. Information about the number of vertices for each polygon is given in Table 2.

The first test compares the run-time when only considering the largest polygon, with and without usage of the RDP algorithm. This is a worst case scenario, as a real-time anti-grounding system should preprocess large polygons such that only the relevant local part is considered. We however include this test for completeness, as it shows the importance of polygon preprocessing. Results are here given in Fig. 9.

The results in Fig. 10 show a run-time analysis for increasing numbers of static obstacles, after using the RDP for polygon simplification. Note that the results considering an increasing number of static obstacles are strongly dependent on the number of vertices for each obstacle, which varies from 3 to 1734 vertices as seen from Table 2 after using RDP on this environment. This is why there is a sharp increase in average run-time when $n_{so} = 13$, because the largest polygon is then included in the consideration. An approximate linear run-time

increase can however be found when considering polygons of fairly the same complexity. The trend from these results is that the GPU implementation becomes more feasible than the CPU one when the number of scheduled parallel threads n_{ct} surpasses around 5000.

D. SAKSHAUG SITUATION

Important parameters for the tuning are given in Table 3, with results shown in Fig. 11 and 12. The first case show results when only the own-ship has a COLAV planning algorithm, whereas the second case show results when all vessels involved use the PSB-MPC. For the first case, waypoints for the obstacles are set such that they will not collide with each other, but would collide with the own-ship if no COLAV planning algorithm was used.

For both the first and second case, the own-ship has difficulties with overtaking purple obstacle $i = 1$ while simultaneously avoid grounding and avoiding blue obstacle $i = 2$ head-on, that adheres to both COLREGS rules 13 and 14 regarding overtaking and head-on. Especially in the time period between t_2 and t_3, the own-ship struggles with figuring out the side to overtake obstacle $i = 1$ on when entering Straumen, hence the oscillations in the trajectory in this period. The black obstacle $i = 3$ and green obstacle $i = 4$ are easier to avoid as the vessels are here less constrained by land.

Thus, the own-ship is in general able to avoid collision with all obstacles in both cases, but COLREGS adherence in the narrow passage is difficult to accomplish with respect to all ships. This is mainly due to constant conservative intent information being used, with uniform prediction scenario probabilities for dynamic obstacle trajectories, essentially assuming that no dynamic obstacle will have specific inclinations towards adhering to the COLREGS. Also needing to avoid grounding in the narrow passage further restricts the PSB-MPC's ability to adhere to COLREGS in a safe manner. The algorithm is however able to keep safe distance to all obstacles in all Monte Carlo simulation runs. The diversity of the environment makes algorithm tuning challenging, as one can argue that the COLAV planning algorithm parameters should be adaptive based on changes in the situation.

When the dynamic obstacles do not explicitly follow COLREGS in the first case, the own-ship can be more excused for not doing the same with respect to all vessels. For the second case, one see the potential for vessel-vessel communication to explicitly reduce trajectory uncertainties and adhere to COLREGS, during the passage through Straumen. Addressing these issues is the topic of future research more focused on multi-ship COLREGS compliance in confined waters.

Regarding run-time complexity for this example, it will be similar as for the first situation when considering increasing dynamic obstacles and their prediction scenarios. There will be a small increase in the run-time due to the Sakshaug situation has larger and more complex static obstacles, although a smaller set than for the Nidelva situation is considered in the proximity of the own-ship. In total, run-time results generated for this example would be fairly similar to the first simulation, albeit with a bias on the static obstacle run-time complexity due to larger obstacles considered.

V. CONCLUSION

The PSB-MPC COLAV planning algorithm presented in this article facilitates both dynamic and static obstacle avoidance, with the most performance-critical part of its algorithm implemented on the GPU. What separates it from current state-of-the-art is the computational speed of the algorithm, where the cost evaluation is parallelized such that the MPC problem scales approximately linearly with increasing control behaviours, static and dynamic obstacles and prediction scenarios, as shown in the run-time results presented. This makes the COLAV planner able to consider more control behaviours and dynamic obstacle prediction scenarios efficiently, which results in real-time capabilities and performance gains in cases where large amounts of situational information and possible own-ship decisions have to be considered.

In simulation, the COLAV planning algorithm is shown to handle both grounding hazards and multiple dynamic obstacles in a safe manner, both in a narrow river environment, and also in a mix of more open sea and narrow waters. However, there is an inherent challenge in finding parameters that will make the algorithm work robustly and adhere to COLREGS for multiple types of situations, especially when the environmental constraints vary a lot.

Future work will involve making the PSB-MPC adaptive to the environment faced, and utilize historical data for tuning the algorithm. Also, the dynamic obstacle prediction and COLREGS penalization cost evaluation should be extended to consider static obstacles, for better applicability in confined spaces.

REFERENCES

[1] *COLREGS—International Regulations for Preventing Collisions at Sea*, Int. Maritime Org., London, U.K., 1972.

[2] 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 Trans. Intell. Transp. Syst.*, vol. 17, no. 12, pp. 3407–3422, Dec. 2016.

[3] T. Tengesdal, T. A. Johansen, and E. F. Brekke, "Ship collision avoidance utilizing the cross-entropy method for collision risk assessment," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11148–11161, Aug. 2022.

[4] C. Tam, R. Bucknall, and A. Greig, "Review of collision avoidance and path planning methods for ships in close range encounters," *J. Navigat.*, vol. 62, no. 3, pp. 455–476, 2009.

[5] Y. Huang, L. Chen, P. Chen, R. R. Negenborn, and P. H. A. J. M. V. Gelder, "Ship collision avoidance methods: State-of-the-art," *Saf. Sci.*, vol. 121, pp. 451–473, Jan. 2020.

[6] A. Vagale, R. Oucheikh, R. T. Bye, O. L. Osen, and T. I. Fossen, "Path planning and collision avoidance for autonomous surface vehicles I: A review," *J. Mar. Sci. Technol.*, vol. 26, no. 4, pp. 1292–1306, Dec. 2021.

[7] A. Vagale, R. T. Bye, R. Oucheikh, O. L. Osen, and T. I. Fossen, "Path planning and collision avoidance for autonomous surface vehicles II: A comparative study of algorithms," *J. Mar. Sci. Technol.*, vol. 26, no. 4, pp. 1307–1323, Dec. 2021.

[8] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[9] C. Chauvin and S. Lardjane, "Decision making and strategies in an interaction situation: Collision avoidance at sea," *Transp. Res. F, Traffic Psychol. Behaviour*, vol. 11, no. 4, pp. 259–269, Jul. 2008.
[10] S. R. Clawson Jr. (2013). *Overtaking or Crossing? Don't Assume What Other Ship Will do*. [Online]. Available: http://www.professionalmariner.com/August-2013/Overtaking-or-crossing-Dont-assume-what-other-ship-will-do/
[11] K. Woerner, M. R. Benjamin, M. Novitzky, and J. J. Leonard, "Quantifying protocol evaluation for autonomous collision avoidance," *Auto. Robots*, vol. 43, no. 4, pp. 967–991, Apr. 2019.
[12] B. C. Shah, P. Švec, I. R. Bertaska, A. J. Sinisterra, W. Klinger, K. V. Ellenrieder, M. Dhanak, and S. K. Gupta, "Resolution-adaptive risk-aware trajectory planning for surface vehicles operating in congested civilian traffic," *Auton. Robots*, vol. 40, no. 7, pp. 1139–1163, 2016.
[13] B.-O. H. Eriksen, G. Bitar, M. Breivik, and A. M. Lekkas, "Hybrid collision avoidance for ASVs compliant with COLREGs rules 8 and 13–17," 2019, *arXiv:1907.00198*.
[14] P. Agrawal and J. M. Dolan, "COLREGS-compliant target following for an unmanned surface vehicle in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 1065–1070.
[15] M. Candeloro, A. M. Lekkas, and A. J. Sørensen, "A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels," *Control Eng. Pract.*, vol. 61, pp. 41–54, Apr. 2017.
[16] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking. Part I. Dynamic models," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 39, no. 4, pp. 1333–1364, Oct. 2003.
[17] L. M. Millefiori, P. Braca, K. Bryan, and P. Willett, "Long-term vessel kinematics prediction exploiting mean-reverting processes," in *Proc. 19th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2016, pp. 232–239.
[18] M. Abdelaal, M. Fränzle, and A. Hahn, "Nonlinear model predictive control for trajectory tracking and collision avoidance of underactuated vessels with disturbances," *Ocean Eng.*, vol. 160, pp. 168–180, Jul. 2018.
[19] H.-T.-L. Chiang and L. Tapia, "COLREG-RRT: An RRT-based COLREGS-compliant motion planner for surface vehicle navigation," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2024–2031, Jul. 2018.
[20] Y. Lyu, J. Hu, B. M. Chen, C. Zhao, and Q. Pan, "Multivehicle flocking with collision avoidance via distributed model predictive control," *IEEE Trans. Cybern.*, vol. 51, no. 5, pp. 2651–2662, May 2021.
[21] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica, Int. J. Geographic Inf. Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.
[22] T. Trym, E. F. Brekke, and T. A. Johansen, "On collision risk assessment for autonomous ships using scenario-based MPC," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14509–14516, 2020.
[23] L. M. Millefiori, P. Braca, K. Bryan, and P. Willett, "Modeling vessel kinematics using a stochastic mean-reverting process for long-term prediction," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, no. 5, pp. 2313–2330, Oct. 2016.
[24] T. Tengesdal, T. A. Johansen, and E. Brekke, "Risk-based autonomous maritime collision avoidance considering obstacle intentions," in *Proc. IEEE 23rd Int. Conf. Inf. Fusion (FUSION)*, Jul. 2020, pp. 1–8.
[25] M. Breivik and T. I. Fossen, "Guidance laws for planar motion control," in *Proc. 47th IEEE Conf. Decis. Control*, Sep. 2008, pp. 570–577.
[26] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Hoboken, NJ, USA: Wiley, 2011.
[27] B. K. Patle, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot," *Defence Technol.*, vol. 15, pp. 582–606, Aug. 2019.
[28] S. Blindheim and T. A. Johansen, "Electronic navigational charts for visualization, simulation, and autonomous ship control," *IEEE Access*, vol. 10, pp. 3716–3737, 2022.
[29] C.-W. Huang and T.-Y. Shih, "On the complexity of point-in-polygon algorithms," *Comput. Geosci.*, vol. 23, no. 1, pp. 109–118, Feb. 1997.
[30] D. K. M. Kufoalor, E. Wilthil, I. B. Hagen, E. F. Brekke, and T. A. Johansen, "Autonomous COLREGs-compliant decision making using maritime radar tracking and model predictive control," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 2536–2542.
[31] S. V. Rothmund, T. Tengesdal, E. F. Brekke, and T. A. Johansen, "Intention modelling and inference for autonomous collision avoidance at sea," *TechRxiv*, Oct. 2021, doi: 10.36227/techrxiv.16825870.v2.
[32] S. Blindheim, S. Gros, and T. A. Johansen, "Risk-based model predictive control for autonomous ship emergency management," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14524–14531, 2020.
[33] D. K. M. Kufoalor, T. A. Johansen, E. F. Brekke, A. Hepsø, and K. Trnka, "Autonomous maritime collision avoidance: Field verification of autonomous surface vehicle behavior in challenging scenarios," *J. Field Robot.*, vol. 37, no. 3, pp. 387–403, Apr. 2020.

TRYM TENGESDAL received the M.Sc. and Ph.D. degrees in engineering cybernetics from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 2019 and 2022, respectively. He is currently a Postdoctoral Researcher at NTNU, affiliated with the Centre for Autonomous Marine Operations and Systems (AMOS) and the Autoship Centre for Research-Based Innovation. His research interest includes risk-based COLREGS-compliant collision avoidance for autonomous ships.

TOR ARNE JOHANSEN (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 1989 and 1994, respectively. He was a Researcher with SINTEF Electronics and Cybernetics, before he was appointed as an Associate Professor with the Norwegian University of Science and Technology, in 1997, and later promoted as a Professor, in 2001. In 2002, he Co-Founded Marine Cybernetics AS, where he has been the Vice President, since 2008. He is currently a Principal Researcher with the Center of Excellence on Autonomous Marine Operations and Systems and the Director of the Unmanned Aerial Vehicle Laboratory, NTNU. He has published more than 100 articles in international journals and numerous conference papers and book chapters in the areas of control, estimation, and optimization with applications in the marine, automotive, biomedical, and process industries.

TOM DANIEL GRANDE received the M.Sc. degree in engineering cybernetics with a specialization in autonomous systems from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 2021. He is currently working with machine learning applications for Minerva, Analysis Department, Norwegian Tax Authorities.

SIMON BLINDHEIM received the M.Sc. degree in engineering cybernetics. He is working with autonomous risk-based decision-making as a Ph.D. Candidate/a Researcher at the Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Norway.