CommonOcean-Sim: A Traffic Simulation Environment for Unmanned Surface Vessels

Hanna Krasowski[1] and Stefan Schärdinger[2]

Abstract— **Autonomous vessels have the potential to enhance maritime safety and mitigate environmental, economic, and injury risks. However, research in this domain remains limited, partly due to a lack of benchmarks and open-source tools tailored to maritime applications. The CommonOcean platform addresses this gap by providing software and traffic scenarios for motion planning research on unmanned surface vessels. In this paper, we introduce a major extension: CommonOcean-Sim, a modular simulation environment for heterogeneous multi-agent maritime traffic. CommonOcean-Sim enables configurable simulation using real-world or handcrafted safety-critical maritime scenarios, and various vessel types. Users can select controllers that ensure reactive and traffic rule compliant navigation according to the Convention on the International Regulations for Preventing Collisions at Sea (COLREGS). CommonOcean-Sim features a modular architecture that allows for seamless integration of custom control algorithms as well as extensive configurability and scalability to a multitude of traffic situations, including heterogeneous high-density traffic. We demonstrate these capabilities on a variety of traffic scenarios, highlighting the potential of CommonOcean-Sim to facilitate research on autonomous vessels. CommonOcean-Sim is available at commonocean.cps.cit.tum.de/commonocean-sim.**

I. Introduction

The most common cause of collisions at sea is human error. In 2023, the European Maritime Safety Agency documented 418 collision events, of which about 25% were categorized as severe incidents, and over 60% caused by human error [1]. Additionally, the steadily growing maritime traffic increases the likelihood of critical encounters [2]. Consequently, autonomous vessels could significantly improve maritime safety, attracting research interest in autonomous docking [3]–[5], autonomous and energy-efficient ferries and coastal transportation [6], [7], and autonomous navigation on the open sea [8]–[12]. However, there are limited open-source tools that support benchmarking and efficient development of autonomous vessel research.

Demonstrated by the widely used autonomous driving simulators CARLA [13] and SUMO [14], a realistic and configurable simulation environment is an essential component for developing autonomous vehicles. For maritime navigation, a realistic setup encompasses the ability to model heterogeneous (i.e., different vessel types, dynamics, and controllers) and rule-compliant traffic. Yet, existing vessel simulators [15]–[18] do only partially allow for such a realistic setup, and repeating simulations is often challenging as scenarios and parameter initialization cannot be shared easily. The open-source framework CommonOcean (**Com**posable benchmarks for **m**otion planning **on Ocean**s) [19] is ideal to build a realistic, repeatable, and configurable simulator as it already includes vessel dynamics and parameters for multiple vessel types, and parametrized formal maritime traffic rules.

In this paper, we propose the simulation environment, CommonOcean-Sim, that facilitates the development of motion planners for autonomous vessels (see Fig. 1). CommonOcean-Sim features a modular software architecture that is based on the CommonOcean framework to allow for simulating realistic and heterogeneous traffic situations. We demonstrate the capabilities of our simulator on multiple traffic situations with recorded traffic data and interactive vessels. Additionally, we implement measures to improve computational efficiency for large-scale simulations and empirically evaluate them. The key features of CommonOcean-Sim are:

- *Heterogeneous Traffic* – CommonOcean-Sim simulates vessels navigating (a) based on recorded real-world traffic data and (b) based on desired waypoints while adhering to traffic rules using different vessel dynamics and vessel types.
- *Configurability* – The modular software architecture eases the integration of custom motion planners, low-level controllers, vessel dynamics, and vessel types. The simulation run is specified through a configuration file that allows for adjusting many parameters, such as the reactivity of traffic participants.
- *Repeatability* – A simulation is unambiguously specified with a configuration file. The result of a simulation run is a CommonOcean XML file that is compatible with the modular CommonOcean cost functions, which ease reporting of performance and safety metrics.
- *Usability* – CommonOcean-Sim is released on commonocean.cps.cit.tum.de/commonocean-sim under the BSD 3-Clause License and will include Jupyter notebook tutorials.
- *Scalability* – Multiple software components can be adjusted to decrease runtime, and we demonstrate that the software scales to high traffic density.

The remainder of the paper is structured as follows: In Section II and III, we discuss related software tools with a focus on traffic simulators and present preliminary concepts. Then, we introduce CommonOcean-Sim in Section IV and

This work is supported by the National Science Foundation under Grant CNS-2111688.

[1]Hanna Krasowski is with the University of California, Berkeley, USA `krasowski@berkeley.edu`

[2]Stefan Schärdinger is with the Technical University of Munich, Germany `stefan.schaerdinger@tum.de`

propose runtime efficiency improvements in Section V. In Section VI, we evaluate CommonOcean-Sim in five traffic situations. Lastly, we discuss future extensions and conclude in Section VII.

II. Related Work

Due to advancements of autonomous systems and increasing traffic on waterways, there is an increasing interest in research on autonomous navigation of vessels, e.g., [8], [10]–[12], [20], [21]. However, only a few studies publish their scenarios [9], [22], [23] or even open-source software tools. The existing software tools can be categorized into single-agent and traffic simulations. For example, a prominent tool for high-fidelity single-agent simulation is the Marine Systems Simulator (MSS)[1]; a MATLAB tool for hydrodynamic simulations to facilitate maritime vessel control design, featuring dynamic models for ships and floating structures. A single-agent simulator that focuses more on motion planning is VRX [22], which is built on Gazebo and allows for modeling complex environmental disturbances such as waves.

For traffic simulators, existing open-source projects [15]–[18] have lower configurability, especially with respect to Convention on the International Regulations for Preventing Collisions at Sea (COLREGS) [24] compliance. uSimMarineV22 [15] is built on the modular framework MOOS[2], which focuses on high-fidelity mission planning of underwater and surface vehicles. While a COLREGS mode is implemented in MOOS, it is not easily re-parametrizable. The Multi-Agent Maritime Traffic Simulator [16] is based on the multi-agent environment NetLogo to simulate high traffic density with homogeneous agents, i.e., using the same controller and dynamics for all agents. Similarly, MultiVessel_Simulation [17] implements homogeneous multi-agent traffic with COLREGS-informed RRT-planning. In contrast, COLSim [18] supports heterogeneous traffic with different vessel types and also allows vessels to track pre-recorded maritime traffic data. However, the COLREGS-mode is not well parameterized, and there is no defined configuration specification, which is necessary for reproducibility. We summarize the features of the related open-source maritime traffic simulators in Table I.

To the best of our knowledge, CommonOcean-Sim is the only traffic simulator that uses unambiguous temporal-logic traffic rule formalizations of the COLREGS for decision-making of interactive traffic participants. Additionally, it is highly configurable, supporting heterogeneous traffic and leveraging real-world traffic data, various vessel types, and controllers. The tutorials and code documentation make CommonOcean-Sim easy to use, and the grounding in the CommonOcean framework allows for seamless benchmarking and reproduction of research.

[1]fossen.biz/MSS
[2]oceanai.mit.edu/moos-ivp

TABLE I
Related maritime traffic simulators and their features

Tool	**Heterogeneous traffic**	**COLREGS compliance**	**Real traffic data**	**Repeatability**	**Documentation**
uSimMarineV22 [15]	✓	◐	✗	◐	✓
Maritime Traffic Simulator [16]	✗	◐	✗	✗	✗
MultiVessel_Simulation [17]	✗	◐	◐	◐	◐
COLSim [18]	✓	◐	✓	✗	◐
CommonOcean-Sim (ours)	✓	✓	✓	✓	✓

III. Preliminaries

Let us briefly introduce the used notation. We use teletype font for Python objects and typewriter font for methods of algorithms, i.e., `Object` and `method`. The state of a vessel is $\mathbf{s} = [p_\mathrm{x}, p_\mathrm{y}, \phi, v]$ with $p_\mathrm{x} \in \mathbb{R}$ and $p_\mathrm{y} \in \mathbb{R}$ being Cartesian coordinates of the current position $\mathbf{p}$ of the vessel, the orientation ϕ and the velocity $v \in \mathbb{R}$ in the direction of $\phi \in [-\pi, \pi]$. The control input is denoted as $\mathbf{u}$ and is a vector of acceleration magnitude and yaw for the yaw-constrained model [19, Eq. (3)] and a vector of acceleration in the x and y direction of the Cartesian coordinate frame for the point mass model [19, Eq. (2)]. Note that the yaw-constrained model is similar to the unicycle model often used for mobile robots. Solving an model predictive control (MPC) problem leads to a sequence of control inputs for the whole prediction horizon T. We denote such a signal produced at time step t as $U_t = [\mathbf{u}_{t+1}, ..., \mathbf{u}_{t+T}]$. Additionally, let us denote the Minkowski distance function `MD` for two signals U_1 and U_2 as:

$$\mathtt{MD}(U_1, U_2) = \left(\sum_{i=1}^{T} (\mathbf{u}_{1,i} - \mathbf{u}_{2,i})^T (\mathbf{u}_{1,i} - \mathbf{u}_{2,i}) \right)^{1/2}.$$

A sequence of states is called trajectory and abbreviated by $\mathrm{T}_j = [\mathbf{s}_{0,j}, ..., \mathbf{s}_{N,j}]$ for vessel j for a time horizon N.

Our simulation software builds on two studies. First, CommonOcean [19] is used for traffic scenario representation, collision checking, vessel dynamics, and vessel types. Second, the Intelligent Sailing Model (ISM) [25] is used for traffic rule-compliant interactive vessels.

A. CommonOcean

CommonOcean benchmarks consist of three components: a maritime traffic scenario, a vessel model with a specific vessel type (i.e., a specific parameter set), and a cost function. The scenario describes the maritime traffic situation, i.e., includes trajectories of traffic participants, static obstacles, and traffic signs, and provides one or multiple planning problems to be solved by a motion planner [19]. The trajectories of traffic participants are usually based on maritime traffic data, i.e., Automatic Identification System (AIS) data, which is

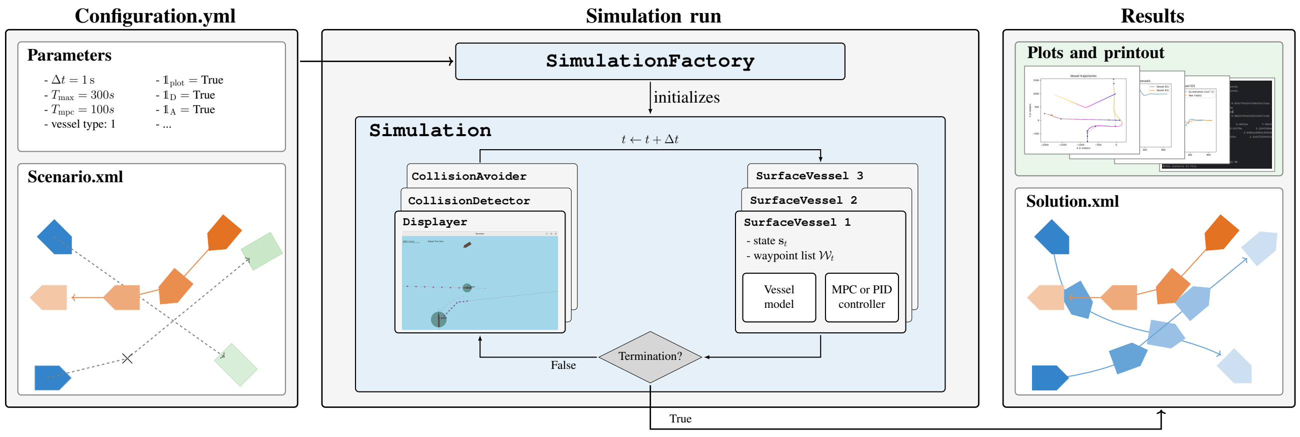


Fig. 1. CommonOcean-Sim pipeline with configuration YAML file, simulation run, and generated results. The traffic scenario consists of two planning problems with initial states displayed as blue vessel and goals in green, and a recorded trajectory. The `SimulationFactory` generates the `Simulation` according to the configuration and a simulation is iteratively stepping between the used `EventListener` objects (here `Displayer`, `CollisionDetector`, and `CollisionAvoider`), and the `SurfaceVessel` objects until termination. The produced results are plots, terminal printouts, and a CommonOcean XML file to store the simulated traffic situation.

transmitted via radio and often provided by Coast Guard offices[3]. A planning problem consists of an initial state s_0 and a goal region $\mathcal{G}$, which is usually specified by a set of positions, orientations, velocities, and a time interval. Optionally, the planning problem can contain a list of waypoints $\mathcal{W}$ that are intermediate goals, usually specified in the position domain. We abbreviate a planning problem PP with the tuple $(s_0, \mathcal{W}, \mathcal{G})$. The vessel models are commonly differential equations describing the kinematics or dynamics of a vessel, where the vessel type defines the parameters used for the vessel model. CommonOcean-Sim uses the yaw-constrained model and the point mass model of CommonOcean [19]. The CommonOcean cost functions facilitate the comparison of motion planners and are, therefore, not directly relevant to the simulation environment CommonOcean-Sim. Next to the CommonOcean scenarios and vessel models, the simulation environment needs the capability to detect collisions between vessels and solve the differential equations of the selected vessel model for a specified time step. These functionalities are implemented in the CommonOcean Drivability Checker and leveraged for the simulation environment. We refer the interested reader to [19] for more details on CommonOcean benchmarks and the Drivability Checker.

B. Intelligent Sailing Model (ISM)

While recorded data is typically more diverse, the lack of interaction between vessels can obscure its realism in robotics settings. For navigation on the open sea, the interactions of traffic participants are defined by traffic rules specified in the COLREGS [24]. However, there is no standard reactive sailing model for maritime navigation. This is in contrast to many robotic systems where interactive models are well established, e.g., driver models for road traffic [26] or encounter models for aerial traffic [27].

[3]CommonOcean benchmarks use a large AIS data set for the US coast available at: hub.marinecadastre.gov/pages/vesseltraffic.

Recently, the ISM [25] was proposed for open-sea traffic. The ISM can be used for interactive maritime traffic simulation of power-driven vessels, as the interaction of the vessels adheres to the rules specified in the COLREGS (more specifically, the rules 11, 13-17 formalized in [9], [23] with metric temporal logic). In particular, the ISM is a combination of a rule-compliant waypoint engine and a low-level model predictive controller. For each vessel, the waypoint engine checks if a traffic rule, based on the temporal logic traffic formalization, is applicable at the current state, given any other vessel in the traffic situation. If a traffic rule is applicable, the waypoint engine generates waypoints that characterize a rule-compliant path until the collision conflict with respect to the other vessel is resolved. The implementation of the waypoint engine in CommonOcean-Sim is the `CollisionAvoider` presented in Section IV-B. For control inputs in continuous state space, the waypoints are transformed into reference positions that are tracked by the low-level MPC, while the vessel dynamics, specifically the yaw-constrained model [19], is considered through constraints. The ISM can be used for different vessel types as demonstrated in [25]. Additionally, the formalized traffic rules are parametrized so that adjusting them, e.g., for near-shore navigation, is easily possible and would directly lead to an adapted waypoint engine as well. For more details on the ISM, we refer the interested reader to [25]. The ISM adds reactivity of traffic participants to CommonOcean-Sim and is, therefore, a core feature.

IV. COMMONOCEAN-SIM

The CommonOcean-Sim is a modular software for open sea traffic simulation. In this section, we introduce the pipeline of a simulation and the implemented Python classes. The overall pipeline is illustrated in Fig. 1. The input is a configuration YAML file, which includes a path to a CommonOcean benchmark file, specifying a maritime traffic situation. Based on the configuration file, the SimulatorFactory (see Subsection IV-A) creates the corresponding SurfaceVessel objects (see Subsection IV-C). The simulation is then performed in a time-discrete manner by iterating between displaying the simulation and stepping the SurfaceVessel objects for a time step with the provided control inputs and vessel dynamics. This simulation loop continues until all vessels terminate or a collision occurs, which is detected by the CollisionDetector (see Subsection IV-B). Upon termination, multiple result files are generated such as a CommonOcean benchmark XML file, which includes the simulated trajectories and logging of the control inputs of the simulated vessels (see Subsection IV-D for details). In the following subsections, we detail the different Python objects and their features.

A. SimulatorFactory

The simulation is initialized with a configuration file that contains general parameters to configure the simulation, e.g., the time step size, parameters to define the traffic situation, and parameters to configure the controlled vessels. We report the configuration parameters in Table II. The configuration file is loaded into the SimulationFactory, which outputs a Simulation object that includes SurfaceVessel objects and EventListener objects.

The to-be-created EventListener objects are defined through the configuration parameters $\mathbb{1}_C$ for using the CollisionDetector, $\mathbb{1}_A$ for using the CollisionAvoider, and $\mathbb{1}_D$ for using the Displayer (see Table II). The generation of the SurfaceVessel objects is more intricate since it has to differentiate between vessels that react to other traffic participants according to the traffic rules and vessels that are non-reactive. This differentiation is made based on the representation of the vessels in the CommonOcean benchmark file and on the value of $\mathbb{1}_A$. The planning problems PP_i in a CommonOcean benchmark file are transformed into a controllable vessel i for each planning problem. The vessel is reacting to traffic rules if $\mathbb{1}_A$ is set to true, i.e., is an ISM vessel, else the vessel is solving the defined motion planning task with a MPC or PID controller without reacting to other vessel. Each dynamic obstacle j in a CommonOcean benchmark file is translated to a vessel that follow the trajectory T_j defined in the CommonOcean Benchmark. The VesselFactory object is responsible for creating the SurfaceVessel objects according to this logic. Note that one can also use CommonOcean-Sim without a CommonOcean benchmark file, which can be useful for debugging purposes or investigating a specific maritime traffic situation. For example, we provide a few of these simulation profiles that are parametrized versions of commonly investigated traffic situations, such as head-on and crossing situations specified in the COLREGS.

B. EventListener

The EventListener is the base class for the Displayer, the CollisionDetector, and the CollisionAvoider objects. The Displayer is the object responsible for live rendering the simulation. It displays the position and orientation of the vessels for the current time step, the waypoints the vessels are currently tracking, and a scale so that spatial dimensions can be assessed (see Fig. 2(a) and (b)). The user change the centering and zoom of the window the Displayer produces by interactions with the keyboard and mouse. The CollisionDetector uses the collision checking functionalities of the CommonOcean Drivability Checker[4] to determine if any vessel is colliding with any other vessel or static obstacles at the current state. If a collision is detected, then the simulation is terminated at that time step. The CollisionAvoider monitors if a collision possibility exists between two vessels and generates rule-compliant waypoints. It effectively is an implementation of the waypoint engine presented in [25] and all vessels that are based on planning problems become ISM vessels if it is activated, i.e., $\mathbb{1}_A$ is true. The rule-compliant waypoints are provided to the respective vessels through their SurfaceVessel objects so that they can track the generated waypoints in order to exhibit interactive rule-compliant behavior typical for ISM vessels. For more details on the waypoint engine and ISM vessels, we refer the interested reader to [25].

The interface of EventListener objects with the simulation loop is the method state_change. In particular, the Simulator iterates over all EventListener objects when the simulation is initialized and at every time after

TABLE II
CONFIGURATION PARAMETERS FOR SIMULATION

Variable	Description	Unit	Mand.
General parameters			
Δt	Time step size	s	✓
$T_{\max}$	Maximum time steps	$\mathbb{N}^+$	✓
scenario	CommonOcean benchmark path	str	✓
vessel type	CommonOcean vessel type	enum	✓
type by id	Vessel type selection by id	dict	-
$\mathbb{1}_{\mathrm{C}}$	Boolean for using CollisionDetector	-	✓
$\mathbb{1}_{\mathrm{A}}$	Boolean for using CollisionAvoider	-	✓
$\mathbb{1}_{\mathrm{D}}$	Boolean for using Displayer	-	✓
$\mathbb{1}_{\mathrm{plot}}$	Boolean for generating result plots	-	✓
plots	Selects plots and runtime logging	list	-
$\mathbb{1}_{\mathrm{sr}}$	Boolean for saving results	-	✓
result	Path to result folder	str	-
$\mathbb{1}_{\mathrm{v}}$	Boolean for verbose logging	-	-
Controller parameters			
control type	MPC or PID controller	enum	✓
Δt_{mpc}	MPC max execution horizon	s	✓
T_{mpc}	MPC prediction horizon	$\mathbb{N}^+$	✓
T_{exe}	MPC maximum execution steps	$\mathbb{N}^+$	-
ϑ_{MPC}	threshold for MPC convergence	$\mathbb{R}$	-

[4]Available at: commonocean.cps.cit.tum.de/commonocean-dc.

the vessels are simulated for the new time step, i.e., after updating $t \leftarrow \Delta t + t$. This setup facilitates adding, removing, or creating `EventListener` objects.

C. SurfaceVessel

The `SurfaceVessel` object is used for representing vessels in the simulation. Its important attributes are the current state of the vessel, a list of the past states of the vessel, waypoints to reach, the spatial dimensions of the vessel, the vessel model including the specific parameters for a vessel type, and the current termination status. The `SurfaceVessel` objects interface with the `Simulation` object through the `get_next_state` method that computes the next state based on a controller and model or updates the next state with the one specified in the CommonOcean benchmark trajectory.

For computing the next states of controllable vessels, there are three different setups:

1) ISM vessel: The next state is computed as described in [25] and the vessel reacts to other traffic participants according to the formalized COLREGS;
2) MPC vessel: A MPC controller for the yaw-constrained model tracks waypoints that are defined in the planning problem;
3) PID vessel: A PID controller for a point-mass model minimizes the cross-track and along-track error to a waypoint path.

Note that for all three vessels, the vessel model and vessel type is specified based on the CommonOcean vessel model tool[5], which was originally introduced in [19].

D. Generated results

The results generated by a simulation run are a CommonOcean benchmark, multiple plots, and a terminal printout. The CommonOcean benchmark file contains all trajectories, including the control inputs of all vessels. The plots generated are showing:

1) the trajectories for all vessels including waypoints and the desired path
2) the cross-track error over time with respect to the path defined by waypoints for all controlled vessels
3) the velocities of all vessels over time
4) each control input dimension of all vessels over time
5) the control inputs over time for each controlled vessel

Note that the plots to be generated can be individually selected in the configuration file. Lastly, the terminal printout states the runtime for the different components, e.g., runtime for the `Displayer`, and the trajectory length for the simulated vessels.

V. Efficiency Improvements

For large-scale evaluation or reinforcement learning, runtime efficiency is typically a key factor for selecting a simulation environment. Thus, we include two features in CommonOcean-Sim to adjust and improve the runtime. First,

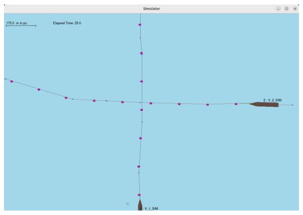
[5]Available at: commonocean.cps.cit.tum.de/commonocean-models.

Algorithm 1 Simulation loop with efficiency adjustments

```
1: Initialize: $t \leftarrow 0$, $k \leftarrow 0$, $\mathbf{s}_t \leftarrow \mathbf{s}_0$, $\mathcal{W}_t \leftarrow \mathcal{W}_0$,
   mpc_converged ← false
2: while ¬terminated do
3:    if $\mathbf{s}_t \neq \mathbf{s}_0 \wedge \mathcal{W}_t \neq \mathcal{W}_{t-1} \wedge k < T_{\text{exe}}$ then
4:       if ¬mpc_converged then
5:          $U \leftarrow$ compute_mpc_signal$(\mathbf{s}_t, \mathcal{W}_t)$
6:          $\mathbf{u}_t \leftarrow$ first$(U)$
7:          mpc_converged ← MD$(U_{\text{prev}}, U) \leq \vartheta_{MPC}$
8:       else
9:          $\mathbf{u}_t \leftarrow$ pop$(U)$
10:      end if
11:   else
12:      $U \leftarrow$ compute_mpc_signal$(\mathbf{s}_t, \mathcal{W}_t)$
13:      $\mathbf{u}_t \leftarrow$ first$(U)$
14:      $k \leftarrow 0$, mpc_converged ← false
15:   end if
16:   $U_{\text{prev}} \leftarrow U$
17:   $\mathbf{s}_{t+1} \leftarrow$ forward_simulation$(\mathbf{s}_t, \mathbf{u}_t, \Delta t)$
18:   $t \leftarrow t + 1$
19:   $W_t \leftarrow$ update_EventListeners( )
20: end while
```

the `Displayer` can be disabled so that the simulation is not rendered at runtime, i.e., $\mathbb{1}_D$ is set to false. Note that even if the `Displayer` is turned off, the resulting scenario XML file still contains all the states simulated, allowing for a comprehensive analysis. Second, the MPC computation can be reduced by checking if the new control signal is similar to the previous one.

We detail the reduction of MPC computations in Alg. 1, which also describes the overall simulation loop. In Line 1, the state $\mathbf{s}$, time t, waypoint list $\mathcal{W}$, control signal U, and the Boolean for checking the similarity of MPC signals, i.e. mpc_converged, are initialized. For the initial state, the simulation loop computes the MPC signal U, and extracts the first element with the method `first` in Lines 12-13. These lines are highlighted in blue since they are executed instead of Lines 3-15 if the efficiency adaptation is not used. For every following state, Line 3 checks if the waypoint list $\mathcal{W}$ has not changed since the last time step and the maximum of lazy execution steps T_{exe} is not reached. If this is the case and mpc_converged is false, then a new MPC signal is computed and its similarity to the previous one is checked with the Minkowski distance. If mpc_converged is true, then the next control input is extracted from the signal U with the function pop. In Lines 17-19, the vessel model is simulated for a time step Δt with the control input $\mathbf{u}_t$, the clock is increased, and the `EventListener` objects are updated. For clarity of presentation, we only describe the simulation loop for one vessel in Alg. 1. For each MPC or ISM vessel, Lines 3 - 17 are repeated. For each PID vessel, Lines 3-16 are replaced with the PID control function and executed together with line 17.


(a) t_1: Initial path

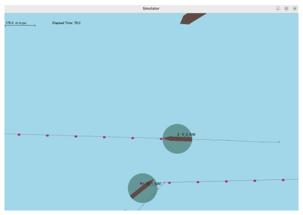
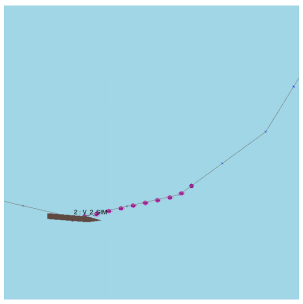
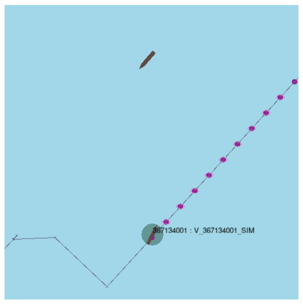
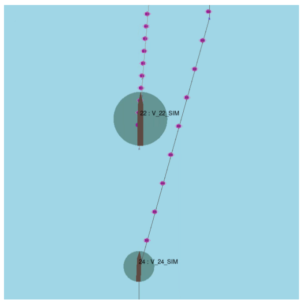

(b) t_2: Avoidance maneuver

Fig. 2. Screenshots of the `Displayer` window for two time steps (t_1 = 25s and t_2 = 78s) for scenario 1 with two ISM vessels and a recorded vessel. The vessels are shown in brown, the desired positions are purple dots, and the desired states computed from the MPC problem are pink diamonds. Gray circles around vessels indicate that a traffic rule is currently applied, and the waypoints are adapted so that the traffic rule-compliant collision avoidance maneuver is performed by the ISM vessels.


(2) Waypoint tracking


(3) Head-on


(4) Overtake

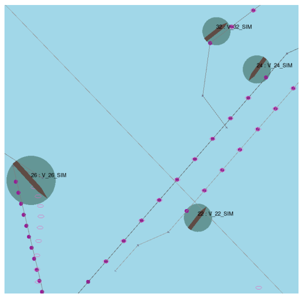

(5) High-density

Fig. 3. Snapshots from simulation scenarios 2-5 (left to right, top to bottom).

VI. Numerical Evaluation

We illustrate the features of CommonOcean-Sim with five traffic situations:

1) Crossing scenario between two ISM vessel and one recorded vessel;
2) Turning waypoint tracking of a MPC vessel;
3) Head-on scenario between one ISM vessel and a recorded vessel based on AIS data;
4) Overtake scenario between two ISM vessels;
5) High traffic density scenario with six ISM vessels.

For scenarios 1, 3, and 5, we use the parameter for a containership, i.e., vessel type 1, while in scenario 4 the overtaken vessel and in scenario 2 the MPC vessel are tankers, i.e., vessel type 2. We set $\Delta t = 1\,\mathrm{s}$, $T_{\max} = 3000$, $\Delta t_{\mathrm{mpc}} = 20\,\mathrm{s}$, $T_{\mathrm{mpc}} = 180$, and $T_{\mathrm{exe}} = 30$. The scenarios can be reproduced by executing the scripts in the experiments folder in the supplementary material. The runtime evaluation is performed on a server with an AMD EPYC 7742 2.2 GHz processor and 1024 GB of DDR4 3200 MHz memory and uses Gurobi as solver for the MPC problem.

First, let us illustrate a simulation run with scenario 1 (see Fig. 2). The two ISM vessels start at the bottom and right of the simulation window (see Fig. 2(a)). The path connecting the waypoints is gray, and the purple dots are the desired positions, which are moving with the vessel at each time step. The recorded vessel does not have the desired path since the trajectory is already pre-defined and is maneuvering at the top of the simulation window. At $t = 47\,\mathrm{s}$, the `CollisionAvoider` detects a crossing situation and Vessel 1 starts evading to the right while Vessel 2 keeps its course (see Fig. 2(b)). Overall, vessels 1 and 2 were simulated for $421\,\mathrm{s}$ and $252\,\mathrm{s}$, respectively, and the simulation took $19\,\mathrm{s}$ in wall-clock time.

Fig. 3 shows snapshots[6] of the simulations of scenarios 2 - 5. In scenario 2, an MPC vessel is tracking a waypoint path that includes many turns. Scenarios 3 and 4 are common encounter situations (i.e., head-on and overtaking) defined in the COLREGS and typically regarded for autonomous vessel research [23], [28]. In Scenario 3, the ISM vessel just finished its avoidance maneuver and is now navigating toward the original waypoints. In Scenario 4, the upper vessel is a tanker and is overtaken by the faster containership on its left side. Lastly, we showcase the scalability to high-density traffic with Scenario 5, which features six ISM vessels with many interactions.

Finally, we evaluate the empirical effects of the efficiency improvements detailed in Section V on scenario 5. We observe that the `Displayer` on average takes $0.01\,\mathrm{s}$ to render a new state. The runtime of the `CollisionDetector` is on average $0.0003\,\mathrm{s}$, which is negligible relative to the overall simulation runtime with six vessels of $0.12\,\mathrm{s} - 0.26\,\mathrm{s}$ per simulation step. To study the effect of different MPC parameters, we set $\mathbb{1}_{\mathrm{D}}$ to false and run Scenario 5, while we randomly select vessels for the runs with fewer than six vessels. The runtime evaluation is shown in Table III. The runtime increases linearly with the number of vessels.

TABLE III
Simulation Runtime Evaluation for Scenario 5

# Ves.	$T_{\mathrm{exe}} = \Delta t = 1\,\mathrm{s}$	$T_{\mathrm{exe}} = 9\,\mathrm{s}$	$T_{\mathrm{exe}} = 30\,\mathrm{s}$
1	0.0689 ± 0.0051	0.0235 ± 0.0059	0.0194 ± 0.0067
2	0.1109 ± 0.0222	0.0444 ± 0.0145	0.0461 ± 0.0273
3	0.1560 ± 0.0319	0.0645 ± 0.0310	0.0584 ± 0.0264
4	0.1759 ± 0.0337	0.0706 ± 0.0233	0.0831 ± 0.0360
5	0.2527 ± 0.0503	0.1485 ± 0.0647	0.1132 ± 0.0590
6	0.2661 ± 0.0081	0.1184 ± 0.0016	0.1226 ± 0.0058

Note: We report the average and standard deviation over 10 runs in seconds, and we set $T_{\mathrm{mpc}} = 30\,\mathrm{s}$ for all configurations.

[6]A video for the simulation runs is available at youtu.be/WfsO4caPlFU.

The runtime variance is significantly higher for settings with fewer than 6 vessels. Using the previously computed MPC control signal up to $T_{\mathrm{exe}} = 9$ halves the runtime in comparison to computing the control signal at every time step. Increasing T_{exe} to the maximum, i.e., $T_{\mathrm{exe}} = T_{\mathrm{mpc}}$, does not further reduce the runtime.

VII. Discussion and Conclusion

While the fundamental motion planning problem remains similar to the presented use cases, there are more navigation tasks for automation of vessels, e.g., docking [5], navigating coastal waters with shallows, static obstacles, traffic signs, and traffic separation zones [29]. Furthermore, beyond feedback and optimization-based controllers, sampling-based motion planners are common for maritime navigation [17], [18]. Thus, future work will extend CommonOcean-Sim to more navigation tasks and provide a larger variety of baseline controllers. Note that the CommonOcean benchmarks [19] already support traffic separation zones, traffic signs, shallows, and static obstacles, which will facilitate this extension.

We present a modular simulation software for motion planning of autonomous surface vessels: CommonOcean-Sim. The software is based on the CommonOcean framework that provides vessel dynamics, vessel types, and a standard representation of traffic scenarios. The implemented architecture ensures repeatable simulations, allows for easy extensions or adaptations of the surface vessel class, and provides multiple monitor classes to visualize and evaluate a simulation run. We illustrated different use cases of CommonOcean-Sim from multi-agent setups with the ISM to heterogeneous traffic situations with complex motion planning tasks specified through waypoints. We believe that CommonOcean-Sim is a significant step towards easing research on maritime navigation and will attract more researchers to the challenging automation tasks for vessels.

References

[1] European Maritime Safety Agency, "Annual overview of marine casualties and incidents 2024," EMSA, Tech. Rep., 2024.

[2] United Nations Conference on Trade and Development, "Review of maritime transport 2024 – navigating maritime chokepoints," United Nations, Tech. Rep., 2024.

[3] E. Dietrich, E. C. Gezer, B. Zhong, M. Arcak, M. Zamani, R. Skjetne, and A. J. Sørensen, "Symbolic control for autonomous docking of marine surface vessels," *arXiv*, no. 2501.13199, 2025.

[4] S. J. N. Lexau, M. Breivik, and A. M. Lekkas, "Automated docking for marine surface vessels — A survey," *IEEE Access*, vol. 11, pp. 132 324–132 367, 2023.

[5] A. B. Martinsen, A. M. Lekkas, and S. Gros, "Autonomous docking using direct optimal control," in *Proc. of the IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles (CAMS)*, 2019, pp. 97–102.

[6] W. Wang, B. Gheneti, L. A. Mateos, F. Duarte, C. Ratti, and D. Rus, "Roboat: An autonomous surface vehicle for urban waterways," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019, pp. 6340–6347.

[7] W. Wang, T. Shan, P. Leoni, D. Fernández-Gutiérrez, D. Meyers, C. Ratti, and D. Rus, "Roboat II: A novel autonomous surface vessel for urban environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020, pp. 1740–1747.

[8] 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.

[9] H. Krasowski and M. Althoff, "Provable traffic rule compliance in safe reinforcement learning on the open sea," *IEEE Transactions on Intelligent Vehicles*, pp. 1–18, 2024.

[10] 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. of the European Control Conf. (ECC)*, 2019, pp. 2536–2542.

[11] E. Meyer, A. Heiberg, A. Rasheed, and O. San, "COLREG-compliant collision avoidance for unmanned surface vehicle using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 165 344–165 364, 2020.

[12] A. Tsolakis, R. R. Negenborn, V. Reppa, and L. Ferranti, "COLREGs-aware trajectory optimization for autonomous surface vessels," *IFAC-PapersOnLine*, vol. 55, no. 31, pp. 269–274, 2022.

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. of the Annual Conf. on Robot Learning*, 2017, pp. 1–16.

[14] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using SUMO," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2018, pp. 2575–2582.

[15] M. Benjamin, "uSimMarineV22: Vehicle simulation," Massachusetts Institute of Technology, Tech. Rep., 2022.

[16] L. Grgičević, "Multi-Agent Maritime Traffic Simulator," *Modeling, Identification and Control*, vol. 45, no. 4, pp. 127–136, 2024.

[17] M. Bayrak and H. Bayram, "COLREG-compliant simulation environment for verifying USV motion planning algorithms," in *OCEANS*, 2023, pp. 1–10.

[18] B. Clement, T. Chaffre, P. Sarhadi, and M. Dubromel, "COLSim, a simulator for hybrid navigation acceptability and safety," *IFAC-PapersOnLine*, vol. 58, no. 20, pp. 147–152, 2024, iFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles CAMS.

[19] H. Krasowski and M. Althoff, "CommonOcean: Composable benchmarks for motion planning on oceans," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2022, pp. 1676–1682.

[20] L. Grgičević, E. M. Coates, R. T. Bye, T. I. Fossen, and O. L. Osen, "Towards decision support in vessel guidance using multi-agent modelling," in *Proc. of the European Control Conf. (ECC)*, 2024, pp. 1131–1138.

[21] P. Namal Senarathne, W. S. Wijesoma, K. W. Lee, B. Kalyan, M. Moratuwage, N. M. Patrikalakis, and F. S. Hover, "MarineSIM: Robot simulation for marine environments," in *OCEANS IEEE*, 2010.

[22] B. Bingham, C. Agüero, M. McCarrin, J. Klamo, J. Malia, K. Allen, T. Lum, M. Rawson, and R. Waqar, "Toward maritime robotic simulation in Gazebo," in *OCEANS MTS/IEEE*, 2019, pp. 1–10.

[23] H. Krasowski and M. Althoff, "Temporal logic formalization of marine traffic rules," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2021, pp. 186–192.

[24] "COLREGs: Convention on the international regulations for preventing collisions at sea," International Maritime Organization (IMO), 1972.

[25] H. Krasowski, S. Schärdinger, M. Arcak, and M. Althoff, "Intelligent sailing model for open sea navigation," *arXiv*, no. 2501.04988, 2025.

[26] B. N. Matcha, S. N. Namasivayam, M. Hosseini Fouladi, K. C. Ng, S. Sivanesan, and S. Y. Eh Noum, "Simulation strategies for mixed traffic conditions: A review of car-following models and simulation frameworks," *Journal of Engineering*, vol. 2020, no. 1, 2020.

[27] M. J. Kochenderfer, M. W. M. Edwards, L. P. Espindle, J. K. Kuchar, and J. D. Griffith, "Airspace encounter models for estimating collision risk," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 2, pp. 487–499, 2010.

[28] T. R. Torben, J. A. Glomsrud, T. A. Pedersen, I. B. Utne, and A. J. Sørensen, "Automatic simulation-based testing of autonomous ships using Gaussian processes and temporal logic," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 237, no. 2, pp. 293–313, 2023.

[29] X. Zhang, C. Wang, L. Jiang, L. An, and R. Yang, "Collision-avoidance navigation systems for maritime autonomous surface ships: A state of the art survey," *Ocean Engineering*, vol. 235, no. 109380, 2021.