IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 23, 2020, accepted September 1, 2020, date of publication September 8, 2020,
date of current version September 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3022600

COLREG-Compliant Collision Avoidance for Unmanned Surface Vehicle Using Deep Reinforcement Learning

EIVIND MEYER[1], AMALIE HEIBERG[1], ADIL RASHEED[1,2], AND OMER SAN[3]
[1]Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7034 Trondheim, Norway
[2]Department of Mathematics and Cybernetics, SINTEF Digital, 7031 Trondheim, Norway
[3]School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, OK 74078-5016, USA

Corresponding author: Adil Rasheed (adil.rasheed@ntnu.no)

This work was supported by the Norwegian Research Council and the Industrial Partners: DNV GL, Kongsberg, and Maritime Robotics of the Autosit Project under Grant 295033.

ABSTRACT Path Following and Collision Avoidance, be it for unmanned surface vessels or other autonomous vehicles, are two fundamental guidance problems in robotics. For many decades, they have been subject to academic study, leading to a vast number of proposed approaches. However, they have mostly been treated as separate problems, and have typically relied on non-linear first-principles models with parameters that can only be determined experimentally. The rise of deep reinforcement learning in recent years suggests an alternative approach: end-to-end learning of the optimal guidance policy from scratch by means of a trial-and-error based approach. In this article, we explore the potential of Proximal Policy Optimization, a deep reinforcement learning algorithm with demonstrated state-of-the-art performance on Continuous Control tasks, when applied to the dual-objective problem of controlling an autonomous surface vehicle in a COLREGs compliant manner such that it follows an a priori known desired path while avoiding collisions with other vessels along the way. Based on high-fidelity elevation and AIS tracking data from the Trondheim Fjord, an inlet of the Norwegian sea, we evaluate the trained agent's performance in challenging, dynamic real-world scenarios where the ultimate success of the agent rests upon its ability to navigate non-uniform marine terrain while handling challenging, but realistic vessel encounters.

INDEX TERMS Deep reinforcement learning, autonomous surface vehicle, collision avoidance, path following, machine learning controllers, the international regulations for preventing collisions at sea (COLREGs).

I. INTRODUCTION

Autonomous vehicles is one of the most interesting prospects associated with the rise of Artificial Intelligence (AI) and Machine Learning (ML) in recent years. Specifically, the success of Deep Learning (DL) applications in an ever-increasing number of domains, ranging from computer vision to imperfect-information games, has put the attractive but difficult proposal of self-driving vehicles on the horizon of technological development. While automated path following, at least in the maritime domain, has been a relatively trivial endeavor in the light of classical control theory and is a well-established field of research [1]–[11], considerably more advanced capabilities are required to navigate unknown, dynamic environments; characteristics that, generally speaking, apply to the real world. Reactive collision avoidance, i.e. the ability, based on a sensor-based perception of the local environment, to perform evasive manoeuvres that mitigate collision risk, remains a very challenging undertaking (e.g., see [12]–[15]). This is not to say, however, that the topic is not well-researched; a wide variety of approaches have been proposed, including especially (but not exhaustively) artificial potential field methods [16]–[18], dynamic window methods [19]–[21], velocity obstacle methods [22], [23] and optimal control-based methods [24]–[28]. However, it appears from a literature review that, when applied to autonomous vehicles with non-holonomic and real-time constraints, the approaches suggested so far suffer from one or more of the following drawbacks [29]–[32]:

- Unrealistic assumptions, or neglect, of the vessel dynamics.

The associate editor coordinating the review of this manuscript and approving it for publication was Jihwan P. Choi.

- Inability to scale to environments of non-trivial complexity (e.g. multi-obstacle scenarios).
- Requirement of a precise mathematical model of the controlled vessel.
- Disregard for desirable output trajectory properties, including smoothness, continuity, feasibility and safety.
- Incompatibility with external environmental forces such as wind, currents and waves.
- Excessive computation time requirements.
- Stability issues caused by singularities.
- Sub-optimal outputs due to local minima.

Although all the drawbacks are worth addressing, we noticed that the first four can be partially tackled using the advanced machine learning algorithm like Deep Reinforcement Learning (DRL). Recently, DRL, which combines DL and Reinforcement Learning (RL) principles, has achieved ground-breaking performance in the field of robotics, healthcare, video games, computer vision, education, transportation, and finance [33]. In the field of robotics, DRL has been used to teach a humanoid robot to walk safely [34], to achieve multi-agent tasks in the context of mobile robots [35], to learn complex reward functions by observing the behaviour of another robot [36], and for path smoothing and control tracking in robotic vehicle navigation [37]. In the context of under-actuated surface vehicles, the methods have also demonstrated remarkable potential, yielding promising results in a multitude of studies, including [38]–[42] for the path following problems and [43]–[46] for the collision avoidance ones. However, to our knowledge, none of the previous works have demonstrated path following and collision avoidance utilising range finder sensors within a RL framework and further demonstrated it to work in a real environment using historical vessel trajectories obtained from automatic identification system (AIS) data. As such, this article's major contributions are • exploring the feasibility of using DRL for the complex task of path following and collision avoidance with moving obstacles while still respecting the collision avoidance regulations (COLREGs), • demonstrating that an DRL agent trained in a very synthetic environment can generalize its learning to an entirely different, unseen and realistic environment, • combining DRL with range finder sensor input and showing its effectiveness, and finally • developing a new reward function that can achieve the full tasks mentioned above.

In order to achieve these contributions we simulate a small-sized supply ship model equipped with a rangefinder sensor suite and train a DRL-based controller using proximal policy optimization (PPO) in a purely synthetic environment. The PPO algorithm described was chosen due to its ability to handle continuous action and state spaces and its convincing track records [47], [48]. A carefully constructed reward function, which balances the prioritization of path adherence versus that of collision avoidance (which can be considered competing objectives), is used to guide the agent's learning process. Finally, we evaluate its performance in challenging, dynamic test scenarios reconstructed from real-world terrain and maritime traffic data. Towards the end, we highlight the strengths and weaknesses of our approach and discuss our next steps for improving the approach, addressing the rest of the drawbacks mentioned above, as well as the weaknesses of the current approach.

II. MOTIVATION

Arguably, the most promising aspect of autonomous vessels is not the obvious economic impact resulting from increased efficiency and the replacement of costly human labor, but instead the potential to eliminate injuries and material damage caused by collisions. According to the European Maritime Safety Agency, which annually publishes statistics on maritime accidents related to the EU member states, almost half of casualties at sea are ''navigational in nature, including contact, collision and grounding or stranding'' [49].

Validating a DRL-based approach to vessel guidance in a simulated environment can pave the way for applying the technology on a real, physical vessel. Since maritime collisions, of which 65.8% can be attributed to human error [50], account for hundreds of injuries each year in the EU alone, a positive result could be a preliminary step on the important path towards the adoption of AI systems for autonomous vessel guidance. Due to the limitations of existing methods, this is yet to take place on a large scale.

III. BACKGROUND

A. MARITIME NAVIGATION RULES

For collision avoidance at sea, adherence to the International Regulations for Preventing Collisions at Sea (COLREGs) [51] is crucial. Before autonomous vessels became a possibility, the COLREGs were formulated to prevent collisions between two or more vessels. The two main takeaways from these rules relevant for this work are; 1) the give way vessel should take early and substantial action, and 2) safe speed should be ensured at all times, such that course alteration is effective towards avoiding collisions where there is sufficient sea-room. Furthermore, the following rules provide clear instructions on how maritime vessels should behave upon encounters with other ships. **Rule 14: Head-on situation**

> *(a) When two power-driven vessels are meeting on reciprocal or nearly reciprocal courses so as to involve risk of collision each shall alter her course to starboard so that each shall pass on the port side of the other.*

Rule 15: Crossing situation

> *When two power-driven vessels are crossing so as to involve risk of collision, the vessel which has the other on her own starboard side shall keep out of the way and shall, if the circumstances of the case admit, avoid crossing ahead of the other vessel.*

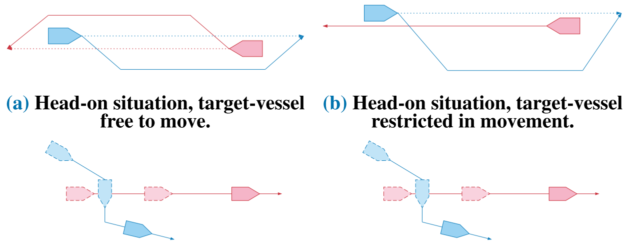
Rule 16: Action by give-way vessel

> *Every vessel which is directed to keep out of the way of another vessel shall, so far as possible, take early and substantial action to keep well clear.*

Rule 18: Responsibilities between vessels

> *(a) A power-driven vessel underway shall keep out of the way of:*
> *(ii) a vessel restricted in her ability to manoeuvre.*

Since the vessel controlled by the RL agent (the own-ship) is significantly smaller than the vessels encountered, it is, as a result of Rule 18, required to act as the give-way vessel in all situations. Figure 1 demonstrates the scenario and expected behaviour of the own-ship.



(a) Head-on situation, target-vessel free to move.

(b) Head-on situation, target-vessel restricted in movement.

(c) Crossing situation, target-vessel free to move.

(d) Crossing situation, target-vessel restricted in movement.

FIGURE 1. Expected behavior from the own-ship (colored in blue) in head-on and crossing encounters according to COLREGs.

B. DYNAMICS OF A MARINE VESSEL

1) COORDINATE FRAMES

In order to model the dynamics of marine vessels, one must first define the coordinate frames. Two coordinate frames typically used in vehicle control applications are of particular interest: The geographical North-East-Down (NED) and the body frame. The NED reference frame $\{n\} = (x_n, y_n, z_n)$ forms a tangent plane to the Earth's surface, making it useful for terrestrial navigation. Here, the x_n-axis is directed north, the y_n-axis is directed east and the z_n-axis is directed towards the center of the earth.

Assumption 1 (State space restriction): The vessel is always located on the surface and thus there is no heave motion. Also, there is no pitching or rolling motion.

The origin of the body-fixed reference frame $\{b\} = (x_b, y_b, z_b)$ is fixed to the current position of the vessel in the NED-frame, and its axes are aligned with the heading of the vessel such that x_b is the longitudinal axis, y_b is the transversal axis and z_b is the normal axis pointing downwards. However, as the vessel is restricted to surface level motion in our application, only the North and East components are of interest.

2) STATE VARIABLES

Following SNAME notation [52], the state vector consists of the generalized coordinates $\boldsymbol{\eta} = [x^n, y^n, \psi]^T$, where x^n and y^n are the North and East positions, respectively, in the reference frame $\{n\}$, and ψ is the yaw angle, i.e. the current angle between the vessel's longitudinal axis x_b and the North axis x_n. Correspondingly, the translational and angular velocity vector $\boldsymbol{\nu} = [u, v, r]^T$ consists of the surge (i.e. forward) velocity u, the sway (i.e. sideways) velocity v as well as yaw rate r.

3) VESSEL MODEL

To facilitate further research, we base the vessel dynamics on CyberShip II, a 1:70 scale replica of a supply ship which has a length of 1.255 m and mass of 23.8 kg [53]. Training the RL algorithm on a small vessel, such as CyberShip II, would allow for a relatively straight-forward deployment on a real-world model ship for further testing of the algorithm. However, the symbolic representation of the dynamics of a surface vessel, which is obtained from well-researched ship maneuvering theory, is the same regardless of the vessel - the distinctions lie solely in the numerical matrix parameters. Thus, if it can be demonstrated that an RL agent can control a small-sized model ship in an intelligent manner, there is reason to believe that controlling a full-sized ship would be within its reach.

As it is equipped with rudders and propellers aft, as well as one bow thruster fore, Cybership II is a fully actuated ship. This means that it could, in principle, be commanded to follow an arbitrary trajectory in the state space, as it is able to accelerate independently in every relevant degrees of freedom (DOF) simultaneously. However, for the purpose of simplifying the RL agent's action space, we disregard the bow thruster in this study and allow only the aft thrusters and control surfaces to be applied by the RL agent as control signals. This omission is further motivated by the fact that bow thrusters have limited effectiveness at higher speeds [54]. Thus, the control vector can be modelled as $\boldsymbol{f} = [T_u, T_r]^T$, where T_u represents the force input in surge and T_r represents the moment input in yaw.

Assumption 2 (Calm sea): There are no external disturbances to the vessel such as wind, ocean currents or waves.

Given Assumption 2, the 3-DOF vessel dynamics can be expressed in a compact matrix-vector form as

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \mathbf{R}_{z,\psi}(\boldsymbol{\eta})\boldsymbol{\nu} \\ \mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} &= \mathbf{B}\boldsymbol{f} \end{aligned} \tag{1}$$

where $\mathbf{R}_{z,\psi}$ represents a rotation of ψ radians around the z_n-axis as defined by

$$\mathbf{R}_{z,\psi} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Furthermore, $\mathbf{M} \in \mathbb{R}^{3\times3}$ is the mass matrix and includes the effects of both rigid-body and added mass, $\mathbf{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$ incorporates centripetal and Coriolis effects and $\mathbf{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$ is the damping matrix. Finally, $\mathbf{B} \in \mathbb{R}^{3\times2}$ is the actuator configuration matrix. The numerical values of the matrices are taken from [55], where the model parameters were estimated experimentally for CyberShip II in a marine control laboratory.

C. DEEP REINFORCEMENT LEARNING

Applications of RL on high-dimensional, continuous control tasks heavily rely on function approximators to generalize over the state space. Even if classical, tabular solution methods such as Q-learning can be made to work (provided a discretizing of the continuous action space), this is not considered an efficient approach for control applications [56]. In recent years, given their remarkable generalization ability over high-dimensional input spaces, the dominant approach has been the application of deep neural networks which are optimized by means of gradient methods. There are, however, different approaches to how the networks are utilized, and thus their semantic interpretation in the context of the learning agent differs. In Q-Learning-based methods such as Deep Q-Learning (DQN) [57], a deep neural network is used to predict the expected value (i.e. long-term, cumulative reward) of state-action pairs, which reduces the policy to an optimization problem over the set of available actions given the current state. In gradient-based policy methods, on the other hand, the policy itself is implemented as a deep neural network whose weights are optimized by means of gradient ascent (or approximations thereof). Lately, several algorithms built on this principle have gained a large traction in the RL research community, most notably Deep Deterministic Policy Gradient (DDPG) [56], Asynchronous Advantage Actor Critic (A3C) [58] and Proximal Policy Optimization (PPO) [59]. For continuous control tasks, this family of DRL methods is commonly considered to be the more efficient approach [60]. Based on previous work, where the PPO algorithm significantly outperformed other methods on a learning problem similar to the one covered in this study [61], we focus our efforts on this method.

1) RL PRELIMINARIES

First, we model the interplay between the agent and the environment as an infinite-horizon discounted Markov Decision Process (MDP), formally defined by the 6-tuple $(\mathcal{S}, \mathcal{A}, p, p_0, r, \Omega, o, \gamma)$ where

- $\mathcal{S}$ is the state space,
- $\mathcal{A}$ is the action space,
- $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ defines the conditional transition probabilities for the next state s' such that $p(s'|s, a) = Pr(S_{t+1} = s'|S_t = s, A_t = a)$,
- $p_0 : \mathcal{S} \rightarrow [0, 1]$ is initial state distribution, i.e. $p_o(s) = Pr(S_0 = s)$,
- $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ returns the numeric reward at each time-step as function of the current state and applied action,
- $\gamma \in [0, 1]$ is the discount factor for future rewards.

The agent draws its actions from its policy π. The policy may be a deterministic function (as in DDPG), but in the context of PPO, it is modelled as a stochastic function. The conditional action distribution given the current state s is given by $\pi(a|s) : \mathcal{S} \times \mathcal{A} \rightarrow\rightarrow [0, 1] = Pr(A_t = a|S_t = s)$. Specifically, we assume that the agent is drawing actions from a non-uniform multivariate Gaussian distribution whose mean is outputted by a neural network parametrized by the weights θ. Formally, this translates to $a_t \sim \pi(s_t)$, where t is the current time-step.

Next, we introduce the state-value function $V^\pi(s)$ and the action-value function $Q^\pi(s, a)$. $V^\pi(s)$ is the expected return from time t onwards given an initial state s, whereas $Q^\pi(s, a)$ is the expected return from time t onwards, but conditioned on the current action a_t. Formally, we have that

$$V^\pi(s_t) = \mathbb{E}_{s_{i\geq t}, a_{i\geq t} \sim \pi} \left[R_t|s_t\right] \tag{2a}$$

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_{i\geq t}, a_{i\geq t} \sim \pi} \left[R_t|s_t, a_t\right] \tag{2b}$$

where the random variable R_t represents the reward at time-step t.

2) POLICY GRADIENTS

The stochasticity of the policy enables us to translate the RL problem, i.e. the search for the optimal policy, into the problem of optimizing the expectation

$$J(\theta) = \mathbb{E}_{s_i, a_i \sim \pi(\boldsymbol{\theta})} \left[R_0\right] \tag{3}$$

The family of policy gradient methods, to which PPO belongs, approach gradient ascent by updating the parameter vector θ according to the approximation $\theta_{t+1} \leftarrow \alpha\theta_t + \widehat{\nabla_\theta J(\theta)}$, where $\widehat{\nabla_\theta J(\theta)}$ is a stochastic estimate of $\nabla_\theta J(\theta)$ satisfying $\mathbb{E}\left[\widehat{\nabla_\theta J(\theta)}\right] = \nabla_\theta J(\theta)$. From the policy gradient theorem [62] we have that the policy gradient $\nabla_\theta J(\theta)$ satisfies

$$\nabla_\theta J(\theta) \propto \sum_s \mu(s) \sum_a \nabla_\theta \pi(a|s) Q^\pi(s, a) \tag{4}$$

here μ is the steady state distribution under π such that $\mu(s) = \lim_{t\to\infty} Pr\{S_t = s|A_{0:t-1} \sim \pi\}$. Following the steps outlined in [63], this can be algebraically transformed to

$$\nabla_\theta J(\theta) \propto \mathbb{E}_\pi \left[\nabla_\theta \ln \pi(A_t|S_t) Q^\pi(S_t, A_t)\right] \tag{5}$$

Also, it can be shown that one can greatly reduce the variance of this expression by replacing the state-action value function $Q^\pi(s, a)$ in Equation 4 by $Q^\pi(s, a) - b(s)$, where the **baseline** function $b(s)$ can be an arbitrary function not depending on the action a, without introducing a bias in the estimate. Commonly, $b(s)$ is set to be the state value function V^π, which yields the **advantage** function

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \tag{6}$$

which represents the expected improvement obtained by an action compared to the default behavior. This leads to

$$\nabla_\theta J(\theta) \propto \mathbb{E}_\pi \left[\nabla_\theta \log \pi(A_t|S_t) A^\pi(s, a)\right] \tag{7}$$

Thus, an unbiased empirical estimate based on N episodic policy rollouts of the policy gradient $\nabla_\theta J(\theta)$ is

$$\widehat{\nabla_\theta J(\theta)} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{\infty} \hat{A}_t^n \nabla_\theta \log \pi(a_t^n|s_t^n) \tag{8}$$

$A^\pi(s, a)$ is, like $Q^\pi(s, a)$ and $V^\pi(s)$, unknown, and must thus be estimated by the function approximator $\hat{A}(s)$. Generalized Advantage Estimation (GAE), as proposed in [64], is the most notable approach. GAE makes use of a function approximator (commonly a neural network) $\hat{V}(s)$ to approximate the actual value function $V(s)$. A common approach is to use an artificial neural network, which is trained on the discounted empirical returns.

3) PROXIMAL POLICY OPTIMIZATION

PPO, as well as its predecessor Trust Region Policy Optimization (TRPO [65]) do not, even though it is feasible, optimize the policy directly via the expression in Equation 8. TRPO instead optimizes the surrogate objective function

$$J^{CPI}(\theta') = \hat{\mathbb{E}}_t\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\hat{A}_t^{\pi_\theta}\right] \tag{9}$$

which provides theoretical guarantees for policy improvement. However, as this relies on an approximation that is valid only in the local neighborhood, carefully choosing the step size is critical to avoid instabilities. Unlike in TRPO, where this is achieved by imposing a hard constraint on the relative entropy between the current and next policy, PPO elegantly incorporates the preference for a modest step-size in the optimization target, yielding a more efficient algorithm [59]. Specifically, it instead focuses on maximizing

$$\begin{aligned} J^{CLIP}(\theta') &= \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t^{\pi_\theta}, \operatorname{clip}_\epsilon\left(r_t(\theta)\right)\hat{A}_t^{\pi_\theta}\right)\right] \\ \operatorname{clip}_\epsilon(x) &= \operatorname{clip}\left(x, 1-\epsilon, 1+\epsilon\right) \end{aligned} \tag{10}$$

where $r_t(\theta)$ is a shorthand for the probability ratio $\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}$.

The PPO training process, which is written in pseudocode format in Algorithm 1, can then be summarized as follows: At each iteration, PPO first collects batches of Markov trajectories from concurrent rollouts of the current policy. Next, the policy is updated according to a stochastic gradient descent update scheme.

Algorithm 1 Proximal Policy Optimisation

```
for iteration = 1, 2, ... do
    for actor = 1, 2, ... $N$ do
        For $T$ time-steps, execute policy $\pi_\theta$.
        Compute advantage estimates $\hat{A}_1, \dots \hat{A}_T$
    for epoch = 1, 2, ... $N_E$ do
        Obtain mini batch of $N_{MB}$ samples from the $N_A T$ simulated time-steps.
        Perform Stochastic Gradient Decent (SGD) update from minibatch $(\mathbf{X}_{MB}, \mathbf{Y}_{MB})$.
        $\theta \leftarrow \theta'$
```

D. TERRAIN DATA

Our maritime simulation environment is made from a digital reconstruction of the Trondheim Fjord (Figure 2), an inlet of the Norwegian sea. Specifically, it is based on a digital terrain model (DTM) provided by the Norwegian Mapping Authority (Kartverket). The data set, which is called DTM10, is generated from airborne laser scanning, and has a horizontal resolution of $10m x 10m$ with coverage of the entire Norwegian mainland [66]. The coordinates are given according to the Universal Transverse Mercator (UTM) rectangular projection system, which partitions the Earth into 60 north-south zones, each of which has a 6 degree longitudinal span. Within each zone, which is indexed consecutively from zone 1 ($180°W$ to $174°W$) to zone 60 ($174°E$ to $180°E$), a mapping from latitude/longitude coordinates to a Cartesian x-y coordinate system is performed based on a local flat earth-assumption. Given the vast number of zones used in the UTM projection system, the approximated coordinates, which of course have inherent distortions because of the spherical shape of the Earth, are of relatively high accuracy. The DTM10 data set is given with respect to zone 33. Extracted terrain data is shown in Figure 3.

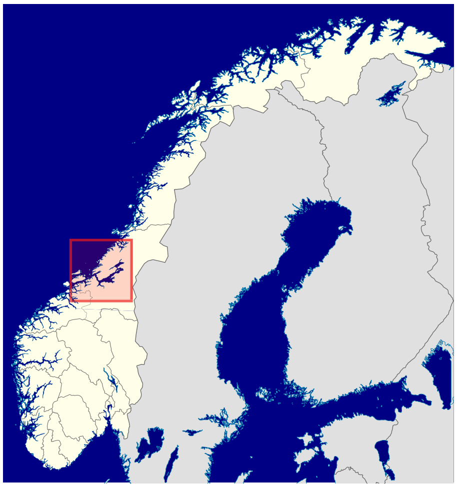


FIGURE 2. Map of the Norwegian mainland highlighting the area of interest.[1]

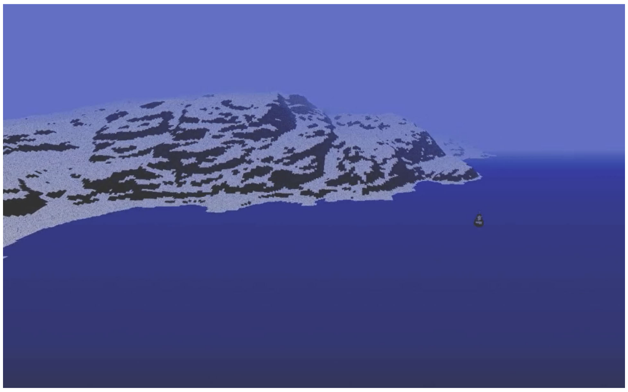


FIGURE 3. Digital terrain reconstructed from DTM10 (Norwegian Mapping Authority) rendered in 3D for debugging and showcasing purposes. Specifically, this shows a view of the Bymarka area, a nature reserve on the west side of Trondheim.

E. TRACKING DATA

We obtain a sample of historical vessel tracking data in the Trondheim Fjord area from a query of the Norwegian Coastal

[1] Original image source: NordNordWest (https://commons.wikimedia.org/wiki/File:Norway_location_map.svg), "Norway location map"

Administration's AIS Norway data service. The AIS is an automatic tracking system which provides both static (e.g. vessel dimensions) and dynamic (e.g. vessel position, heading and speed) information based on vessel transmissions. Within the field of autonomous surface vehicle guidance, AIS information is often used as a supplementary data source that is, by method of sensor fusion, combined with marine radar in collision avoidance algorithms. Additionally, given a large enough sample time within the area of interest, it provides a historical model of the marine traffic in the area. In our case, our historical data results from a 10 day data query ranging from January 26, 2020 to February 6, 2020 of all recorded traffic (Figure 4) within a rectangular area around the Trondheim Fjord.

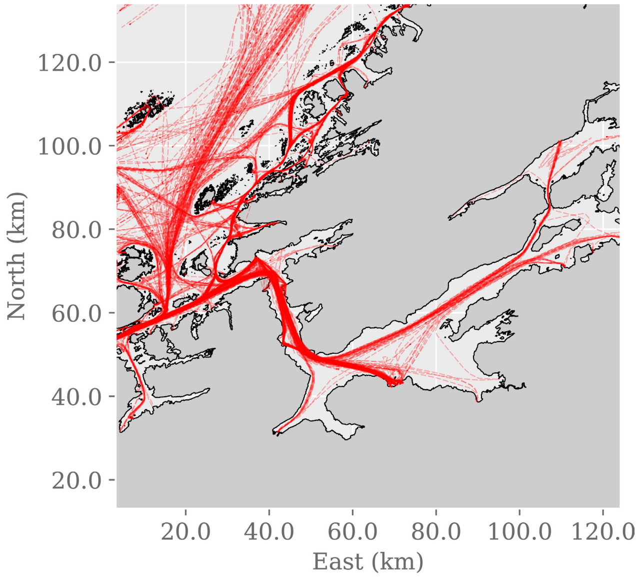


FIGURE 4. Snapshot of the marine traffic from January 26, 2020 to February 6, 2020 in the Trondheim Fjord area based on AIS tracking data. Each line represents one recorded travel.

Depending on the transmitter characteristics for each individual vessel, the resulting tracking data resolution varies from 2-20 seconds, facilitating a high-accuracy reconstruction of each vessel's trajectory in our simulation. As the AIS tracking data represents vessel position by latitude/longitude coordinates, a conversion to the zone 33 UTM x-y coordinate system is called for. To do the conversion, we utilize the *from_latlon* method provided by the Python package **utm** [67].

IV. METHODOLOGY

A. TRAINING ENVIRONMENT

DRL-based autonomous agents have a remarkable ability to generalize their policy over the observation space, including the domain of unseen observations. And given the complexity and heterogeneity of the Trondheim Fjord environment, with archipelagos, shorelines and skerries (see Figure 4), this ability will be fundamental to the agent's performance. However, the training environment, in which the agent is supposed to evolve from a blank slate to an intelligent vessel controller, must be both representative, challenging and unpredictable to facilitate the generalization. Of course, the most representative choice for a training scenario would be the Trondheim Fjord itself, which would, if it was not for the generalization issues associated with this approach [68], also allow for training the agent via behavior cloning based on the available vessel tracking data. However, given the resolution of our terrain data, the resulting obstacle geometry is typically very complex, leading to overly high computational demands for simulating the functioning of the distance sensor suite. Thus, the better choice is to carefully craft an artificial training scenario with simple obstacle geometries. To reflect the dynamics of a real marine environment, we let the stochastic initialization method of the training scenario spawn other target vessels with deterministic, linear trajectories. Additionally, circular obstacles, which are scattered around the environment, are used as a substitute for real-world terrain. A randomly chosen initialization of the training environment is shown in Figure 5.

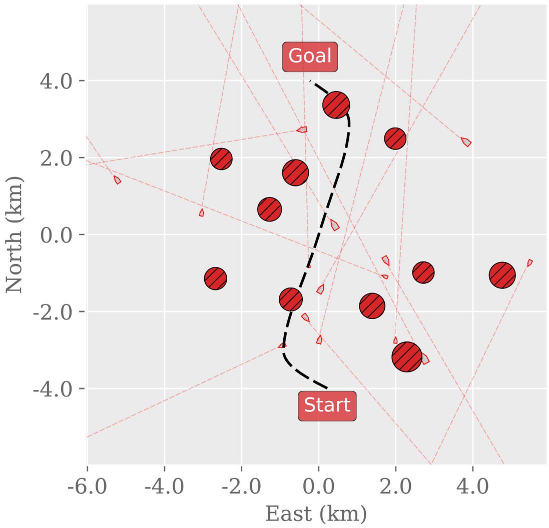


FIGURE 5. Random sample of the stochastically generated path following training scenario with moving obstacles. The circles are static obstacles, whereas the vessel-shaped objects are moving according to the trajectory lines.

B. OBSERVATION VECTOR

Here, the goal is to engineer an observation vector s containing sufficient information about the vessel's state relative to the path, as well as information from the sensors. To achieve this, the full observation vector is constructed by concatenating navigation-based and perception-based features, which formally translates to $s = [s_n, s_p]^T$. In the context of this article, we consider the term *navigation* as the characterization of the vessel's state, i.e. its position, orientation and velocity, with respect to the desired path. On the other hand, *perception* refers to the observations made via the rangefinder sensor measurements. In the following, the path navigation feature vector s_n and the elements culminating in the perception-based feature vector s_p are covered in detail.

1) PATH NAVIGATION

A sufficiently information-rich path navigation feature vector would be such that it, on its own, could facilitate a satisfactory path-following controller (without any consideration for obstacle avoidance). A few concepts often used in the field of vessel guidance and control are useful in order to formalize this.

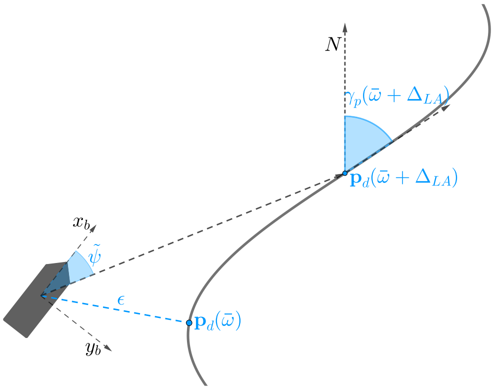

N
$\gamma_p(\bar{\omega}+\Delta_{LA})$
$\mathbf{p}_d(\bar{\omega}+\Delta_{LA})$
x_b
$\tilde{\psi}$
ϵ
$\mathbf{p}_d(\bar{\omega})$
y_b

FIGURE 6. **Illustration of key concepts for navigation with respect to path following. The path reference point $p_d(\omega)$, i.e. point yielding the closest Euclidean distance to the vessel, is here located right of the vessel, while the look-ahead reference point $p_d(\bar{\omega}+\Delta_{LA})$ is located a distance Δ_{LA} further along the path.**

First, we introduce the mathematical representation of the parameterized path, which is expressed as

$$\boldsymbol{p}_d(\omega) = [x_d(\omega), y_d(\omega)]^T \tag{11}$$

where $x_d(\omega)$ and $y_d(\omega)$ are given in the NED-frame. Navigation with respect to the path necessitates a reference point on the path which is continuously updated based on the current vessel position. Even though other approaches exist, this reference point is best thought of as the point on the path that has the closest Euclidean distance to the vessel, given its current position, as visualised in the example illustration shown in Figure 6. To find this, we calculate the corresponding value of the path variable $\bar{\omega}$ at each time-step. This is an equivalent problem formulation because the path is defined implicitly by the value of ω. Formally, this translates to the optimization problem

$$\bar{\omega} = \arg\min_{\omega} \left(x^n - x_d(\omega)\right)^2 + \left(y^n - y_d(\omega)\right)^2 \tag{12}$$

Which, using the Newton–Raphson method, can be calculated accurately and efficiently at each time-step. Here, the fact that the Newton–Raphson method only guarantees a local optimum is a useful feature, as it prevents sudden path variable jumps given that the previous path variable value is used as the initial guess [69].

Accordingly, we define the corresponding Euclidean distance to the path, i.e. the deviation between the desired path and the current track, as the cross-track error (CTE) ϵ. Formally, we thus have that

$$\epsilon = \left\| \left[x^n, y^n\right]^T - \boldsymbol{p}_d(\bar{\omega}) \right\| \tag{13}$$

TABLE 1. **Path-following feature vector s_n at timestep t.**

Feature	Definition
Surge velocity	$u^{(t)}$
Sway velocity	$v^{(t)}$
Yaw rate	$r^{(t)}$
Cross-track error	$\epsilon^{(t)}$
Heading error	$\tilde{\psi}^{(t)}$
Look-ahead heading error	$\tilde{\psi}_{LA}^{(t)}$

Next, we consider the look-ahead point $\boldsymbol{p}_d(\bar{\omega} + \Delta_{LA})$ to be the point which lies a constant distance further along the path from the reference point $\boldsymbol{p}_d(\bar{\omega})$. The parameter Δ_{LA}, the look-ahead distance, is set by the user and controls how aggressively the vessel should reduce the distance to the path. Look-ahead based steering, i.e. setting the look-ahead point direction as the desired course angle, is a commonly used guidance principle [70].

We then define the heading error $\tilde{\psi}$ as the change in heading needed for the vessel to navigate straight towards the look-ahead point from its current position, as illustrated in Figure 6. This is calculated from

$$\tilde{\psi} = \operatorname{atan2}\left(\frac{y_d(\bar{\omega}+\Delta_{LA}) - y^n}{x_d(\bar{\omega}+\Delta_{LA}) - x^n}\right) - \psi \tag{14}$$

where ψ is the vessel's current heading and x^n, y^n are the current NED-frame vessel coordinates as defined earlier.

However, even if minimizing the heading error will yield good path adherence, taking into account the path direction at the look-ahead point might improve the smoothness of the resulting vessel trajectory. Referring to the first order path derivatives as $x_p'(\bar{\omega})$ and $y_p'(\bar{\omega})$, we have that the path angle γ_p, in general, can be expressed as a function of arc-length ω such that

$$\gamma_p(\bar{\omega}) = \operatorname{atan2}\left(y_p'(\bar{\omega}), x_d'(\bar{\omega})\right) \tag{15}$$

As visualized in Figure 6, the path direction at the look-ahead point is then given by $\gamma_p(\bar{\omega} + \Delta_{LA})$. Accordingly, we can then define the look-ahead heading error, which is zero in the case when the vessel is heading in a direction that is parallel to the path direction at the look-ahead point, as

$$\tilde{\psi}_{LA} = \gamma_p(\bar{\omega} + \Delta_{LA}) - \psi \tag{16}$$

Our assumption is then that the navigation feature vector s_n, defined as outlined in Table 1, should provide a sufficient basis for the agent to intelligently adhere to the desired path.

Formally, we thus have that

$$s_n^{(t)} = \left[u^{(t)}, v^{(t)}, r^{(t)}, \epsilon^{(t)}, \tilde{\psi}^{(t)}, \tilde{\psi}_{LA}^{(t)}\right]^T \tag{17}$$

2) SENSING

Using a set of rangefinder sensors as the basis for obstacle avoidance is a well justified choice, as it yields a comprehensive, easily interpretable representation of the neighbouring obstacle environment. This should also enable a relatively straightforward transition from the simulated environment to a real-world one, given the availability of common rangefinder sensors, be it lidars, radars, sonars or depth cameras. In our setup, the vessel is equipped with N distance sensors with a maximum detection range of S_r, which are distributed uniformly with 360 degree coverage, as illustrated in Figure 7. While the area behind the vessel is obviously of lesser importance, and not necessary to consider for navigating purely static terrain [61], the possibility of overtaking situations where the agent must react to another vessel approaching from behind makes full sensor coverage a necessity.

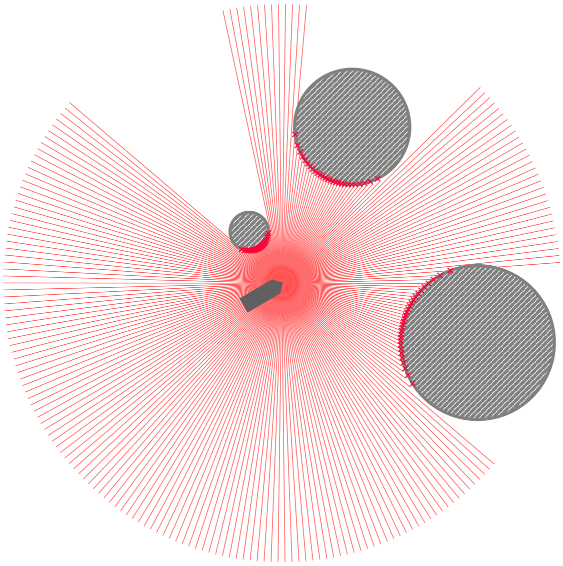


FIGURE 7. Rangefinder sensor suite attached to autonomous surface vessel.

3) SENSOR PARTITIONING

The most obvious approach to constructing the final observation vector would then be to concatenate the path information feature vector with the array of sensor outputs. However, initial experiments with this approach were aborted as it became apparent that the training process had stagnated - at a very dissatisfactory agent performance level. A likely explanation for this failure is the size of the observation vector which was fed to the agent's policy and value networks; as it becomes overly large, the agent suffers from the well-known *curse of dimensionality*. Due to the resulting network complexity, as well as the exponential relationship between the dimensionality and volume of the observation space, the agent fails to generalize new, unseen observations in an intelligent manner [71]. This calls for a significant dimensionality reduction. This can, of course, be achieved simply by reducing the number of sensors, something which would also have the fortunate side effect of reducing the simulation's computational needs. Unfortunately, this approach also turned out unsuccessful, even after testing a wide range of smaller sensor setups. Clearly, when the sensor count becomes too low, the agent's perception of the neighboring obstacle environment is simply too scattered to facilitate satisfactory obstacle-avoiding behavior in challenging scenarios such as the ones used for training the agent. As balancing the trade-off between sensor resolution and observation dimensionality appears intractable, this calls for a more involved approach.

The sensor suite is partitioned into D sectors, each of which produces a scalar measurement which is included in the final observation vector, effectively summarizing the local sensor readings within the sector. However, given our desire to minimize its dimensionality, dividing the sensors into sectors of uniform size is likely sub-optimal, as obstacles located in front of the vessel are significantly more critical and thus require a higher degree of perception accuracy than those that are located at its rear. In order to realize such a non-uniform partitioning, we use a logistic function - a choice that also fulfills our general preference for symmetry. Assuming a counter-clockwise ordering of sensors and sectors starting at the rear of the vessel, we map a given sensor index $i \in \{1, \ldots, N\}$ to sector index $k \in \{1, \ldots, D\}$ according to

$$\kappa : i \mapsto \kappa(i) = \left\lfloor \underbrace{D\sigma\left(\frac{\gamma_C i}{N} - \frac{\gamma_C}{2}\right)}_{\text{Non-linear mapping}} - \underbrace{D\sigma\left(-\frac{\gamma_C}{2}\right)}_{\text{Constant offset}} \right\rfloor \tag{18}$$

where σ is the logistic sigmoid function and γ_C is a scaling parameter controlling the density of the sector distribution such that decreasing it will yield a more evenly distributed partitioning. In Figure 8, the practical output of this sensor mapping procedure is visualised, with the sectors being the narrowest near the front of the vessel.

We can then formally define the distance measurement vector for the k^{th} sector, which we denote by $\boldsymbol{w}_k$, according to

$$\boldsymbol{w}_{k,i} = x_i \quad \text{for } i \in \{1, \ldots, N\} \text{ such that } \kappa(i) = k$$

Next, we seek a mapping $f : \mathbb{R}^n \mapsto \mathbb{R}$, which takes the vector of distance measurements $\boldsymbol{w}_k$, for an arbitrary sector index k, as input, and outputs a scalar value based on the current sensor readings within the sector. Always returning the smallest measured obstacle distance within the sector, i.e. $f = \min$ (in the following referred to as *min pooling*), is a natural approach which yields a conservative and thereby safe observation vector. As can be seen in Figure 9a, however, this approach might be overly restrictive in certain obstacle scenarios, where feasible openings in between obstacles are inappropriately overlooked. However, even if the opposite approach (*max pooling*, i.e. $f = \max$) solves this problem, it is straight-forward to see, e.g. in Figure 9b by considering the fact that the presence of the small obstacle near the vessel is ignored, that it might lead to dangerous navigation strategies. In order to alleviate the problems associated with

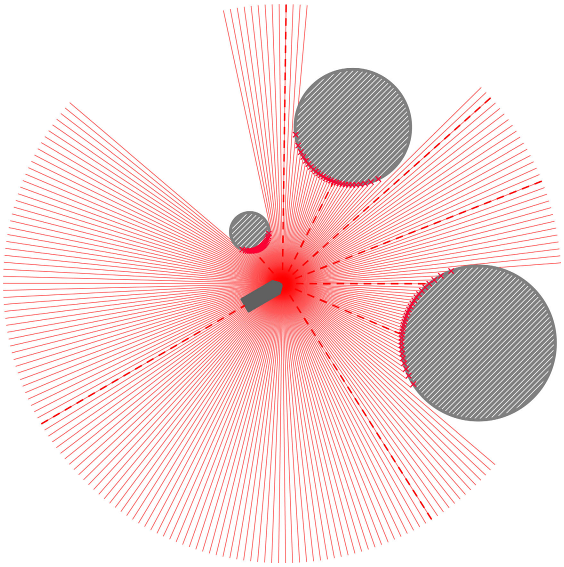
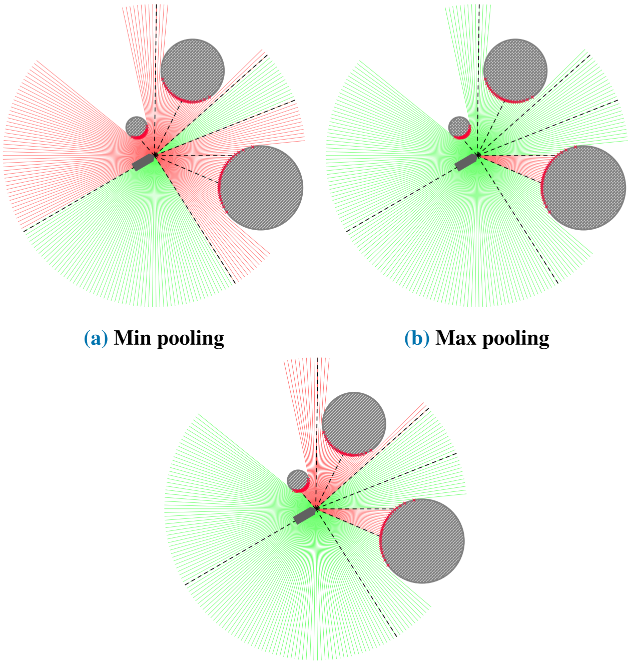


FIGURE 8. Rangefinder sensor suite partitioned into $D = 9$ sectors according to the the mapping function κ with the scale parameter $\gamma_C = 0.13$.

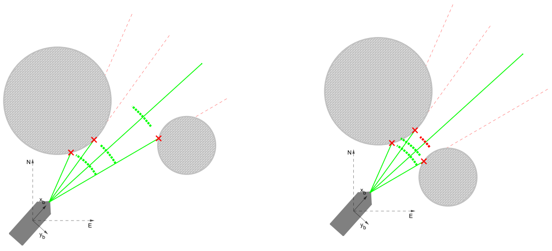


(a) Min pooling

(b) Max pooling

(c) Feasibility pooling

FIGURE 9. Pooling techniques for sensor dimensionality reduction. For the sectors colored green, the maximum distance S_r was outputted, implying that the sector is clear of any obstacles. It is obvious that min-pooling yields an overly restrictive observation vector, effectively telling the agent that a majority of the travel directions are blocked. On the other hand, max pooling yields overly optimistic estimates, potentially leading to dangerous situations. The feasibility pooling algorithm, however, mirrors an intuitive reasoning about the reachability within each sector, producing a more intelligent estimate.



(a) Full distance is reachable.

(b) Less than half the distance is reachable.

FIGURE 10. Illustration of the feasibility algorithm for two different scenarios. After sorting the sensor indices according to the corresponding distance measurements, the algorithm iterates over them in ascending order, and, at each step, decides if the vessel can feasibly continue past this point. In the scenario displayed in the figure on the right, the opening is deemed too narrow for the full distance to be reachable.

min and max pooling mentioned above, a new approach is required. The *feasibility pooling* procedure, which was introduced in [61], calculates the maximum reachable distance within each sector, taking into account the location of the obstacle sensor readings as well as the width of the vessel. This method requires us to iterate over the sensor reading in ascending order corresponding to the distance measurements, and for each resulting distance level check whether it is feasible for the vessel to advance beyond this level (see Figure 10). As soon as the widest opening available within a distance level is deemed too narrow given the width of the vessel, the maximum reachable distance has been reached. Formally, we define f to be the algorithm outlined in Algorithm 2.

Algorithm 2 Feasibility Pooling for Rangefinder Sensors [61]

```
Require:
  Vessel width $W \in \mathbb{R}^+$
  Angle between neighboring sensors $\theta$
  Sensor rangefinder measurements for current sector $\boldsymbol{x} = \{x_1, \ldots, x_n\}$
  procedure FeasibilityPooling($\boldsymbol{x}$)
     Initialize $\mathcal{I}$ to be the indices of $\boldsymbol{x}$ sorted in ascending order according to the measurements $x_i$
     for $i \in \mathcal{I}$ do
        Arc-length $d_i \leftarrow \theta x_i$
        Opening-width $y \leftarrow d_i/2$
        Opening was found $s_i \leftarrow false$
        for $j \leftarrow 0$ to $n$ do
           if $x_j > x_i$ then
              $y \leftarrow y + d_i$
              if $y > W$ then
                 $s_i \leftarrow true$
                 break
           else
              $y \leftarrow y + d_i/2$
              if $y > W$ then
                 $s_i \leftarrow true$
                 break
              $y \leftarrow 0$
        if $s_i$ is $false$ then return $x_i$
```

4) MOTION DETECTION

Simply feeding the pooled current rangefinder sensor readings to the agent's policy network, will, without any doubt, be insufficient for the agent to learn a policy for intelligently avoiding moving obstacles. A continuous snapshot of the environment can facilitate a purely reactive (but still intelligent [61]) agent in a static environment, but without explicit or implicit knowledge of the nearby obstacles' velocities, such an agent will invariantly fail when placed in a dynamic environment, as it will be unable to distinguish between stationary and moving obstacles.

An implicit approach worth mentioning is to process the sensor readings sequentially using a Recurrent Neural Network (RNN). In recent years, RNN architectures, such as Long Short-Term Memory LSTM, have gained a lot of traction in the ML research community [72] and been successfully applied to sequential RL problems. An example of this is the LSTM-based AlphaStar agent, which reached grandmaster level in the popular real-time strategy game StarCraft II [73]. It is therefore possible that a high-performing collision avoidance policy could be found by feeding a recurrent agent with sensor readings. If such an implementation was shown to be successful, it would facilitate a very straightforward transition to an implementation on a physical vessel, as no specialized sensor equipment for measuring object velocities would be needed. However, even if sequentially feeding sensor readings to a recurrent network might sound relatively trivial, the motion of the vessel would induce rotations of the observed environment, complicating the situation. Initial experimentation with an off-the-shelf recurrent policy compatible with our simulation environment confirmed the difficulties with this approach. Even with a purely static environment, the recurrent agent was incapable of learning how to avoid collisions.

Thus, this preliminary study will focus on the explicit approach, i.e. providing the obstacles' velocities as features in the agent's observation vector. Admittedly, while the implementation of this is trivial in a simulated environment, as obstacle velocities can simply be accessed as object attributes, a real-world implementation will necessitate a reliable way of estimating obstacle velocities based on sensor data. However, even if this can be challenging due to uncertainty in the sensor readings, object tracking is a well-researched computer vision discipline. We reserve the implementation of such a method to future research, but refer the reader to [74] for a comprehensive overview of the current state of the field.

For each sector, we provide the decomposed velocity of the closest moving obstacle within the sector as features for the agent's observation vector. Specifically, the decomposition, which yields the x and y component of the obstacle velocity, is done with respect to the coordinate frame in which the y-axis is parallel to the center line of the sensor sector in which the obstacle is detected. This is illustrated in Figure 11. For each sector k, we denote the corresponding decomposed x and y velocities as $v_{x,k}$ and $v_{y,k}$, respectively. If there are no moving obstacles present within the sector, both components are zero.

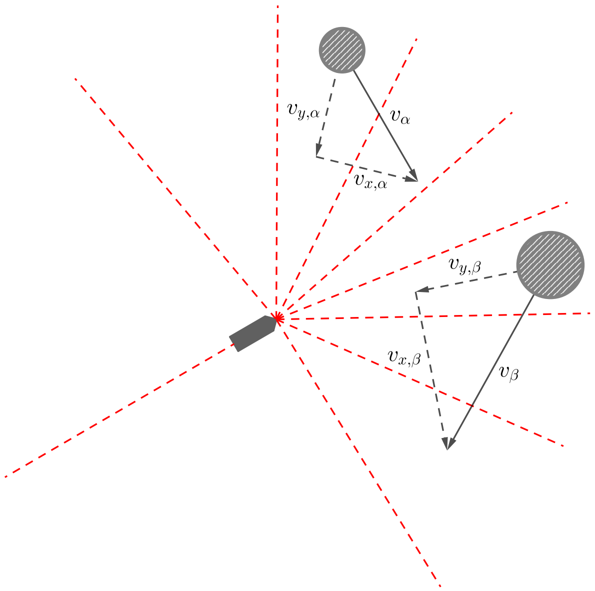


FIGURE 11. Velocity decomposition for two moving obstacles, α and β. For each obstacle, its velocity vector is decomposed into x and y components relative to the obstacle sector, such that the decomposed y-component is parallel to the center line of the corresponding sector, and has a positive value if it is moving *towards* the vessel.

5) PERCEPTION STATE VECTOR

As having access to both obstacle distances and obstacles velocities is critical to achieve satisfactory obstacle-avoiding agent behavior, we include both in the perception state vector.

To avoid discontinuities in the obstacle distance features caused by the sudden transition from 0 to S_r at the point of detection, we introduce the concept of obstacle *closeness*. The *closeness* to an obstacle is such that it is 0 if the obstacle is undetected, i.e. further away from the vessel than the maximum range of the distance sensors, and 1 if the vessel has collided with the obstacle. Furthermore, within this range, is it reasonable to map distance to closeness in a logarithmic fashion, such that, in accordance with human intuition, the difference between 10m and 100m is more significant than the difference between, for instance, 510m and 600m. Formally, we have that a distance d maps to closeness $c(d) : \mathbb{R} \mapsto [0, 1]$ according to

$$c(d) = \text{clip}\left(1 - \frac{\log(d+1)}{\log(S_r+1)},\ 0,\ 1\right) \tag{19}$$

By concatenating the reachable distance and the decomposed obstacle velocity from every sector, we then define the perception state vector s_p as

$$s_p^{(t)} = \left[\underbrace{c\left(\left(\boldsymbol{w}_1^{(t)}\right)\right),\ v_{x,1}^{(t)},\ v_{y,1}^{(t)},}_{\text{First sector}}\ \ldots\right]^T \tag{20}$$

C. REWARD FUNCTION

Any RL agent is motivated by the pursuit of maximizing its reward. The simplest, and thus highly sought-after approach to rewarding RL agents is to reward it at the end of each episodes - at that point, one already knows if the agent succeeded or failed. However, given the length of a full episode, such a reward function turns out extremely sparse, leaving the agent with a near impossible learning task. This calls for a continuous reward signal, rewarding the agent based on its current adherence to its objectives, i.e. how well it is currently doing with respect to both path following and obstacle avoidance. Given the complexity of the dual-objective learning problem focused on in this study, as well as the general tendency of RL agents' to exploit the reward function in any way possible (e.g. standing still, going in circles), designing an appropriate rewards function $r^{(t)}$ is paramount to the agent exhibiting the desired behavior after training.

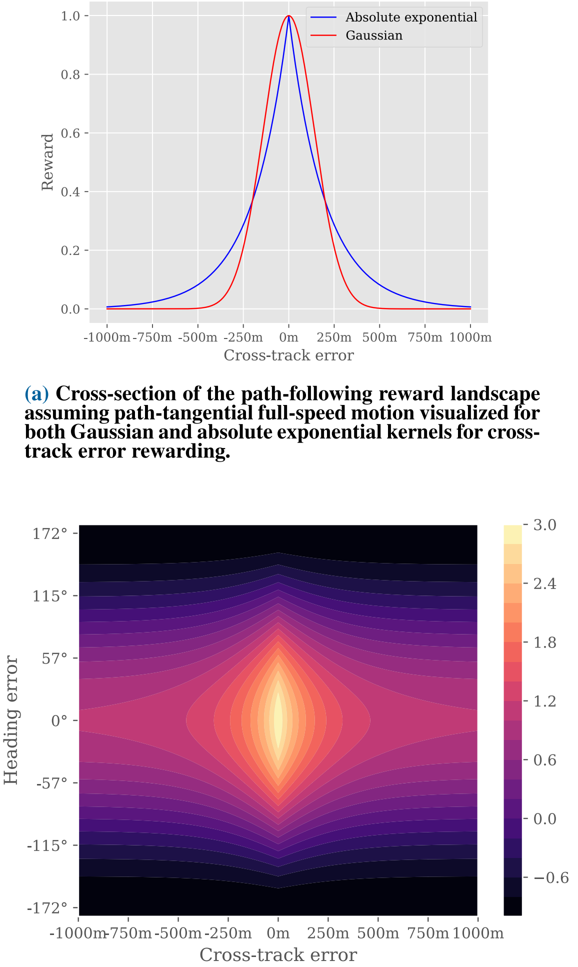
The agent is separately rewarded for its performance in the two relevant domains: path following and collision avoidance. Thus, we introduce the independent reward terms $r_{path}^{(t)}$ and $r_{colav}^{(t)}$, representing the path-following and the obstacle-avoiding reward components, respectively, at time t. Furthermore, as suggested in [61], we introduce the weighting coefficient $\lambda \in [0, 1]$ to regulate the trade-off between the two competing objectives. In addition, as it is crucial to penalize the agent whenever it collides with an obstacle, we represent this by the negative reward term $r_{collision}$, which is activated upon collision. This leads to the preliminary reward function

$$r^{(t)} = \begin{cases} r_{collision}, & \text{if collision} \\ \lambda r_{path}^{(t)} + (1-\lambda)\, r_{colav}^{(t)}, & \text{otherwise} \end{cases} \tag{21}$$

1) PATH FOLLOWING PERFORMANCE

An intuitive approach to incentivize path adherence is to reward the agent for minimizing the current absolute cross-track error $\left|\epsilon^{(t)}\right|$. In [69], a Gaussian reward function centered at $\epsilon = 0$ with standard deviation σ_e was suggested. However we argue that the absolute exponential reward function $\exp\left(-\gamma_\epsilon \left|\epsilon^{(t)}\right|\right)$ has more desirable characteristics due to its fatter tails, as seen in Figure 12a. By avoiding the vanishing improvement gradient of the Gaussian reward occurring at large absolute cross-track errors, the absolute exponential reward function ensures that the agent is rewarded even for a slight improvement to a very unsatisfactory state. However, this alone does not reflect our desire for the agent to actually make progress along the path - and thus, the RL agent, greedy as it is, will eventually develop a policy of standing still indefinitely after closing the gap to the path. Thus, the reward signal must be expanded upon so that it incorporates the incentivization of motion - and not just arbitrary motion, but movement in the right direction.

The already defined look-ahead heading error term $\tilde{\psi}$ is a natural basis for formalizing this. Specifically, we consider the term $\frac{u^{(t)}}{U_{max}} \cos \tilde{\psi}^{(t)}$, with U_{max} being the maximum vessel speed, which effectively yields zero reward if the vessel is heading in a direction perpendicular to the path, and a negative reward if the agent is tracking backwards. Multiplying this with the cross-track error reward component defined earlier is the obvious choice, and yields the provisional reward function



(a) Cross-section of the path-following reward landscape assuming path-tangential full-speed motion visualized for both Gaussian and absolute exponential kernels for cross-track error rewarding.

(b) Path-following reward function assuming full-speed motion.

FIGURE 12. Cross-section and level curves for the path-following reward function with $\gamma_\epsilon = 0.05$.

$$r_{path}^{(t)} = \underbrace{\frac{u^{(t)}}{U_{max}} \cos \tilde{\psi}^{(t)}}_{\text{Velocity-based reward}} \; \underbrace{\exp\left(-\gamma_\epsilon |\epsilon^{(t)}|\right)}_{\text{CTE-based reward}}$$

Given this reward function, however, we note that, if the vessel is standing still (i.e. $u^{(t)} = 0$), or if it is heading in a direction perpendicular to the path (i.e. $\tilde{\psi}^{(t)} = \pm\frac{\pi}{2}$), the agent will receive zero reward regardless of the cross-track error, which is undesired. Similarly, if the cross-track error grows very large, i.e. $\exp\left(-\gamma_\epsilon \left|\epsilon^{(t)}\right|\right) \rightarrow 0$, the reward signal will be zero regardless of the vessel velocity and heading. Thus, we add constant multiplier terms γ_r to both reward components, yielding the following expression for the final path-following reward function

$$r_{path}^{(t)} = \underbrace{\left(\frac{u^{(t)}}{U_{max}} \cos \tilde{\psi}^{(t)} + \gamma_r\right)}_{\text{Velocity-based reward}} \underbrace{\left(\exp\left(-\gamma_\epsilon |\epsilon^{(t)}|\right) + \gamma_r\right)}_{\text{CTE-based reward}} - \gamma_r^2 \tag{22}$$

where the $-\gamma_r^2$ term is added to remove the constant reward bias implied by the function choice.

2) STATIC OBSTACLE AVOIDANCE PERFORMANCE

Collision avoidance involves both collisions with other vessels as well as avoiding running ashore (or colliding with some other static obstacle). However, the two aspects should be treated separately, as would any human sailor. In the following, we refer to the former as dynamic, and the latter as static obstacle avoidance.

In order to encourage obstacle-avoiding guidance behavior, penalizing the agent for the closeness of nearby terrain in a strictly increasing manner seems reasonable. However, we note that the severity of closeness intuitively does not increase linearly with distance, but instead increases in some quasi-exponential fashion.

Furthermore, given the presence of a nearby static obstacle, it seems clear that the penalty given to the agent must depend on the orientation of the vessel with regards to the obstacle in such a manner that obstacles located near the stern of the vessel are of significantly lower importance than obstacles that are currently right in front of the it.

Thus, given a static obstacle located at distance x, at the angle θ with respect to the centerline of the vessel, we propose the penalty function

$$r_{obst,stat}^{(t)} = -\underbrace{\frac{1}{1 + \gamma_{\theta,stat}|\theta|}}_{\text{Weighting term}} \underbrace{\alpha_x \exp\left(-\gamma_x x\right)}_{\text{Raw closeness penalty}} \tag{23}$$

where α_x is in the order of magnitude of the sensor range, such that sufficiently high negative rewards are given as objects get closer to the own-ship.

For practical reasons, we use the distances measured by the rangefinder sensors as surrogates for obstacle closeness, and penalize each sensor reading according to $r_{obst,stat}(x_i, \theta_i)$, where x_i is the i^{th} distance sensor measurement and θ_i is the vessel-relative angle of the corresponding sensor ray. In order to to cancel the dependency on the specific sensor suite configuration, i.e. the number of sensors and their vessel-relative angles, that arises when this penalty term is summed over all sensors, we compute the overall static obstacle-avoidance reward according to the weighted average

$$r_{colav,stat}^{(t)} = -\frac{\sum_{i=1}^{N} \frac{1}{1 + \gamma_{\theta,stat}|\theta_i|} \alpha_x \exp\left(-\gamma_x x_i\right)}{\sum_{i=1}^{N} \frac{1}{1 + \gamma_{\theta,stat}|\theta_i|}} \tag{24}$$

which is visualised on a logarithmic scale in Figure 13.

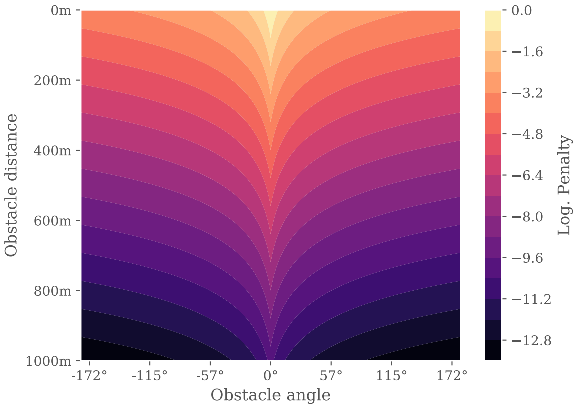


FIGURE 13. Static obstacle closeness penalty landscape as a function of obstacle distance and angle relative to the vessel with the scale parameters $\gamma_\theta = 10$, $\gamma_x = 0.1$. The maximum penalty is imposed for obstacles located right in front of the vessel.

3) DYNAMIC OBSTACLE AVOIDANCE PERFORMANCE

For dynamic obstacle avoidance, we expand on the framework developed for static obstacles. Firstly, the penalty needs to reflect the relevant COLREGs. Since the COLREGs are defined according to the bearing of a target ship relative to the own-ship, an intuitive way to guide the RL agent towards COLREGs compliance is to adjust the static obstacle penalty (Equation 23) according to the relative bearing of the dynamic obstacle. The area around a vessel is normally split into three sectors: port, starboard, and stern, as illustrated in Figure 14. Therefore, a tunable parameter ζ_x was added to allow for differentiated weighting of these sectors.

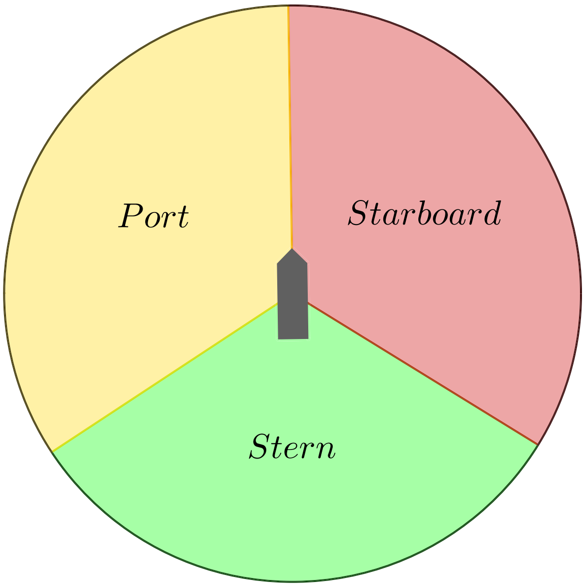


FIGURE 14. Illustration of sectors around the own-ship.

According to the COLREGs, it is desirable that crossings take place on the port side, meaning that the weighting of sensor readings on the starboard side should be higher. However, since it is assumed in this work that the target vessels have restricted maneuverability, sensor readings on the port side and astern must also be sufficiently penalized. Denoting starboard as "st.b.", we thus have that $\gamma_{x,st.b.} < \gamma_{x,port} \leq \gamma_{x,stern}$.

$$\zeta_x(\theta) = \begin{cases} \gamma_{x,st.b.}, & \text{if } \theta \geq 0° \text{ and } \theta < 112.5° \\ \gamma_{x,port}, & \text{if } \theta \geq -112.5° \text{ and } \theta < 0° \\ \gamma_{x,stern}, & \text{if } \theta \geq 112.5° \text{ or } \theta < -112.5° \end{cases} \tag{25}$$

Furthermore, the reward must reflect the variable risk associated with the direction of a target ship's velocity; an approaching target ship gives rise to a much higher risk than a receding one. In addition, the relatively steep function used as weighting term in Equation 23 was exchanged for a flatter function of the form $1/(1 + \exp(x))$, so as to give dynamic obstacles detected around the own-ship sufficient priority. Making adjustments to the static obstacle penalty to adhere to these requirements, the penalty for a single dynamic obstacle was chosen as

$$r_{obst,dyn} = -\underbrace{\frac{1}{1+\exp(\gamma_{\theta,dyn}|\theta|)}}_{\text{Weighting term}} \underbrace{\alpha_x \exp\left((\zeta_v v_y - \zeta_x)x\right)}_{\text{Raw penalty}} \tag{26}$$

where x is the distance to the obstacle, θ is the vessel-relative angle (azimuth angle), and v_y is the velocity component in the direction towards the vessel. The scaling factor ζ_v is given as a function of the angle θ and the velocity v_y, such that the reward efficiently guides the agent towards COLREGs-compliant behavior. It was found that an algorithm with less explicit classification of situations and therefore fewer parameters was in fact harder to tune due to the subsequent high level of dependency between different encounter situations. For instance, since the starboard side is already heavily penalised, lighter weighting of velocity was needed to prevent the agent from reacting too strongly when detecting a target ship on the starboard side. The sign of the velocity component of the target ship towards the own-ship is therefore used to determine whether the target ship is moving towards the own-ship or moving away, which together with the sensor angle θ provides a good basis for determining a reasonable scaling factor for v_y. This scaling factor, ζ_v, is therefore given as

$$\zeta_v(\theta, v_y) = \begin{cases} \begin{cases} \gamma^+_{v,st.b.} & \text{if } v_y \geq 0 \\ \gamma^-_{v,st.b.} & \text{if } v_y < 0 \end{cases} & \begin{matrix} \text{if } \theta > 0° \text{ and} \\ \theta < 112.5° \end{matrix} \\ \begin{cases} \gamma^+_{v,port} & \text{if } v_y \geq 0 \\ \gamma^-_{v,port} & \text{if } v_y < 0 \end{cases} & \begin{matrix} \text{if } \theta > -112.5° \\ \text{and } \theta < 0° \end{matrix} \\ \begin{cases} \gamma^+_{v,stern} & \text{if } v_y \geq 0 \\ \gamma^-_{v,stern} & \text{if } v_y < 0 \end{cases} & \text{otherwise} \end{cases} \tag{27}$$

Finally, as was done for static obstacles, we then compute the dynamic obstacle-avoidance reward according to the weighted average

$$r^{(t)}_{colav,dyn} = -\frac{\displaystyle\sum_{i=1}^{N} \frac{(1-\lambda_i)}{1+\exp(\gamma_{\theta,dyn}|\theta_i|)} \alpha_x \exp\left((\zeta_v v^i_y - \zeta_x)x_i\right)}{\displaystyle\sum_{i=1}^{N} \frac{1}{1+\exp(\gamma_{\theta,dyn}|\theta_i|)}} \tag{28}$$

where λ_i is a parameter regulating the relative importance of path following and collision avoidance in an encounter situation. This parameter function depends on the velocity v^i_y detected, and takes the distance x_i measured by the sensor as input, according to the logistic function

$$\lambda^{(t)}_i = \frac{1}{1 + \exp\left(-\gamma_\lambda(v^i_y)x^{(t)}_i + \alpha_\lambda(v^i_y)\right)} \tag{29}$$

Here, $\alpha_\lambda(v_y)$ and $\gamma_\lambda(v_y)$ are tunable parameters. Two sets of constant values were chosen such that the overall function for λ_i would depend solely on the sign of the speed v_y of the target ship towards the own-ship, giving higher values when $v_y < 0$. In other words, λ_i incorporates the difference in risk between crossing ahead and astern of a target ship, allowing the agent to return to path following quicker in a situation where the target ship is moving away from the own-ship. Formally, we thus have

$$\alpha_\lambda(v_y) = \begin{cases} \alpha^+_\lambda, & \text{if } v_y \geq 0 \\ \alpha^-_\lambda, & \text{if } v_y < 0 \end{cases} \tag{30}$$

and

$$\gamma_\lambda(v_y) = \begin{cases} \gamma^+_\lambda, & \text{if } v_y \geq 0 \\ \gamma^-_\lambda, & \text{if } v_y < 0 \end{cases} \tag{31}$$

4) TOTAL REWARD

Combining the penalties for static and dynamic obstacle avoidance introduced in Equation 24 and 28, the total collision avoidance penalty function becomes

$$r^{(t)}_{colav} = \underbrace{r^{(t)}_{colav,stat}}_{\text{Static component}} + \underbrace{r^{(t)}_{colav,dyn}}_{\text{Dynamic component}} \tag{32}$$

Further, in order to encourage the agent to complete the path within a reasonable time frame, a constant penalty $r_{exists} < 0$ was added. Combining all the elements presented, the expression for the final overall reward function then becomes

$$r^{(t)} = \begin{cases} r_{collision}, & \text{if collision} \\ \lambda^{(t)} r^{(t)}_{path} + r^{(t)}_{colav} + r_{exists}, & \text{otherwise} \end{cases} \tag{33}$$

The relative weighting of the path and collision avoidance rewards regulated by $\lambda^{(t)}$ was, as previously discussed, found necessary to avoid more lenient collision avoidance manoeuvres when encountering a target ship close to the path. Note that each component i in the sum r_{colav} is multiplied by a weighting term $(1 - \lambda_i)$.

Since small values for λ_i indicate a critical presence of another ship (and hence that less priority should be given to the path following objective), the smallest value of λ_i is chosen to regulate r_{path}. Formally, this translates to

$$\lambda^{(t)} = \min_i \lambda^{(t)}_i \tag{34}$$

TABLE 2. Hyperparameters for PPO algorithm.

Parameter	Interpretation	Value
γ	Discount factor	0.999
T	Timesteps per training iteration	1024
N_A	Number of parallel actors	8
K	Training epochs	10^6
η	Learning rate	0.0002
N_{MB}	Number of minibatches	32
λ_{PPO}	Bias vs. variance parameter	0.95
c_1	Value function coefficient	0.5
c_2	Entropy coefficient	0.01
ϵ	Clipping parameter	0.2

TABLE 3. Vessel configuration.

Parameter	Interpretation	Value
U_{max}	Maximum vessel speed	2 m/s
N	Number of sensors	180
S_r	Sensor distance	1.5 km
d	Number of sensor sectors	9
Δ_{LA}	Look-ahead distance	3 km

D. SOFTWARE IMPLEMENTATION

1) TOOLS AND LIBRARIES

Our solution is based on the Python framework OpenAI Gym [75], which has become a de facto standard for DRL interfaces. By implementing our simulation environment as an extension of OpenAI Gym, it is straight-forward to train state-of-the-art, parallelizable RL agents on our scenarios. We use **Stable Baselines** [76], a Python library providing a wide range of well-documented, off-the-shelf RL algorithms, including PPO, for training our agent. The most challenging aspect of the simulation, which is the calculation of the intersection points between the sensor rays and the boundaries of the nearby obstacles, is handled efficiently by the **shapely** Python library [77], which offers an easy-to-use interface to a wide range of geometric analysis-related operations.

2) SIMULATION PARAMETERS

In our setup, both the policy network as well as the value network used in the PPO algorithm's advantage estimation have two hidden layers with 64 units each, and use the *tanh* activation function across the networks. Furthermore, the hyperparameter values presented in Table 2 were used for the PPO algorithm. In terms of the vessel setup, the values in Table 3 were used. Finally, the parameters in Table 4 were used for customizing the reward function. This choice of reward function parameters stems from intuitive reasoning about the desired characteristics of the agent's guidance behavior and how it relates to the parameters. In addition, adjustments were made based on observations made during testing.

TABLE 4. Reward configuration.

Parameter	Interpretation	Value
γ_e	Cross-track error scaling	0.5
α_x	Raw COLAV penalty scaling	75
$\gamma_{\theta,stat}$	Sensor angle scaling for static obst.	10
$\gamma_{\theta,dyn}$	Sensor angle scaling for dyn. obst.	1
γ_x	Static obstacle distance scaling	0.01
$\gamma^+_{v,st.b.}$	Scaling of $v_y \geq 0$, s.b. side	0.004
$\gamma^-_{v,st.b.}$	Scaling of $v_y < 0$, s.b. side	0.05
$\gamma^+_{v,port}$	Scaling of $v_y \geq 0$, port side	0.007
$\gamma^-_{v,port}$	Scaling of $v_y < 0$, port side	0.005
$\gamma^+_{v,stern}$	Scaling of $v_y \geq 0$, astern	0.007
$\gamma^-_{v,stern}$	Scaling of $v_y < 0$, astern	0.005
$\gamma_{x,st.b.}$	Dyn. obst. distance scaling, st.b.	0.007
$\gamma_{x,port}$	Dyn. obst. distance scaling, port	0.009
$\gamma_{x,stern}$	Dyn. obst. distance scaling, stern	0.01
α^+_λ	Translation of λ, $v_y \geq 0$	4
α^-_λ	Translation of λ, $v_y < 0$	2
γ^+_λ	Distance scaling of λ, $v_y \geq 0$	0.003
γ^-_λ	Distance scaling of λ, $v_y < 0$	0.005
r_{coll}	Collision reward	-10000
r_{exists}	Living penalty	-1

E. EVALUATION

To provide a comprehensive basis for evaluating the agent's performance, we test the trained agent in three different test domains.

1) COLREG-COMPLIANCE

First, artificial vessel encounter scenarios, in which COLREG-compliance easily can be categorized as a success or failure in binary terms, are created to quantify the trained agent's performance in a simple and unambiguous manner. Specifically, we simulate head-on and crossing scenarios, and expect the vessel to adhere to the relevant COLREGs rules. The head-on scenario and the first crossing scenario represent the scenarios illustrated in Figure 1, which allows for easy comparison later on.

2) TRAINING ENVIRONMENT PERFORMANCE

Next, we provide a statistical evaluation of the agents based on random samples of the training scenario. More precisely, we evaluate the degree to which the agent is avoiding collisions, as well as the degree to which it adheres to its path following objective, by simulating its behavior in new (i.e. unseen) permutations of the training scenario. As described, the training environment is challenging, with a dense scattering of both static and dynamic obstacles.

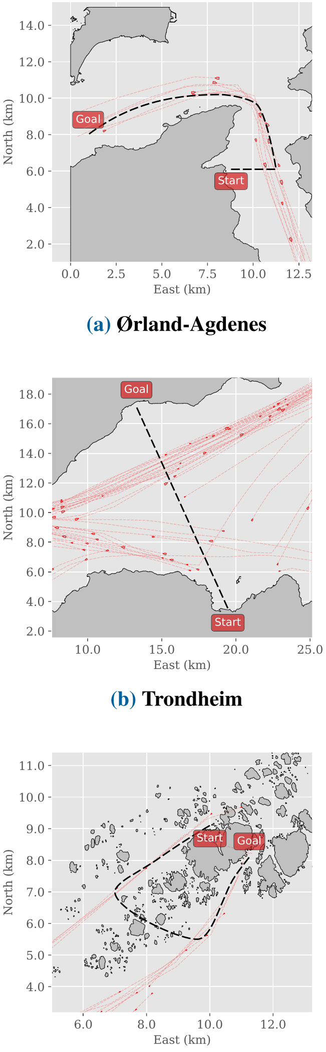
3) REAL-WORLD-BASED EXPERIMENTS

Finally, based on combining high-fidelity terrain data with AIS tracking data from the Trondheim Fjord area, we construct three digital real-world environments in which the vessel's performance can be evaluated.

The dashed black line represents the desired vessel trajectory. Each other vessel is drawn as its initial position with an arrow whose length corresponds to its initial speed. Additionally, each other vessel's trajectory is drawn as a transparent and dotted red line.

a: ØRLAND-AGDENES

This scenario takes place in the heavily trafficked entrance region of the fjord: The region between the municipalities Ørland and Agdenes (Figure 15a). After spawning near the coastline, the vessel must blend into two-way traffic and follow the path until it reaches the opening of the fjord. In particular, the agent will be tested on its ability to handle head-on and overtaking situations.



(a) Ørland-Agdenes

(b) Trondheim

(c) Sorbuoya

FIGURE 15. Maps of the test scenario. The dashed black line represents the desired vessel trajectory. Each other vessel is drawn at its initial position. Also, each other vessel's trajectory is drawn as a transparent and dotted red line.

b: TRONDHEIM

Spawning next to the Trondheim city center, the agent is expected to cross the fjord end and up at the village Vanvikan (Figure 15b). In order to succeed in this scenario, the agent must avoid collisions with the crossing traffic, which is dominated by larger ships.

c: FROAN

Froan, which is located off the Trøndelag coast, is an archipelago encompassing hundreds of small, rocky islands (Figure 15c). For this reason, it offers uniquely challenging terrain. In this scenario, the agent must carefully navigate through a cluster of small islands, before merging into traffic going to and from Sørburøy, the most populated island in the area. The challenging terrain will test the agent's ability to navigate static obstacles, whereas the traffic, comprised of smaller, fast-moving vessels, will lead to challenging head-on situations, especially in the narrow strait in which the goal is located.

V. RESULTS AND DISCUSSIONS

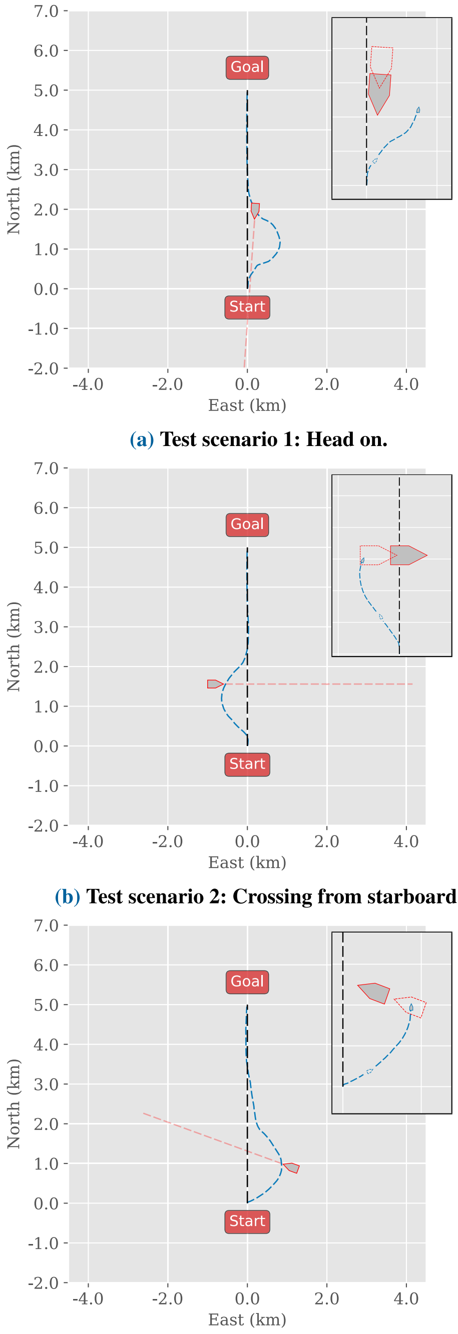
A. COLREGs COMPLIANCE

To provide insight into the agent's fundamental COLREGs compliance, simple encounter scenarios similar to those seen in Figure 1 were constructed. The results of these simulations can be seen in Figure 16. Clearly, the agent adheres to the main COLREGs rules outlined. In Figure 16a, the own-ship adheres to **Rule 14** by altering her course to starboard in a head-on situation. This behaviour was reliably observed when varying the incoming angle of the target ship, denoted θ_t, such that $\theta_t \in [-5°, 5°]$. Further, as seen in Figures 16b and 16c, the agent avoids crossing ahead of a target ship when it can make a reasonable maneuver to cross astern, as described by **Rules 15**, **16**, and **18**. It was noted, however, that there is a "cut-off" when the target ship approaches from an angle $\theta_t > 45°$. In these situations, it chooses to cross ahead, although with a good margin. This makes intuitive sense, as it effectively resolves the conflict without making sharp maneuvers. However, it is important to note the ambiguity of the COLREGs in these cases, stating that the give-way vessel should "keep well clear".

As a means of thoroughly testing COLREG-compliance, the three testing scenarios were simulated for 100 episodes each. For the head-on scenario, the incoming angle of the TS and the path angle α_p of the TS were varied such that $\theta_t \in [-5°, 5°]$ and $\alpha_p \in [-5°, 5°]$. Similarly, the incoming angle and path angle of the TS were varied with $\pm 5°$ relative to the angles utilised for the crossing from starboard and crossing from port scenarios in Figure 16. Results are presented in Table 5.

TABLE 5. Results from repetitive testing of COLREGs with slightly varying scenarios, 100 episodes.

Scenario	Success rate
Head-on	100%
Crossing from starboard	100%
Crossing from port	100%



(a) Test scenario 1: Head on.

(b) Test scenario 2: Crossing from starboard.

(c) Test scenario 2: Crossing from port.

FIGURE 16. Agent trajectories in the test scenarios are drawn as blue dashed lines, and the target ships with trajectories are drawn in red.

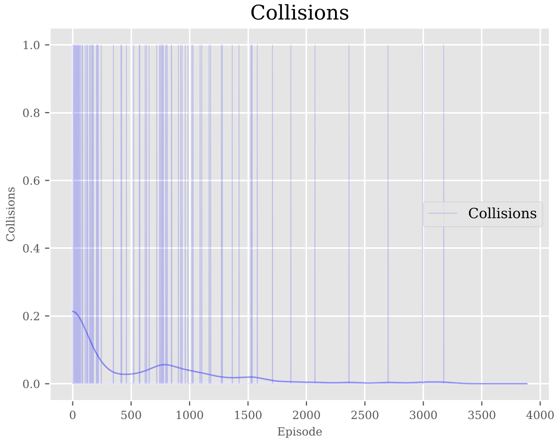


FIGURE 17. Collisions during training. Each vertical line indicates a collision, and the line corresponds to the ratio of number of collisions to number of episodes.

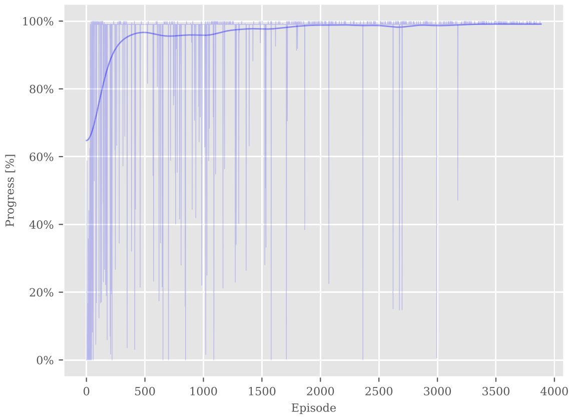


FIGURE 18. Path progress during training.

B. SYNTHETIC TRAINING ENVIRONMENT

To choose a suited agent for testing, statistics from the training process were monitored. When the agent did no longer record collisions for over 500 episodes and the episode rewards stagnated after about 4000 episodes the training was halted. In Figure 17 and 18 the frequency of collisions and progress are presented, showcasing a low collision rate and high path completion rate at the end of training. It is interesting to note that the collision rate quickly decreases, and after only 2000 episodes, the agent is already performing very well. The episode rewards are plotted in Figure 19, showing a sharp increase in reward over the first 500 episodes, before slowly climbing and converging at a steady level. It is worth mentioning that the influence of the randomness present in the training environments is likely insignificant, given that the environment (i.e. the obstacles) is resampled at the beginning of each episode. As the agent experiences more than 1000 independent (but identically distributed) training scenarios during its training period, the effect of the random structure of each episode's obstacle environment is negligible to the final agent's performance.

Next, common collision avoidance maneuvers from the training environment are shown in Figure 20. The snippets presented are representative of the agent's behavior in realistic encounter situations, which indicate that it is COLREG-compliant in the situations where the rules can be accurately discerned. As can be seen from Figure 20a, the agent prefers to cross ahead of the TS at a distance of approximately 600 m to keep it on its port side in the head-on situation. Although not clearly defined in the COLREGs, there is a distance at which crossing ahead of other ships is considered safe,

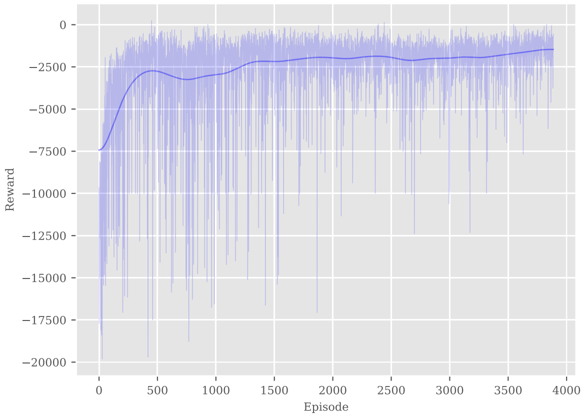
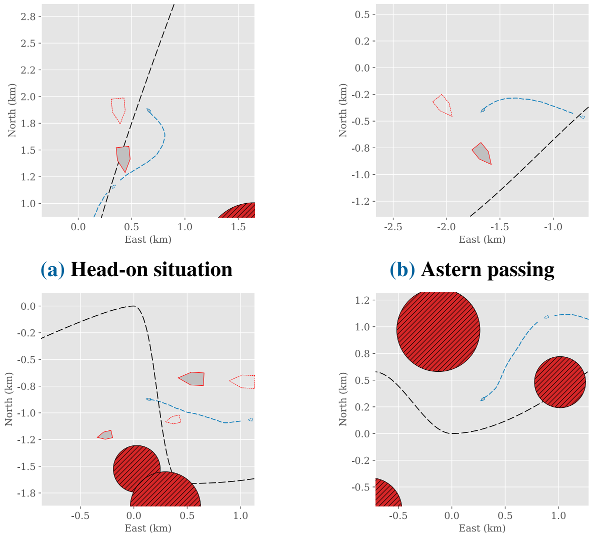


FIGURE 19. Accumulative reward during training.



(a) Head-on situation

(b) Astern passing

(c) Overtaking

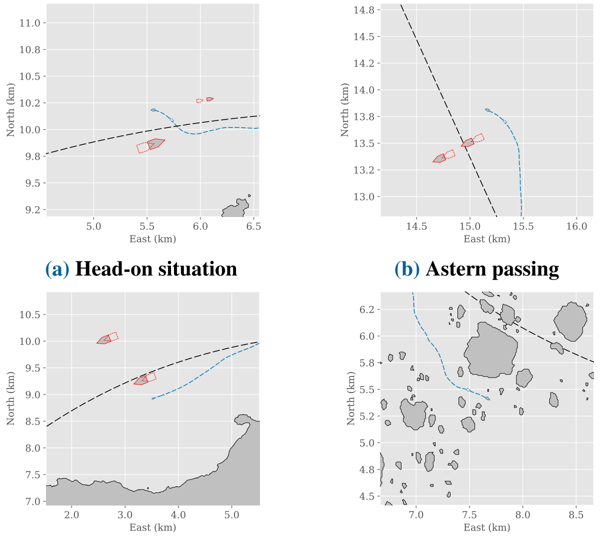
(d) Static COLAV

FIGURE 20. Agent performing common naval collision avoidance maneuvers in the training environment. Agent trajectories are drawn with blue dashed lines, and the target ships are drawn in red.

which has been indirectly communicated through the reward function. In situations such as this one, there is a trade-off between the penalty for keeping a TS on its starboard side and crossing ahead of it. Situations that distinctly require astern crossings, on the other hand, are handled with ease, as shown in Figure 20b.

C. AIS-BASED TEST ENVIRONMENT

Extending the testing to scenarios based on real-world AIS data, it can be seen that the agent behaves in a COLREGs-compliant manner in situations where the COLREGs clearly define an expected behaviour. Some examples of this are presented in Figure 21, where situations similar to those chosen from the training environment (Figure 20) were chosen for comparison. The main difference between the training environment and the AIS-based environment is the shapes and sizes of the static obstacles, which represent land and islands in the AIS-based environment. As seen in Figure 21d, the agent has generalized sufficiently to tackle these scenarios with ease. Furthermore, overall trajectories undertaken by the agent in the Trondheim, Ørland-Agdenes and Froan scenarios can be seen in Figure 23. Although the agent had no issue traversing the complex geography of Froan, it struggled when encountering target ships in restricted waters (see Figure 22). The main explanation of this is likely that the training environment does not reflect these situations properly for the agent to be prepared for them. For instance, in the training environment, the own-ship can always sail around a circular obstacle when encountering a target ship close to such an obstacle. In the Froan scenario, this is not the case, and the own-ship is prone to get lost while attempting to find other ways to the goal. It should therefore be noted that in the scenario presented in Figure 23c, the own-ship did not encounter a target ship after entering the narrow end section of the desired path, but is included to showcase the agent's ability to navigate in restricted waters in the absence of target ships.



(a) Head-on situation

(b) Astern passing

(c) Overtaking

(d) Static COLAV

FIGURE 21. COLREGs-compliant agent performing common naval collision avoidance maneuvers in the AIS-based environment. Agent trajectories are drawn with blue dashed lines, and the target ships are drawn in red. The dotted vessel outlines show their positions 100 time steps prior to the present time.

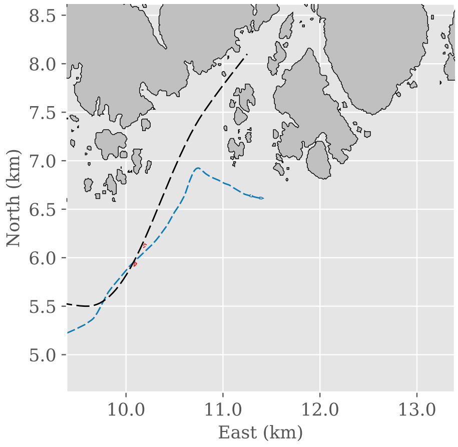
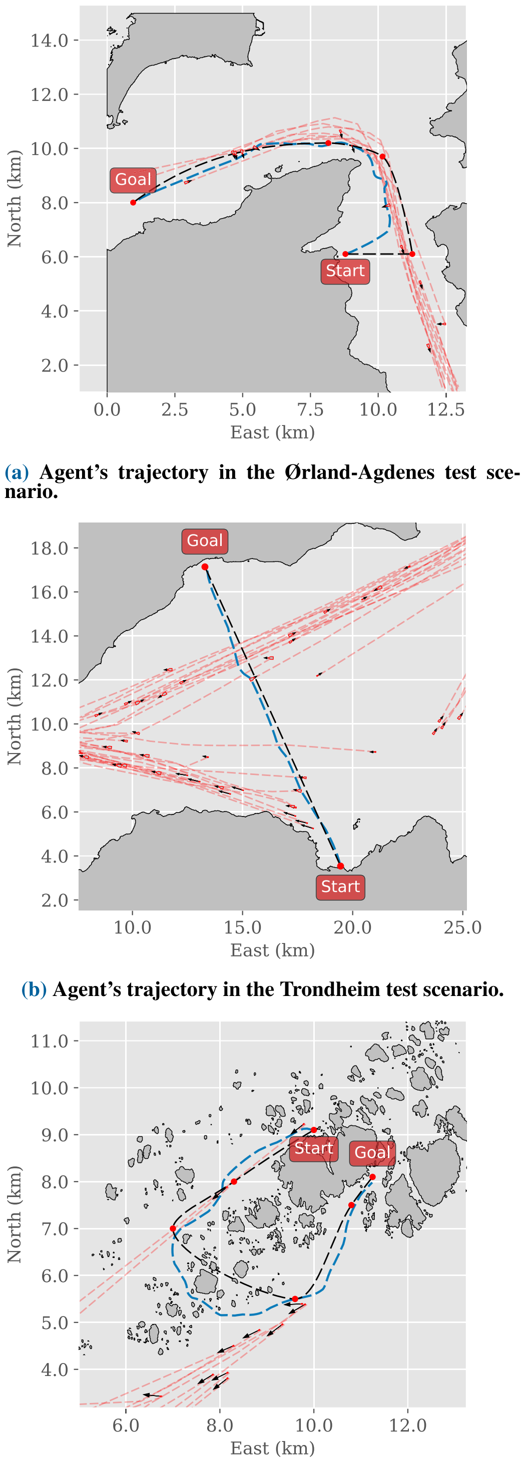


FIGURE 22. Agent (in blue) getting lost attempting to find an alternate route to the goal after encountering a target ship (in red).



(a) Agent's trajectory in the Ørland-Agdenes test scenario.

(b) Agent's trajectory in the Trondheim test scenario.

(c) Agent's trajectory in the Froan test scenario.

FIGURE 23. COLREGs-compliant agent trajectories in the test scenarios drawn as blue dashed lines, and target ships and trajectories are drawn in red.

VI. CONCLUSION AND FUTURE WORK

In this section we present the main conclusions including the strengths and weaknesses of the current work, as well as suggestions for further research.

A. CONCLUSION

The main conclusions from the work are as follows

- A DRL-based autonomous vessel can avoid collisions with other vessels (using COLREGs to the extent possible), while following a pre-determined desired path. With no a priori knowledge of the environment except for the way-points of its desired path, the agent makes reactive control decisions based on rangefinder sensors measuring the distance to nearby obstacles, be it static obstacles such as the shoreline or dynamic obstacles such as other vessels.
- The DRL agent trained in an artificial, simulated environment comprising of a desired path and circular obstacles (stationary and randomly moving) could successfully generalize its learning to an unseen environment consisting of complex shorelines of the Trondheim fjord and real traffic data. Our preliminary results suggest that DRL agents, if trained in a stochastic, generic obstacle environment, can be capable of performing complex guidance tasks. However, despite an acceptable level of success, such an agent will find little acceptability in any safety critical autonomous system because of their black box nature. In the current study we could intuitively explain the behaviour of the agent by relating it to the reward function. At the same time, nothing could be said about the overall stability of the system.
- The design of the reward function, at least for a dual-objective RL environment of this complexity, is best conceptualized as an iterative process; several additional features had to be engineered into an initial reward function outline before a satisfactory performance level was achieved. Of course, the resulting increase in complexity is not desired, but should be considered a necessary compromise for arriving at a competent agent. More often than not, improvements in the agent's performance can be observed immediately from visual inspection after incorporating new reward function features that were identified as likely fixes for correcting undesired patterns in the agent's guidance behavior.
- The approach requires no knowledge of the internal dynamics of the vessel, and allows us to easily adapt the agent behavior by customizing the performance measure. Although the dynamics of the vessel used in this work is reasonably known, the approach can be used with any new class of vessels whose dynamics is too complex or expensive to be derived.

B. FUTURE WORK

As already mentioned, in the current work, we treated DRL as a black box method for attaining the dual objective of path following and collision avoidance, ignoring the effects of wind, waves and current on the vessel. Doing so, we ended up with a complex reward function, which notably did not exploit the fact that other vessels might assist each other for more optimal maneuvers. Most of the shortcomings of this work can be addressed through the following research directions:

- A major point of criticism towards AI methods such as DRL is the lack of explainability and analysability. Neural networks are considered to be black boxes, meaning that their inner workings are difficult to access and evaluate. Furthermore, stability analyses are necessary to assess the robustness of any control system in a safety-critical application. Consequently, it is crucial to move towards less opaque AI. We foresee the use of symbolic regression in combination with convolutional graph networks to express the learning of the agent into mathematical forms that can be subjected to further stability analysis. Some encouraging results on the approach can be found in one of our recent work [78].
- A fundamental assumption here has been calm sea (the absence of environmental disturbances such as wind, ocean currents, and waves). An interesting extension would be to include such disturbances to determine whether the DRL system is able to deal with them. Based on the findings and experiences working with DRL (and the PPO algorithm specifically), it seems likely that a DRL system would efficiently counteract moderate disturbances. In [69], promising path following results are obtained for a DRL-based path following system in the face of disturbances, providing evidence for this.
- In the work and results presented in this work, the agent is interacting with vessels that are blindly following predetermined paths. Needless to say, such a setup is not very realistic, as most encountered target vessels would make attempts to avoid collisions. Therefore, extending the work to a multi-agent environment would be a logical next step in the process of investigating COLREG-compliance in a DRL framework.

Towards the end, we stress that our paper lays the groundwork for further research, which may, given equally positive results, bring significant value to the field of autonomous guidance.

AUTHOR CONTRIBUTIONS

Eivind Meyer developed the software framework facilitating this research and is the lead author. Amalie Heiberg was responsible for incorporating COLREGs compliance and is the author of the corresponding sections and results. Adil Rasheed and Omer San supervised the research and provided guidance throughout the process, as well as proofreading.

REFERENCES

[1] M. Breivik and T. I. Fossen, "Path following for marine surface vessels," in *Proc. Oceans MTS/IEEE Techno-Ocean*, vol. 4, Nov. 2004, pp. 2282–2289.

[2] W. Caharija, K. Y. Pettersen, M. Bibuli, P. Calado, E. Zereik, J. Braga, J. T. Gravdahl, A. J. Sorensen, M. Milovanovic, and G. Bruzzone, "Integral line-of-sight guidance and control of underactuated marine vehicles: Theory, simulations, and experiments," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 5, pp. 1623–1642, Sep. 2016.

[3] S. Moe, W. Caharija, K. Y. Pettersen, and I. Schjølberg, "Path following of underactuated marine underwater vehicles in the presence of unknown ocean currents," in *Proc. ASME 33rd Int. Conf. Ocean, Offshore Arctic Eng.* New York, NY, USA: American Society Mechanical Engineers, 2014, Art. no. V007T05A014.

[4] T. I. Fossen, M. Breivik, and R. Skjetne, "Line-of-sight path following of underactuated marine craft," *IFAC Proc. Volumes*, vol. 36, no. 21, pp. 211–216, Sep. 2003.

[5] J. Guerrero, J. Torres, V. Creuze, and A. Chemori, "Observation-based nonlinear proportional–derivative control for robust trajectory tracking for autonomous underwater vehicles," *IEEE J. Ocean. Eng.*, early access, Jul. 26, 2019, doi: 10.1109/JOE.2019.2924561.

[6] D. Belleter, M. A. Maghenem, C. Paliotta, and K. Y. Pettersen, "Observer based path following for underactuated marine vessels in the presence of ocean currents: A global approach-with proofs," *Automatica*. Amsterdam, The Netherlands: Elsevier, vol. 100, Feb. 2019, pp. 123–134. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2018.11.008, doi: 10.1016/j.automatica.2018.11.008.

[7] M. F. Reis, R. P. Jain, A. P. Aguiar, and J. B. de Sousa, "Robust moving path following control for robotic vehicles: Theory and experiments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3192–3199, Oct. 2019.

[8] C. Paliotta, E. Lefeber, K. Y. Pettersen, J. Pinto, M. Costa, and J. T. de Figueiredo Borges de Sousa, "Trajectory tracking and path following for underactuated marine vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 4, pp. 1423–1437, Jul. 2019.

[9] Y. Liu and R. Bucknall, "Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment," *Ocean Eng.*, vol. 97, pp. 126–144, Mar. 2015.

[10] T. Liu, Z. Dong, H. Du, L. Song, and Y. Mao, "Path following control of the underactuated USV based on the improved line-of-sight guidance algorithm," *Polish Maritime Res.*, vol. 24, no. 1, pp. 3–11, Mar. 2017.

[11] Y. Singh, S. Sharma, R. Sutton, D. Hatton, and A. Khan, "A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents," *Ocean Eng.*, vol. 169, pp. 187–201, Dec. 2018.

[12] S. Campbell, W. Naeem, and G. W. Irwin, "A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres," *Annu. Rev. Control*, vol. 36, no. 2, pp. 267–283, Dec. 2012.

[13] Y. Zhao, W. Li, and P. Shi, "A real-time collision avoidance learning system for unmanned surface vessels," *Neurocomputing*, vol. 182, pp. 255–266, Mar. 2016.

[14] S. Xie, X. Chu, M. Zheng, and C. Liu, "Ship predictive collision avoidance method based on an improved beetle antennae search algorithm," *Ocean Eng.*, vol. 192, Nov. 2019, Art. no. 106542.

[15] B. Shi, Y. Su, C. Wang, L. Wan, and Y. Luo, "Study on intelligent collision avoidance and recovery path planning system for the waterjet-propelled unmanned surface vehicle," *Ocean Eng.*, vol. 182, pp. 489–498, Jun. 2019.

[16] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, Mar. 1986, doi: 10.1177/027836498600500106.

[17] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Trans. Robot. Autom.*, vol. 7, no. 3, pp. 278–288, Jun. 1991.

[18] D. Panagou, "Motion planning and collision avoidance using navigation vector fields," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2014, pp. 2513–2518.

[19] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.

[20] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, May 1999, pp. 341–346.

[21] B.-O.-H. Eriksen, M. Breivik, K. Y. Pettersen, and M. S. Wiig, "A modified dynamic window algorithm for horizontal collision avoidance for AUVs," in *Proc. IEEE Conf. Control Appl. (CCA)*, Sep. 2016, pp. 499–506.

[22] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, Jul. 1998. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijrr/ijrr17.html#FioriniS98

[23] D. K. M. Kufoalor, E. F. Brekke, and T. A. Johansen, "Proactive collision avoidance for ASVs using a dynamic reciprocal velocity obstacles method," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 2402–2409.

[24] Y. Chen, H. Peng, and J. Grizzle, ‘‘Obstacle avoidance for low-speed autonomous vehicles with barrier function,’’ *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 1, pp. 194–206, Jan. 2018.

[25] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, ‘‘A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games,’’ *IEEE Trans. Autom. Control*, vol. 50, no. 7, pp. 947–957, Jul. 2005.

[26] B.-O. H. Eriksen, M. Breivik, E. F. Wilthil, A. L. Flåten, and E. F. Brekke, ‘‘The branching-course MPC algorithm for maritime collision avoidance,’’ *J. Field Robot.*, vol. 36, no. 7, pp. 1222–1249, Aug. 2019. [Online]. Available: http://dx.doi.org/10.1002/rob.21900, doi: 10.1002/rob.21900.

[27] I. B. Hagen, D. K. M. Kufoalor, E. F. Brekke, and T. A. Johansen, ‘‘MPC-based collision avoidance strategy for existing marine vessel guidance systems,’’ in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 7618–7623.

[28] G. Bitar, M. Breivik, and A. M. Lekkas, ‘‘Energy-optimized path planning for autonomous ferries,’’ *IFAC-PapersOnLine*, vol. 51, no. 29, pp. 389–394, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896318321451

[29] Z. Yan, Y. Zhao, S. Hou, H. Zhang, and Y. Zheng, ‘‘Obstacle avoidance for unmanned undersea vehicle in unknown unstructured environment,’’ *Math. Problems Eng.*, vol. 2013, pp. 1–12, Nov. 2013.

[30] Y. Koren and J. Borenstein, ‘‘Potential field methods and their inherent limitations for mobile robot navigation,’’ in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, Apr. 1991, pp. 1398–1404.

[31] S. Moe and K. Y. Pettersen, ‘‘Set-based line-of-sight (LOS) path following with collision avoidance for underactuated unmanned surface vessel,’’ in *Proc. 24th Medit. Conf. Control Autom. (MED)*, Jun. 2016, pp. 402–409.

[32] J. Mínguez and L. Montano, ‘‘Robot navigation in very complex, dense, and cluttered indoor/outdoor environments,’’ *IFAC Proc. Volumes*, vol. 35, no. 1, pp. 397–402, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667015397123

[33] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, ‘‘An introduction to deep reinforcement learning,’’ *Found. Trends Mach. Learn.*, vol. 11, nos. 3–4, pp. 219–354, 2018, doi: 10.1561/2200000071.

[34] J. García and D. Shafie, ‘‘Teaching a humanoid robot to walk faster through safe reinforcement learning,’’ *Eng. Appl. Artif. Intell.*, vol. 88, Feb. 2020, Art. no. 103360. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197619302921

[35] A. Martínez-Tenor, J. A. Fernández-Madrigal, A. Cruz-Martín, and J. González-Jiménez, ‘‘Towards a common implementation of reinforcement learning for multiple robotic tasks,’’ *Expert Syst. Appl.*, vol. 100, pp. 246–259, Jun. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417417307613

[36] K. Bogert and P. Doshi, ‘‘Multi-robot inverse reinforcement learning under occlusion with estimation of state transitions,’’ *Artif. Intell.*, vol. 263, pp. 46–73, Oct. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370218303606

[37] W. Zhang, J. Gai, Z. Zhang, L. Tang, Q. Liao, and Y. Ding, ‘‘Double-DQN based path smoothing and tracking control method for robotic vehicle navigation,’’ *Comput. Electron. Agricult.*, vol. 166, Nov. 2019, Art. no. 104985. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169919302066

[38] A. B. Martinsen and A. M. Lekkas, ‘‘Straight-path following for underactuated marine vessels using deep reinforcement learning,’’ *IFAC-PapersOnLine*, vol. 51, no. 29, pp. 329–334, 2018.

[39] A. B. Martinsen and A. M. Lekkas, ‘‘Curved path following with deep reinforcement learning: Results from three vessel models,’’ in *Proc. OCEANS MTS/IEEE Charleston*, Oct. 2018, pp. 1–8.

[40] Q. Zhang, J. Lin, Q. Sha, B. He, and G. Li, ‘‘Deep interactive reinforcement learning for path following of autonomous underwater vehicle,’’ *IEEE Access*, vol. 8, pp. 24258–24268, 2020. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2020.2970433, doi: 10.1109/access.2020.2970433.

[41] J. Woo, C. Yu, and N. Kim, ‘‘Deep reinforcement learning-based controller for path following of an unmanned surface vehicle,’’ *Ocean Eng.*, vol. 183, pp. 155–166, Jul. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0029801819302203

[42] A. B. Martinsen, A. M. Lekkas, S. Gros, J. A. Glomsrud, and T. A. Pedersen, ‘‘Reinforcement learning-based tracking control of USVs in varying operational conditions,’’ *Frontiers Robot. AI*, vol. 7, p. 32, Mar. 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2020.00032

[43] S. Guo, X. Zhang, Y. Zheng, and Y. Du, ‘‘An autonomous path planning model for unmanned ships based on deep reinforcement learning,’’ *Sensors*, vol. 20, no. 2, p. 426, Jan. 2020, doi: 10.3390/s20020426.

[44] C. Lin, H. Wang, J. Yuan, D. Yu, and C. Li, ‘‘An improved recurrent neural network for unmanned underwater vehicle online obstacle avoidance,’’ *Ocean Eng.*, vol. 189, Oct. 2019, Art. no. 106327. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0029801819304937

[45] L. Zhao and M.-I. Roh, ‘‘COLREGs-compliant multiship collision avoidance based on deep reinforcement learning,’’ *Ocean Eng.*, vol. 191, Nov. 2019, Art. no. 106436. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0029801819305840

[46] C. Lin, H. Wang, J. Yuan, D. Yu, and C. Li, ‘‘Research on UUV obstacle avoiding method based on recurrent neural networks,’’ *Complexity*, vol. 2019, pp. 1–16, Jan. 2019.

[47] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, ‘‘Deep reinforcement learning that matters,’’ in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–27.

[48] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, ‘‘Implementation matters in deep RL: A case study on PPO and TRPO,’’ in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=r1etN1rtPB

[49] European Maritime Safety Agency. (2020). *Marine Casualties and Incidents—Preliminary Annual Overview of Marine Casualties and Incidents 2014-2019*. [Online]. Available: http://www.emsa.europa.eu/accident-investigation-publications/annual-overview/download/6132/2713/23.html

[50] European Maritime Safety Agency. (2019). *Annual Overview of Marine Casualties and Incidents 2019*. [Online]. Available: http://www.emsa.europa.eu/emsa-documents/latest/download/5854/3734/23.html

[51] International Maritime Organization. (1972). *Colregs—International Regulations For Preventing Collisions at Sea*. [Online]. Available: http://www.imo.org/en/About/Conventions/ListOfConventions/Pages/COLREG.aspx

[52] Society of Naval Architects and Marine Engineers (U.S.). Technical and Research Committee. Hydrodynamics Subcommittee, *Nomenclature for Treating the Motion of a Submerged Body Through a Fluid: Report of the American Towing Tank Conference* (Technical and Research Bulletin). Society of Naval Architects and Marine Engineers. 1950. [Online]. Available: https://books.google.no/books?id=VqNFGwAACAAJ

[53] R. Skjetne, Ø. N. Smogeli, and T. I. Fossen, ‘‘A nonlinear ship manoeuvering model: Identification and adaptive control with experiments for a model ship,’’ *Model., Identificat. Control, Norwegian Res. Bull.*, vol. 25, no. 1, pp. 3–27, 2004.

[54] M. E. N. Sorensen, M. Breivik, and B.-O.-H. Eriksen, ‘‘A ship heading and speed control concept inherently satisfying actuator constraints,’’ in *Proc. IEEE Conf. Control Technol. Appl. (CCTA)*, Aug. 2017, pp. 323–330.

[55] R. Skjetne, Ø. Smogeli, and T. I. Fossen, ‘‘Modeling, identification, and adaptive maneuvering of CyberShip II: A complete design with experiments,’’ *IFAC Proc. Volumes*, vol. 37, no. 10, pp. 203–208, Jul. 2004.

[56] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, ‘‘Continuous control with deep reinforcement learning,’’ 2015, *arXiv:1509.02971*. [Online]. Available: https://arxiv.org/abs/1509.02971

[57] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, C. Beattie, S. Petersen, M. Bellemare, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, ‘‘Human-level control through deep reinforcement learning,’’ *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[58] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, ‘‘Asynchronous methods for deep reinforcement learning,’’ 2016, *arXiv:1602.01783*. [Online]. Available: https://arxiv.org/abs/1602.01783

[59] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, ‘‘Proximal policy optimization algorithms,’’ *ArXiv*, vol. abs/1707.06347, 2017.

[60] L. Tai, J. Zhang, M. Liu, J. Boedecker, and W. Burgard, ‘‘A survey of deep network solutions for learning control in robotics: From reinforcement to imitation,’’ *arXiv:1612.07139*, 2016. [Online]. Available: https://arxiv.org/abs/1612.07139

[61] E. Meyer, H. Robinson, A. Rasheed, and O. San, ‘‘Taming an autonomous surface vehicle for path following and collision avoidance using deep reinforcement learning,’’ *IEEE Access*, vol. 8, pp. 41466–41481, 2020.

[62] R. Sutton, D. Mcallester, S. Singh, and Y. Mansour, ‘‘Policy gradient methods for reinforcement learning with function approximation,’’ in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 12, 2000, pp. 1057–1063.

[63] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, vol. 135. Cambridge, MA, USA: MIT Press, 1998.

[64] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *CoRR*, vol. abs/1506.02438, 2016.

[65] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," 2015, *arXiv:1502.05477*. [Online]. Available: http://arxiv.org/abs/1502.05477

[66] Norwegian Mapping Authority. (2019). *Høydedata Og terrengmodeller for Landområdene*. [Online]. Available: https://www.kartverket.no/data/hoydedata-og-terrengmodeller/ and https://www.kartverket.no/data/hoydedata-og-terrengmodeller/

[67] B. van Andel, T. Bieniek, and T. I. Bø. (2012). *Bidirectional UTM-wgs84 converter for Python*. [Online]. Available: https://github.com/Turbo87/utm and https://github.com/Turbo87/utm

[68] F. Codevilla, E. Santana, A. Lopez, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 9328–9337.

[69] A. B. Martinsen, "End-to-end training for path following and control of marine vehicles," Dept. Eng. Cybern., NTNU, Tech. Rep., 2018.

[70] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Hoboken, NJ, USA: Wiley, 2011.

[71] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[72] Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *ArXiv*, vol. abs/1506.00019, 2015.

[73] O. Vinyals *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[74] K. Granstrom and M. Baum, "Extended object tracking: Introduction, overview and applications," *ArXiv*, vol. abs/1604.00970, 2016.

[75] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *ArXiv*, vol.abs/1606.01540, 2016.

[76] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu. (2018). *Stable Baselines*. [Online]. Available: https://github.com/hill-a/stable-baselines

[77] S. Gillies, A. Bierbaum, K. Lautaportti, and O. Tonnhofer. (2007). *Shapely: Manipulation and Analysis of Geometric Objects*. [Online]. Available: https://github.com/Toblerity/Shapely and https://github.com/Toblerity/Shapely

[78] H. Vaddireddy, A. Rasheed, A. E. Staples, and O. San, "Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensor observation data," *Phys. Fluids*, vol. 32, no. 1, Jan. 2020, Art. no. 015113.

EIVIND MEYER is currently pursuing the master's degree in cybernetics and robotics with the Norwegian University of Science and Technology (NTNU), Trondheim. Having specialized in real time systems, his research interest includes adopting state-of-the-art artificial intelligence methods for autonomous vehicle control.

AMALIE HEIBERG is currently pursuing the master's degree in cybernetics and robotics with the Norwegian University of Science and Technology (NTNU), Trondheim. Having specialized in autonomous systems, her research interest includes artificial intelligence for autonomous vehicle control.

ADIL RASHEED received the bachelor's degree in mechanical engineering and the master's degree in thermal and fluids engineering from IIT Bombay and the Ph.D. degree in multiscale modeling of urban climate from the Swiss Federal Institute of Technology Lausanne. He is currently a Professor of big data cybernetics with the Department of Engineering Cybernetics, Norwegian University of Science and Technology, where he is also working to develop novel hybrid methods at the intersection of big data, physics driven modeling, and data driven modeling in the context of real-time automation and control. He also holds a part time Senior Scientist position at the Department of Mathematics and Cybernetics, SINTEF Digital, where he led the Computational Sciences and Engineering Group, between 2012 and 2018.

OMER SAN received the bachelor's degree in aeronautical engineering from Istanbul Technical University, in 2005, the master's degree in aerospace engineering from Old Dominion University, in 2007, and the Ph.D. degree in engineering mechanics from Virginia Tech, in 2012. He worked as a Postdoctoral Researcher with Virginia Tech, from 2012 to 2014, and the University of Notre Dame, Indiana, from 2014 to 2015. He has been an Assistant Professor of mechanical and aerospace engineering with Oklahoma State University, Stillwater, OK, USA, since 2015. His research interest includes the development, analysis, and application of advanced computational methods in science and engineering with a particular emphasis on fluid dynamics across a variety of spatial and temporal scales. He was a recipient of the U.S. Department of Energy 2018 Early Career Research Program Award in Applied Mathematics.