Neurocomputing 568 (2024) 127097

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Enhanced method for reinforcement learning based dynamic obstacle avoidance by assessment of collision risk

Fabian Hart [*], Ostap Okhrin

Chair of Econometrics and Statistics, esp. in the Transport Sector, Technische Universität Dresden, Dresden, 01062, Germany

ARTICLE INFO

Communicated by B. Luo

Keywords:
Dynamic obstacle avoidance
Reinforcement learning
Training environment
Collision risk metric

ABSTRACT

Naturally inspired designs of training environments for reinforcement learning (RL) often suffer from highly skewed encounter probabilities, with a small subset of experiences being encountered frequently, while extreme experiences remain rare. Despite recent algorithmic advancements, research has demonstrated that such environments present significant challenges for reinforcement learning algorithms. In this study, we first demonstrate that traditional designs in training environments for RL-based dynamic obstacle avoidance show extremely unbalanced probabilities for obstacle encounters in a way that high-risk scenarios with multiple threatening obstacles are rare. To address this limitation, we propose a traffic-type-independent training environment that allows us to exert control over the difficulty of obstacle encounter experiences. This allows us to customarily shift obstacle encounter probabilities towards high-risk experiences, which are assessed via two metrics: The number of obstacles involved and an existing collision risk metric.

Our findings reveal that shifting the training focus towards higher-risk experiences, from which the agent learns, significantly improves the final performance of the agent. To validate the generalizability of our approach, we designed and evaluated two realistic use cases: a mobile robot and a maritime ship facing the threat of approaching obstacles. In both applications, we observed consistent results, underscoring the broad applicability of our proposed approach across various application contexts and independent of the agent's dynamics. Furthermore, we introduced Gaussian noise to the sensor signals and incorporated different non-linear obstacle behaviors, which resulted in only marginal performance degradation. This demonstrates the robustness of the trained agent in handling environmental uncertainties.

1. Introduction

Nowadays, autonomous robots have a wide range of applications, such as mobile robots in industry or hospitals, drones, autonomous underwater vehicles, etc. The fundamental task of autonomous real-world navigation is to reach a global goal (global navigation) while maneuvering in a dynamic environment where it is necessary to react to static and dynamic obstacles (local obstacle avoidance) [1]. In global navigation, a complete path is computed based on a known environmental map. Different approaches have been proposed, such as A-star [2], Ant Colony Optimization [3], or Rapid Exploration Random Tree [4]. However, an autonomous robot must be able to react to changes in a dynamic environment. Various methods for obstacle avoidance have been studied, such as the Bug algorithm [5], Artificial Potential Field [6], Virtual Force Field [7], Vector Field Histogram [8] and its extension [9], Dynamic Windows Approach [10], Nearness Diagram [11], Curvature Velocity Method [12], and Elastic Band Concept [13]. In developing autonomous robots, local obstacle avoidance and global path planning are usually combined in a hybrid system [14].

Traditional methods for autonomous navigation face two main challenges. First, there is a contradiction between the accuracy of grid-based map representation and its memory requirements. Second, in dynamic environments, these algorithms require intensive calculations for real-time replanning of the navigation path. Consequently, their reactivity is somewhat limited [15]. To overcome these issues, reinforcement learning (RL) has widely been studied in the field of autonomous robotics. Autonomous navigation based on RL has demonstrated its potential in different domains of traffic, such as for unmanned aerial vehicles, unmanned surface vehicles, and autonomous mobile robots. Comprehensive reviews for all three domains are given by Azar et al. [16] and Hu et al. [17], and Zhu and Zhang [15].

Similar to traditional approaches, an often applied strategy in RL-based autonomous navigation is to separate the overall goal of global navigation and local obstacle avoidance since efficiently reaching a global goal usually stands in contrast to ensuring safety under the threat of obstacles [18,19]. In this work, we focus solely on local obstacle

* Corresponding author.
E-mail address: fabian.hart@tu-dresden.de (F. Hart).

https://doi.org/10.1016/j.neucom.2023.127097
Received 1 December 2022; Received in revised form 16 June 2023; Accepted 29 November 2023
Available online 2 December 2023

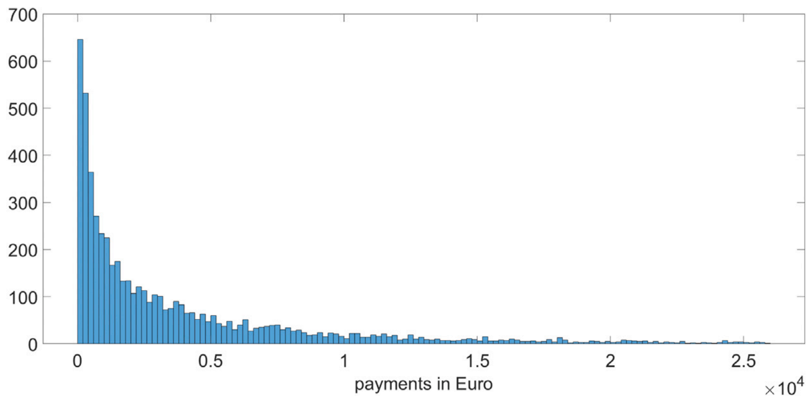

Fig. 1. Insurance payments for bodily injuries in car accidents in the Czech Republic following Zipf's law.

avoidance due to its complexity for safe navigation, especially under consideration of moving obstacles.

In RL, an agent learns by interacting with a training environment. To facilitate effective learning, several RL algorithms utilize an experience replay buffer (ERB) that serves as a storage for all experiences the agent encounters during its interactions with the environment [20]. This collection of experiences is then leveraged during the training process by sampling from the ERB. The purpose of sampling from the ERB is to provide the agent with a diverse range of experiences, enabling it to learn from a broader perspective. By randomly selecting samples from the ERB, the agent can gain insights from past encounters and explore different trajectories taken in the environment. This approach not only enhances the agent's learning efficiency but also improves its ability to generalize its knowledge across various scenarios.

When addressing dynamic obstacle avoidance in RL, a natural approach is to randomly position the agent and obstacles within the training environment [18,19,21–25]. Subsequently, the ERB is filled with random encounters with moving obstacles. However, we emphasize that human learning in the real world does not typically follow a uniform encounter distribution. Instead, real-world encounters often exhibit a power-law distribution, known as Zipf's law [26], which defines the frequency of an encounter, entity, or event as inversely proportional to the rank. An illustrative example of Zipf's law can be observed in Fig. 1, showcasing the distribution of insurance payments for bodily injuries in car accidents within the Czech Republic, indicating that harmless accidents with mild injuries are frequent, whereas accidents resulting in severe injuries are rare. Similar patterns of Zipf's law could be observed in the frequency of objects encountered in early visual learning [27,28] and the formation of relationships in social networks [29,30].

Whereas we as humans have evolved mechanisms like the hippocampus that facilitate effective learning from rare events [31], RL algorithms struggle to learn from such highly skewed distributions, which has recently been investigated by Chan et al. [32]. The authors found that natural experience following Zipf's law poses a substantial challenge for statistical learning systems based on an investigation of different training environments incorporating highly skewed encounter distributions. Even algorithmic advances like modified learning objectives, memory architectures, and self-supervised auxiliary led to only moderate improvements in rare situations. However, performance could substantially be improved by using uniform distributions in training.

Based on these insights, we hypothesize that the traditional design of a training environment for dynamic obstacle avoidance using a random positioning of the agent and obstacles results in obstacle encounter situations that follow Zipf's law. In other words, we assume that the ERB in learning is highly skewed towards experiences of low collision risk or completely lacking a threat through obstacles, whereas high-risk situations are rare. Such high-risk situations can be, for example, characterized as situations where simultaneously multiple obstacles pose a threat to the agent or where just a small portion of steering maneuvers are possible to avoid collisions.

Therefore, instead of a natural and intuitive design of a training environment, we propose a training method where the distribution of experiences in the ERB can be shifted towards high-risk situations that are rare in the real world. Following Chan et al. [32], we assume that through this approach, the performance on effective obstacle avoidance can massively be enhanced. Some approaches use at least simple heuristics to increase the frequency of high-risk experiences in training: While initializing a training episode, one obstacle (among multiple) is being created "to pose a threat" to the safety of the agent [22], however, we miss a clear definition of how this is done. Chun et al. [33] use obstacles that are initialized randomly but in a particular range and with a particular moving direction to increase the threat for the agent. Lan et al. [34] and Xie et al. [35] define movement trends of dynamic obstacles that intersect with that of the agent in a way that collisions are inevitable without an avoidance strategy. Li et al. [36] define one dynamic obstacle that is forming a threat to the planned path of the agent. However, from our point of view, the mentioned approaches are mainly based on the designer's intuition, and a more formal approach is missing.

To overcome this issue, we propose a training approach where the task difficulty of experiences in the ERB, from which the RL agent learns, can be directly manipulated. The task difficulty is formally defined via two metrics: (i) a collision risk metric that aims to quantify the risk of a collision considering the agent's dynamics and (ii) the number of simultaneous obstacles that pose a threat to the agent.

Another problem in most RL-based studies is that agents are evaluated on the same frequency distribution they encounter in training [32]. As a result, situations that occur frequently carry more weight in the evaluation process, while failures in rare situations may have minimal impact on the overall evaluation metrics. For the problem of dynamic obstacle avoidance, this means that the agent is credited to behave well in situations that are frequent, however, which might be easy to solve. On the other hand, high-risk situations, for example, situations where multiple obstacles form a threat at once and which only rarely occur, are less focused since they contribute only little to the final evaluation score. Consequently, achieving satisfactory performance on evaluations that differ from the training distribution presents an ongoing and challenging problem. Previous research has highlighted the difficulties in adapting RL agents to perform well in scenarios with distributional shifts, as indicated by Zhang et al. [37], Hessel et al. [38], Cobbe et al. [39] and Kirk et al. [40]. To address this concern in our study, we evaluate the trained agents in scenarios sampled from difficulty distributions distinct from those encountered during training. Specifically, during evaluation, we employ a uniform difficulty distribution. This distribution ensures that equal importance is given to the entire spectrum of possible obstacle avoidance situations regarding their task difficulty. By using a uniform difficulty distribution, we aim to create a balanced evaluation environment that encompasses various levels of challenge, ranging from easy to difficult.

Summarizing, our contribution is as follows:

- We demonstrate in a simulation study that a conventional and intuitive design of a training environment for dynamic obstacle avoidance leads to situations where the task difficulty approximately follows Zipf's law leading to impaired final performance in high-risk situations.
- We propose a standardized training approach where the task difficulty of experiences in the ERB, from which the RL agent learns, can directly be manipulated. The difficulty is assessed via two metrics: (i) a collision risk metric that aims to quantify the risk of a collision considering the agent's dynamics and (ii) the number of simultaneous obstacles that pose a threat to the agent.
- We perform an analysis that investigates how the task difficulty of experiences in the ERB affects the final policy, and we demonstrate that shifting experiences towards greater difficulty can massively increase the final performance of the obstacle avoidance task.
- We compare our approach with a traditional training environment for dynamic obstacle avoidance based on random initialization of obstacles. The results indicate that our training method significantly outperforms traditional approaches to designing a training environment.
- To mitigate the issue commonly encountered in traditional approaches, where frequently occurring situations (which might be easy to solve) carry more weight in the evaluation process, we evaluate all trained agents in scenarios whose task difficulty is sampled from a uniform distribution. This guarantees an unbiased evaluation of the traditional and proposed approach.
- To demonstrate the generalizability, we transfer our proposed approach to two further application domains: Dynamic obstacle avoidance for a mobile robot and an autonomous ship. The results indicate that, due to its standardization, our approach is not restricted to a specific application but can rather be applied in all domains of obstacle avoidance. Furthermore, we prove the trained agent's robustness by adding Gaussian noise to the sensor signals and using different non-linear obstacle behavior.

This work is structured as follows: In Section 2, we give a formal definition of the problem, followed by a definition of a collision risk metric in Section 3. We further describe the RL methodology in Section 4 and define a training environment for dynamic obstacle avoidance in Section 5 with its training results detailed in Section 7. We investigate the generalizability of the proposed training approach in Section 8 and its robustness in Section 9, and we conclude in Section 10.

2. Problem statement

This study investigates the capability of successful obstacle avoidance for an agent trained with reinforcement learning (RL). For reasons of simplification, we construct a general, two-dimensional use case detached from a specific transportation or robot context, where obstacles follow a linear trajectory. In this work, we specifically focus on high-risk situations where multiple obstacles form a threat at once and where the agent does not have many maneuvers to escape the conflict situation. In such circumstances, the significance of global navigation diminishes, and the sole objective becomes the avoidance of collisions, irrespective of the global goal. Given the specific emphasis on these situations in our work, we do not consider the influence of a destination.

Furthermore, we do not assume an interaction between obstacles in this work. There are several reasons for this decision. First and foremost, our motivation, as stated in the introduction, is to improve upon existing studies on RL-based dynamic obstacle avoidance. We carefully reviewed the relevant literature and found that none of the previous works on RL-based obstacle avoidance we referenced explicitly considered the interaction between obstacles. Furthermore, no interaction makes the task of collision avoidance even harder for the agent since obstacles are passive and do not react to the agent. Since our work focuses specifically on high-risk situations, obstacles that do not react to the agent and are not taking part in resolving the conflict situation are interesting to be investigated.

We define a general agent with two different control inputs: A longitudinal force and a torque. This configuration is similar to many real applications, such as (i) autonomous mobile robots with a longitudinal acceleration and a lateral steering input [41], (ii) autonomous ships with a longitudinal thrust and a rudder angle input which can be translated into a torque [42], or (iii) plane-like drones with a longitudinal force via engines or propellers and a rudder input [43]. We assume a two-dimensional space, resulting in a three-degree-of-freedom (3-DoF) model, however, the proposed space can be straightforwardly generalized to higher dimensions. The state of such a model is defined via the position vector with heading angle $\eta = (x, y, \psi)^\top \in \mathbb{R}^3$ and the velocity vector $\nu = (u, v, r)^\top \in \mathbb{R}^3$ containing velocities in x- and y-direction and the yaw rate r. Based on these state vectors, the dynamics of a 3-DoF model can generally be expressed via two differential equations:

$$\dot{\eta} = g(\nu), \tag{1}$$

$$\dot{\nu} = f(\nu, \tau), \tag{2}$$

with the control vector $\tau = (\tau_u, \tau_v, \tau_r)^\top \in \mathbb{R}^3$ and where $f(\cdot,\cdot)$ and $g(\cdot)$ are deterministic functions specifying dynamics. Through this work, we define three different agent dynamics: At first, we investigate the problem of dynamic obstacle avoidance with a general agent, and, in Section 8, we will define two more realistic agent dynamics.

The initial agent is a simple, point-mass agent where the longitudinal force (τ_u) and the torque (τ_r) are controlled by the RL agent. This translates equation (2) and (1) to:

$$\begin{aligned} \dot{x} &= u\cos(\psi), & \dot{u} &= \tau_u/m, \\ \dot{y} &= u\sin(\psi), & \dot{v} &= 0, \\ \dot{\psi} &= r, & \dot{r} &= \tau_r/I, \end{aligned} \tag{3}$$

with agent mass $m = 15\,\text{kg}$ and moment of inertia $I = 30\ \text{kg m}^2$.

In accordance with the definition of the agent, an obstacle i for $i = 1, \dots, N_{\text{obst}}$ is defined via the position vector η^i_{obst} and the velocity vector ν^i_{obst}. For the sake of simplicity, we just consider obstacles that follow a linear trajectory which leads to $\nu^i_{\text{obst}} = (u^i_{\text{obst}}, 0, 0)^\top$ with u^i_{obst} being constant over time. Furthermore, we define a collision area with radius $R_{\text{coll}} = 3\,\text{m}$ around agent and obstacles, and an overlap of both areas is defined as a collision. Some of the variables are depicted in Fig. 3 later on.

3. Collision risk metric

As motivated in the introduction, we aim for a dynamic obstacle training environment that increases the difficulty of obstacle avoidance experiences by using a collision risk metric. For this study, we adopt the general definition of risk based on probability concepts from [44]:

Definition 3.1. Risk is the probability of an unwanted event.

This definition is pretty natural and lies in $[0; 1]$ because of the properties of probabilities. In this study, an unwanted event refers to the collision of an agent with an obstacle. This leads to the definition:

Definition 3.2. Collision risk is defined by the probability of a future collision with an obstacle.

It is worth mentioning that it is not in the scope of this study to propose the best possible collision risk metric but more how the performance of obstacle avoidance can be increased by using a common and known collision risk metric in training. A list of different metrics to quantify the risk of a collision has already been proposed in the literature. Most related studies focus on maritime traffic, where risk analysis is an essential step in conducting ship avoidance maneuvers. Various studies use the closest point of approach (CPA) as a basis. The CPA defines the point where two ships come closest under the assumption of constant velocities. Time to the closest point of approach (TCPA) and distance to the closest point of approach (DCPA) are common indicators for collision risk in various studies, such as [45] or [46]. Another approach for collision risk analysis is to formulate an area around a ship that should be kept clear to avoid a collision [47].

In this work, we adopt the concept of collision risk from [48], who defined a collision risk metric as *the fraction of all possible maneuvers that lead to a collision*, following Definition 3.2. An advantage of their approach over others is the consideration of the agent's maneuverability to compute the collision risk. This is important when, for example, an agent with good maneuverability might be able to avoid a collision with a close by obstacle while an agent with poor maneuverability might not, assuming the same scenario. Formally, the collision risk according to Huang and Van Gelder [48] from the law of total probabilities is:

$$CR = P(collision) = \sum_{i=1}^{n} P(collision|\tau_i)P(\tau_i),$$

where τ_i is a control command, $P(\tau_i)$ is the probability to choose τ_i, $P(collision|\tau_i)$ is the probability to collide when choosing τ_i, and n is the number of all possible control commands. Following Huang and Van Gelder [48], the following assumption is made:

Assumption 3.1. The probability of a maneuver $P(\tau_i)$ yields a uniform distribution.

This means that all possible maneuvers are equally likely to be chosen without any prior knowledge. For the computation of the collision probability $P(collision|\tau_i)$, the *Generalized Velocity Obstacles* (GVO) approach is used, introduced by Wilkie et al. [49]. GVO is a common method for obstacle avoidance and has widely been applied, such as in [50–52]. Following this approach, the occurrence of future collision can be analytically calculated under the following simplifying assumptions [48]:

Assumption 3.2. Agent and obstacles are disk-shaped.

Assumption 3.3. The obstacles' trajectories are known.

Assumption 3.4. Only those collisions are addressed that will happen before a finite time horizon t_{lim}.

Assumption 3.5. The control command τ_i remains constant within t_{lim}.

From our perspective, especially the last assumption impairs the suitability of this approach to give a good estimate of the current collision risk: It would be more realistic for the agent to be able to react to a threat by choosing another control input within the time horizon t_{lim}. To overcome this issue, we plan to develop our own collision risk metric in the future.

Fig. 2 illustrates the concept of collision risk with the help of three exemplary scenarios. The obstacle with its collision area for different time points R_{coll} is represented in gray. The agent's possible trajectories under constant control input τ and within the time horizon t_{lim} are depicted in blue and red. For reasons of readability, the collision area of the agent is not drawn in. For this illustration, we considered 11 equally spaced values for each control command, longitudinal force τ_u, and torque τ_r, resulting in 121 different trajectories. This configuration is kept unchanged for the entire study. Blue trajectories represent collision-free trajectories, while red ones represent trajectories that lead to a collision with one of the obstacles. The collision risk can now be simply computed by dividing the number of collision trajectories by the number of all possible trajectories. The final CR value is displayed above each scenario.

Fig. 2 only shows linearly moving obstacles. However, we want to emphasize that different kinds of obstacle motion can be considered to compute the CR value as long as the trajectories are known (following Assumption 3.3). But even when the obstacle trajectories are unknown, behavior prediction techniques can be utilized to anticipate trajectories in order to compute possible future collisions with the autonomous vehicle [53].

4. Reinforcement learning methodology

RL is a sub-field of machine learning and has widely been used to develop self-learning agents. In the RL framework, an RL agent learns by interacting with the environment and thus is able to optimize sequential decision-making problems. In this study, the RL agent represents an autonomous robot that has to make decisions about what maneuver to execute in order to avoid dynamic obstacles.

4.1. Basics

RL is grounded on Markov decision processes that are defined by the following components [54]: A state space $\mathcal{S}$ that, in our case, includes state variables of agent and obstacles; an action space $\mathcal{A}$ that includes longitudinal force and torque; an initial state distribution $\mathcal{T}_0 : \mathcal{S} \to [0,1]$ that defines the initial position and velocities of agent and obstacles; a state transition probability distribution $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ that defines the movement of agent and obstacles; a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that should incentives the agent to avoid collisions; and a discount factor $\gamma \in [0,1]$. In an iterative fashion, the agent receives a state information $s_t \in \mathcal{S}$ at each time step t, then it selects a two-dimensional action $a_t \in \mathcal{A}$, gets an instantaneous reward r_{t+1}, and transits to the next state $s_{t+1} \in \mathcal{S}$. Formally, a capital notation is used to describe random variables, e.g., S_t, and a small notation, e.g., s_t or s, to describe realizations of random variables.

The objective of RL is to maximize the expected discounted cumulative reward when starting from state S_0: $J(\pi) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{k+1} \middle| S_0 = s\right]$. Therefore, the RL agent has to learn a policy π that maps from states $s \in \mathcal{S}$ to actions $a \in \mathcal{A}$. In value-based RL, the action-value function $Q^\pi(s,a)$ is used to assess the quality of a state–action pair. Formally, the action-value is defined as the expected return when starting in state s, taking action a, and following policy π afterward: $Q^\pi(s,a) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$. Furthermore, we assume the existence of an optimal action-value function $Q^*(s,a) = \max_\pi Q^\pi(s,a)$. Accordingly, there is a deterministic optimal policy $\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a)$ that we try to find. To approximate $Q^*(s,a)$, the [55] optimality equation is a fundamental practice:

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in S} \mathcal{P}_{sa}^{s'} \max_{a' \in \mathcal{A}} Q^*(s',a').$$

An important algorithm that uses the Bellman Equation is the Q-learning algorithm [56], where a finite number of state–action pairs are stored as a tabular representation. However, this allows just for discrete state and action spaces. An extension of Q-learning for continuous state spaces is the deep Q-network (DQN) algorithm [57] where Q-values are approximated by more complex functions instead of finite tables. Commonly used function approximators are neural networks. To train the function $Q^\omega(s,a)$ with parameter vector ω, the gradient descend algorithm is applied:

$$\omega \leftarrow \omega + \alpha \left\{ y - Q^\omega(s,a) \right\} \nabla_\omega Q^\omega(s,a),$$
$$y = r + \gamma \max_{a' \in \mathcal{A}} Q^{\omega'}(s',a'),$$

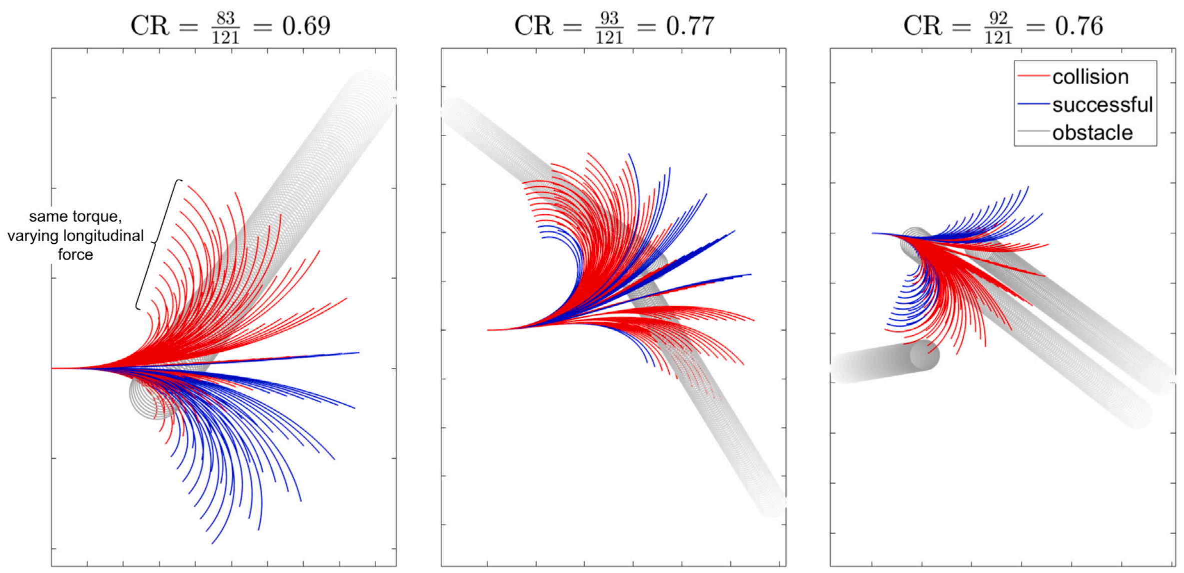

Fig. 2. Agent trajectories for different, constant control inputs (τ). Trajectories leading to a collision with an obstacle trajectory (gray) are marked with red, successful ones with blue color. The resulting initial collision risk CR is displayed in the caption of each scenario.

with learning rate α and $Q^{\omega'}(s,a)$ as target network. The target network represents a copy of the original network, but it is held constant for some fixed number of time steps in order to stabilize the training process. Another extension of DQN is experience replay, which enables the agent to store past experiences that are used for learning in an experience replay buffer (ERB). These experiences are sampled randomly from the ERB for training in a mini-batch scheme. One drawback coming along with the DQN algorithm is the restriction to discrete action spaces. However, our use case requires continuous control commands. We, therefore, use the twin delayed deep deterministic policy gradient (TD3) algorithm, introduced by Fujimoto et al. [58].

4.2. Twin delayed deep deterministic policy gradient (TD3)

TD3 uses several features of the deep deterministic policy gradient (DDPG) algorithm [59]. DDPG is a dual-network algorithm that incorporates a Q-function (Critic) $Q^{\omega}(s,a)$, just as the DQN algorithm, and a deterministic policy-function (Actor) $\mu^{\theta} : \mathcal{S} \rightarrow \mathcal{A}$ with parameter vector θ. Both functions are usually represented as neural networks. Lillicrap et al. [59] adopted experience replay and target networks from DQN but introduced a soft update of the target networks:

$$\begin{aligned} \omega' &= \tau\omega + (1-\tau)\omega', \\ \theta' &= \tau\theta + (1-\tau)\theta', \end{aligned} \tag{4}$$

with hyperparameter τ as the soft target update rate and ω' and θ' as parameters of the target networks. For small values of τ, the target networks update smoothly, which has been shown to improve the training stability. Furthermore, exploration is enhanced by adding random noise to the actions. However, there are still some limitations of the DDPG algorithm. A main issue is the overestimation in Q-values which can lead to a sub-optimal policy [60,61]. Another problem is the sensitivity to the choice of hyperparameters which results in impaired training stability. To overcome these limitations, Fujimoto et al. [58] introduced the TD3 algorithm. The first modification over DDPG is the incorporation of double Q-learning where the minimum of two Q-functions is selected as the target Q-value:

$$y = r + \min_{j=1,2} Q^{\omega'_j}(s,a).$$

Using the minimum of both Q-functions has been found to combat the overestimation of Q-values. A further modification in TD3 is the addition of a truncated normal distribution noise to each action:

$$\tilde{a}_{i+1} = \mu^{\theta'}(s_{i+1}) + \tilde{\epsilon}, \quad \text{with } \tilde{\epsilon} \sim \mathrm{clip}\{\mathcal{N}(0,\tilde{\sigma}), -c, c\}, \tag{5}$$

for some $c > 0$. This regularization method is applied to reduce the variance of the critic update and to smooth the computation of Q-values. To further improve the convergence performance, Fujimoto et al. [58] proposed updating policy and target networks less frequently. A typical strategy is to perform the update every $d = 2$ steps. The complete algorithm is detailed in Algorithm 1.

Table 1
List of TD3 hyperparameters.

Hyperparameter	Value
Discount factor γ	0.99
Batch size N	32
Experience replay buffer size $\lvert D \rvert$	10^5
Learning rate actor α_{actor}	0.0001
Learning rate critic α_{critic}	0.0001
Soft target update rate τ	0.001
Optimizer	Adam
Target noise σ	0.2
Target noise clip c	0.5
Number of hidden layers	2
Neurons per hidden layer	128
policy update frequency d	2

4.3. Architecture and hyperparameters

Actor μ^{θ} and critic function $Q^{\omega}(s,a)$ are feed-forward neural networks with two layers of hidden neurons and 128 neurons in each layer. All layers use ReLU activation functions [62], except the output layer of the actor network where a $\tanh(\cdot)$ activation function is applied. The learning rates α_{actor} and α_{critic} that are used to update actor and critic are set to 0.001. Optimization is performed with Adam [63]. Table 1 displays the complete list of hyperparameters.

5. Training

5.1. State, action and reward

Although we used different agent dynamics throughout this work, we aim to define a generic action space, state space, and reward that

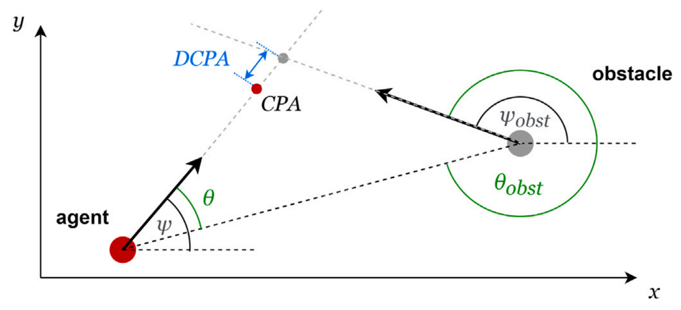

Fig. 3. Technical illustration of the agent and an exemplary obstacle with the closest point of approach (CPA) and the distance to the closest point of approach (DCPA).

is kept constant for all agent dynamics. Since we consider an agent to be controlled via a longitudinal force and a torque, we configured a two-dimensional action $a_t = (a_{u,t}, a_{r,t})^\top \in [-1,1]^2$ that is mapped to the control vector τ_t at each time step t:

$$\tau_t = \tau_{\max} a_t,$$
$$\tau_{\max} = \begin{pmatrix} \tau_{u,\max} & 0 \\ 0 & 0 \\ 0 & \tau_{r,\max} \end{pmatrix},$$

with the maximum longitudinal force $\tau_{u,\max}$ and the maximum torque $\tau_{r,max}$. To make adequate decisions, the agent receives information about itself and surrounding obstacles at each time step t, resulting in the state s_t, where scaling parameters are used to bring all state variables in approximately the same range:

$$s_t = \left(\frac{u_t}{u_{\text{scale}}}, \frac{r_t}{r_{\text{scale}}}, \frac{d_t^i}{d_{\text{scale}}}, \frac{u^i_{\text{obst,t}}}{u_{\text{scale}}}, \frac{\theta_t^i}{\pi}, \frac{\theta^i_{\text{obst,t}}}{\pi}, \frac{DCPA_t^i}{d_{\text{scale}}}, \frac{TCPA_t^i}{t_{\text{scale}}}\right)^\top, \quad (6)$$

with a Euclidean distance to the obstacle i being:

$$d_t^i = \sqrt{(x_t - x^i_{\text{obst,t}})^2 + (y_t - y^i_{\text{obst,t}})^2}.$$

The distance to obstacle i at the closest point of approach $DCPA_t^i$ under control command $\tau = 0$ is defined as:

$$DCPA_t^i = \min_{t'}(d^i_{t'} | \tau = 0, t' > t),$$

and $TCPA_t^i$ is the time until the closest point of approach for obstacle i is reached:

$$TCPA_t^i = \arg\min_{t'}(d^i_{t'} | \tau = 0, t' > t).$$

The angles θ_t^i and $\theta^i_{\text{obst,t}}$, as well as the closest point of approach CPA and the distance to the closest point of approach $DCPA$, are illustrated in Fig. 3. The state variables $DCPA_t^i$ and $TCPA_t^i$ are used to provide the agent with more information about the criticality of an obstacle i at time point t. These additional state variables have been shown to enhance the final performance of obstacle avoidance. The first two variables of the state vector are agent related, while the other six contain information about surroundings. Consequently, the size of the state vector s_t is $N = 2 + 6N_{\text{obst}}$. The obstacles are sorted into the state vector s_t with ascending $TCPA$. All environment parameters with corresponding values can be found in Table 2.

Following RL literature, we aim to keep the reward function as simple as possible [64]. Therefore, we impose a non-zero reward just at the end of an episode t_{end}:

$$r_t = \begin{cases} 0, & \text{for } t < t_{\text{end}}, \\ +1, & \text{for } t = t_{\text{end}} \text{ and episode is successful}, \\ -1, & \text{for collision}, \end{cases} \quad (7)$$

where a collision is defined as an overlap between the collision area R_{coll} of an obstacle and the agent. The episode ends in case of a collision.

Table 2
Parameters of the environment.

Parameter	Value
$\tau_{u,\max}$	1 N
$\tau_{r,\max}$	1 N m
u_{scale}	1.5 m/s
r_{scale}	0.5 s^{-1}
d_{scale}	15 m
t_{scale}	20 s
Δt	0.1 s

Table 3
Variables defining a training scenario with their corresponding range of values.

Scenario variable	Description	Range
u_0	Agent's initial velocity	$[1,2]$ m/s
r_0	Agent's initial angular velocity	$[-0.1, 0.1]$ s^{-1}
$x^i_{\text{obst},0}$	Initial position for obstacle i	$(-\infty, \infty)$
$y^i_{\text{obst},0}$	Initial position for obstacle i	$(-\infty, \infty)$
$\psi^i_{\text{obst},0}$	Initial heading for obstacle i	$[0, 2\pi]$
$u^i_{\text{obst},0}$	Initial velocity for obstacle i	$[0,2]$ m/s
N_{obst}	Number of obstacles	$[1,3]$
CR	Collision risk	$[0,1]$

5.2. CR-based training environment

As motivated in the introduction, we aim to enhance the final performance of obstacle avoidance by controlling the task difficulty distribution of experiences in the ERB. We quantify the difficulty via the combination of two metrics: First, we use the collision risk metric by Huang and Van Gelder [48], described in Section 3. However, as stated above, our method is not limited to a specific collision risk metric. And as a second metric for the difficulty in training, we use the number of obstacles that pose a threat to the agent.

In many studies on RL-based obstacle avoidance, one training episode is quite long and contains encounters with multiple obstacles in a sequential fashion [18,19,21–25,33,65–67]. Additionally, a random initialization of obstacles and agents is a common practice in training, as outlined in the introduction. However, we want to gain more control over the criticality of a collision in training. Therefore, we propose a training environment that consists of multiple short scenarios instead of long episodes with sequential obstacle encounters. Each scenario, which is equal to one training episode, is defined by the agent's initial position $\eta_0 = (0,0,0)^\top$, the agent's initial velocity $\nu = (u_0, 0, r_0)$, the number of obstacles N_{obst}, the initial position $\eta^i_{\text{obst},0} = (x^i_{\text{obst},0}, y^i_{\text{obst},0}, \psi^i_{\text{obst},0})^\top$ and velocity $\nu^i_{\text{obst},0} = (u^i_{\text{obst},0}, 0, 0)^\top$ of all obstacles $i = 1, \dots, N_{\text{obst}}$ and the corresponding collision risk CR (based on Section 3). These variables, defining a scenario, are listed in Table 3 with the corresponding ranges. We limit our study to a maximum of three obstacles, however, the approach can be easily scaled up.

To control for the difficulty in training, we aim to define an environment where the CR of all training scenarios follows a predefined distribution. Therefore, we need to generate scenarios based on a given collision risk. There is no explicit solution for this problem, but the following equations generally describe the dependence between the CR value and the initial agent and obstacle states, exemplary for one obstacle. On one side, the agent trajectory based on a constant action vector $a = (a_u, a_r)^\top$ can be computed via integrating the agent dynamics (3):

$$x_t(a_u, a_r) = \int_0^t \left(u_0 + \frac{\tau_{u,\max} a_u}{m} t'\right) \cos \frac{\tau_{r,\max} a_r}{2I} t'^2 dt',$$
$$y_t(a_u, a_r) = \int_0^t \left(u_0 + \frac{\tau_{u,\max} a_u}{m} t'\right) \sin \frac{\tau_{r,\max} a_r}{2I} t'^2 dt'.$$

The obstacle's trajectory on the other side is defined as:

$$x_{\text{obst},t} = x_{\text{obst},0} + u_{\text{obst},0} \cos \psi_{\text{obst},0} t,$$

$y_{\mathrm{obst},t} = y_{\mathrm{obst},0} + u_{\mathrm{obst},0} \sin \psi_{\mathrm{obst},0} t.$

Having the distance between an agent and an obstacle as

$$D_t(a_u, a_r) = \sqrt{\left\{x_t(a_u, a_r) - x_{\mathrm{obst},t}\right\}^2 + \left\{y_t(a_u, a_r) - y_{\mathrm{obst},t}\right\}^2},$$

the CR can now be computed via:

$$CR = \frac{1}{N_u N_r} \sum_{i=1}^{N_u} \sum_{j=1}^{N_r} \mathbb{1}\left\{\min_t D_t(a_{u,i}, a_{r,j}) < 2R_{\mathrm{coll}}\right\}.$$

where the actions $a_{u,i}$ and $a_{r,j}$ are equally spaced between $[-1,1]$ with N_u and N_r defining the number of actions considered. We chose $N_u = N_r = 11$, resulting in 121 different agent trajectories.

We see that the overall function $CR(u_0, r_0, x_{\mathrm{obst},0}, y_{\mathrm{obst},0}, \psi_{\mathrm{obst},0}, u_{\mathrm{obst},0})$ is heavily non-linear and that an explicit inverse with respect to any argument does not exist. For this reason, a Monte Carlo approach was used to generate a huge pool of 5×10^6 scenarios from which we can draw scenarios with a predefined value of CR. Having this pool, a custom distribution of collision risk can be achieved in training. The distributions we used in this study are defined in Section 7.1.

To generate a single scenario, we first sampled $u_0, r_0, \psi_{\mathrm{obst},0}$, and $u_{\mathrm{obst},0}$ from the defined ranges in Table 3. Next, we computed $x_{\mathrm{obst},0}$ and $y_{\mathrm{obst},0}$ in a way that an obstacle collides with at least one possible agent's trajectory within the time interval $[0.2t_{\lim}, t_{\lim}]$. This condition ensures that all obstacles pose a threat to the agent. Furthermore, the lower bound of the time interval ensures that the agent has enough time to react before a possible collision. Now that we have chosen the initial positions and velocities for the agent and the obstacles, the collision risk is computed by simulating the scenario for a varying control input a_u and a_r.

A training episode ends when either the agent collides with an obstacle or when the distance d_t^i to each obstacle i starts increasing. After an episode, the agent is getting rewarded according to (7). Since the general motion dynamic Eqs. (1) and (2) are for continuous time, the Euler and ballistic methods [68] are used to update the velocity and position vector of the agent and obstacles at discrete time step $t+1$. The Euler method was chosen as the specific numerical integration scheme in our implementation, as it is a straightforward and computationally efficient approach. It is worth noting that alternative methods such as Runge–Kutta could also be utilized for this purpose. Consequently, an update for agent and obstacle i is performed by:

$$v_{t+1} = v_t + \dot{v}_t \Delta t, \tag{8}$$

$$\eta_{t+1} = \eta_t + \frac{\dot{\eta}_t + \dot{\eta}_{t+1}}{2} \Delta t, \tag{9}$$

$$\eta^i_{\mathrm{obst,t+1}} = \eta^i_{\mathrm{obst,t}} + \frac{\dot{\eta}^i_{\mathrm{obst,t}} + \dot{\eta}^i_{\mathrm{obst,t+1}}}{2} \Delta t, \tag{10}$$

with Δt corresponding to the simulation step size.

6. Traditional training approach

This section investigates the weakness of traditional training approaches that use a random positioning of the agent and obstacles in the environment and relatively longer training episodes. As outlined in the introduction, we hypothesize that such intuitive and natural training approach results in experiences that follow a highly skewed distribution towards a lower collision risk and fewer obstacles involved at once, following Zipf's law. To validate this assumption, we implemented the training scheme of Xu et al. [22], which used a quadratic box with fixed borders and five linearly moving obstacles. To ensure the obstacles are navigating within the specified area, we beam an obstacle to the other side of the box when a boundary is reached. The box itself always moves with the agent so that the agent always remains in the center of this box. Since we extended the duration of an episode to 500 time steps for training, we changed the reward structure to:

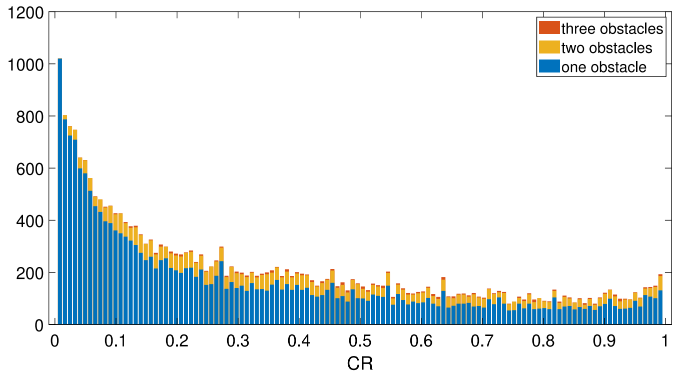

Fig. 4. Distribution of collision risk in the baseline training approach, the colors indicating a different number of obstacles involved.

$$r_t = \begin{cases} -1, & \text{when obstacle and agent are overlapping,} \\ 0.01, & \text{else.} \end{cases} \tag{11}$$

At the beginning of an episode, the obstacles are initialized with random directions and random velocities in the range defined in Table 3.

To investigate the distribution of CR and the number of threatening obstacles, we simulated a sample of 64 000 situations based on random initialization of the agent and obstacles. We found that roughly 36% of experiences had zero collision risk according to the collision risk metric defined in Section 3, and 24% showing a collision risk equal to one, meaning that a collision is unavoidable. Fig. 4 illustrates the collision risk distribution for the remaining experiences, with different colors indicating a different number of involved obstacles. We observe that this training approach suffers from (i) a high coverage of experiences with zero collision risk, (ii) a high coverage of experiences involving just one obstacle (blue bars), and the distribution shifted towards lower collision risk, (iii) a low coverage of more difficult experiences including two or three obstacles (yellow and red bars), and (iv) a high coverage of experiences where the agent has no chance to avoid the obstacle ($CR = 1$). For $CR < 1$, we can interpret these results as an approximation of a Zipf distribution with a high frequency of low-risk experiences (low CR value and only one obstacle involved) and extremely rare high-risk experiences (high CR value and three obstacles involved).

To compare this traditional training environment with our proposed approach, we trained a baseline agent using the defined training environment and reward structure. We ran a hyperparameter search over the reward structure and the edge length of the quadratic box and found the reward according to (11) and an edge length of 35 m to be optimal.

7. Results

7.1. Main study

This study investigates how the distribution of experiences in the ERB affects the final performance of obstacle avoidance. To control the experiences the agent learns from, we adjust the initialization of the training episodes via two factors. First, we varied the distribution of collision risk. Fig. 5 depicts the seven different distributions we used, with darker colors indicating increasing scenario difficulty according to the collision risk. And second, we varied the proportions of scenarios with one, two, and three obstacles. The resulting configurations are depicted in Table 4.

Each collision risk distribution has been combined with each obstacle configuration, resulting in 21 different agents. The training process

Table 4
Three different training configurations with varying proportions of obstacles.

Configuration	1 obstacle	2 obstacles	3 obstacles
4/2/1	14.29%	28.57%	57.14%
1/1/1	33.33%	33.33%	33.33%
1/2/4	57.14%	28.57%	14.29%

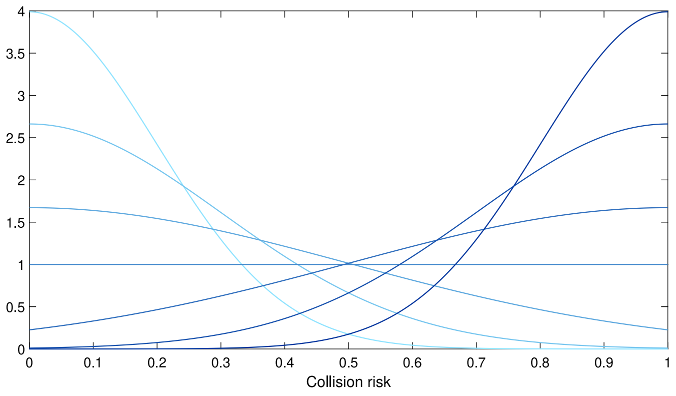

Fig. 5. Seven different collision risk distributions we use in the main study, darker colors indicating distributions towards a higher collision risk.

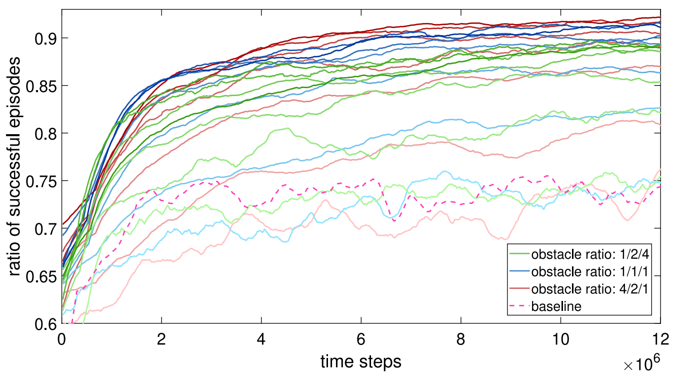

Fig. 6. Training process for 21 agents, trained with all combinations of collision risk distributions (Fig. 5) and obstacle ratios (Table 4) and darker colors indicating distributions towards a higher collision risk.

of each agent is illustrated in Fig. 6. The color coding from Fig. 5 has been adopted, with darker colors indicating a higher proportion of high collision risk scenarios. The x-axis describes the number of steps in training. For every 2×10^4 time step, the same 2000 validation scenarios have been conducted with the current policy. These validation scenarios show equally distributed collision risk and number of obstacles and are identical for all the trained agents. The y-axis describes the ratio of successful validation scenarios. The baseline agent, depicted with a dashed line, was introduced in Section 6. Generally, there are two main trends to observe: The final performance increases (i) with a higher percentage of high collision risk scenarios and (ii) with more obstacles.

Next, we investigated if we obtain even better results when the training solely covers high collision risk scenarios instead of collision risk distributions ranging between 0 and 1 (as in Fig. 5). We, therefore, trained five different agents where the collision risk was uniformly distributed in the intervals: (i) $(0, 0.2]$, (ii) $(0.2, 0.4]$, (iii) $(0.4, 0.6]$, (iv) $(0.6, 0.8]$, (v) $(0.8, 1)$. The number of obstacles was equally distributed for all agents. Fig. 7(a) shows the training progress for all five configurations. Moreover, we trained ten agents in each category, initialized with different seeds. The results after 2×10^7 steps of training are depicted in Fig. 7(b) as boxplots. Apparently, the training with the lowest collision risk (between 0 and 0.2) leads to significantly lower final performance. Among the other four agents, no clear difference is observable. Eventually, slightly better performance can be assumed for the training with CR values between 0.6 and 0.8. But overall, the results of the previous study cannot be reached (compare with the training results of Fig. 6). Interpreting these findings, we assume that the RL agent lacks generalization abilities when just being trained with a small interval of different collision risks. Training only with high collision risks (interval $(0.8, 1)$) seems to be so complex that the agent struggles to learn satisfactorily. We, therefore, recommend defining collision risk distributions that cover all kinds of scenarios between $CR = 0$ and 1, however, strongly skewed towards higher collision risks.

7.2. Scenario-based validation

To further validate our proposed approach, we take a closer look at the trajectories of different trained agents. Fig. 8 illustrates three scenarios, each at four different points in time. The first three time frames show the agent's and obstacles' collision area, while the fourth time frame illustrates the agent's location of collision with an obstacle marked by a cross. We simulated four agents, their colors being identical to the colors in the training plot from Fig. 6. Among them, (i) the agent trained with an obstacle ratio of 1/2/4 (compare with Table 4) and the most extreme collision risk distribution (compare with Fig. 5), labeled as 'red agent', (ii) the agent trained with an obstacle ratio of 1/1/1 and a uniform distribution of collision risk ('blue agent'), (iii) the agent trained with an obstacle ratio of 4/2/1 and the lowest collision risk distribution ('green agent'), and (iv) the baseline agent ('pink agent'). Obstacles are marked with gray.

The first scenario (top row) shows one threatening obstacle and the reaction of the four defined agents. While the red, blue, and green agents try to avoid the obstacle by moving downwards, the pink agent makes the initial decision to move up, resulting in a collision with the obstacle at $t = 7$ s. By making the wrong decision at the start of the episode, we assume that this agent lacks the ability to correctly anticipate the trajectory of the obstacle. Although the green agent decides to move down, it suffers from enough acceleration, leading to a collision at $t = 7.8$ s. Only the red and blue agents are able to simultaneously control both actions, torque and acceleration, in a way that the task is mastered successfully.

The second scenario consists of two threatening obstacles, the upper one forcing the agent to move down, however, the second obstacle blocking this way to escape. The pink agent seems to only consider the closer obstacle (lower one) and decides to move up, however, resulting in a collision at $t = 8.8$ s with the upper obstacle. The green agent also seems unable to consider both obstacles simultaneously and decides to move down, resulting in a collision shortly after $t = 6.4$ s. In contrast, the blue and red agents are able to correctly estimate the trajectory of both obstacles and to react accordingly. The blue agent successfully finds his way through the obstacles, yet not far away from a collision at $t = 6.4$ s. The red agent masters the scenario as well, with enough safety distance to the lower obstacle, however, also closely missing a collision at $t = 10.6$ s.

In a last scenario (bottom row), the agent is facing three threatening obstacles. The green and pink agents again seem to consider just the closest and most threatening obstacle, and both react by moving downwards, the green one colliding with an obstacle at $t = 9.6$ s. On the contrary, red and blue agents seem to be able to include all three obstacles in the decision of maneuver planning, resulting in a more complex, "S"-shaped trajectory. However, among them, the red agent is the only one without collision.

Looking at all scenarios together, the red agent seems to know best how to predict future obstacle trajectories and how to avoid these threats under consideration of its own maneuverability. Especially in

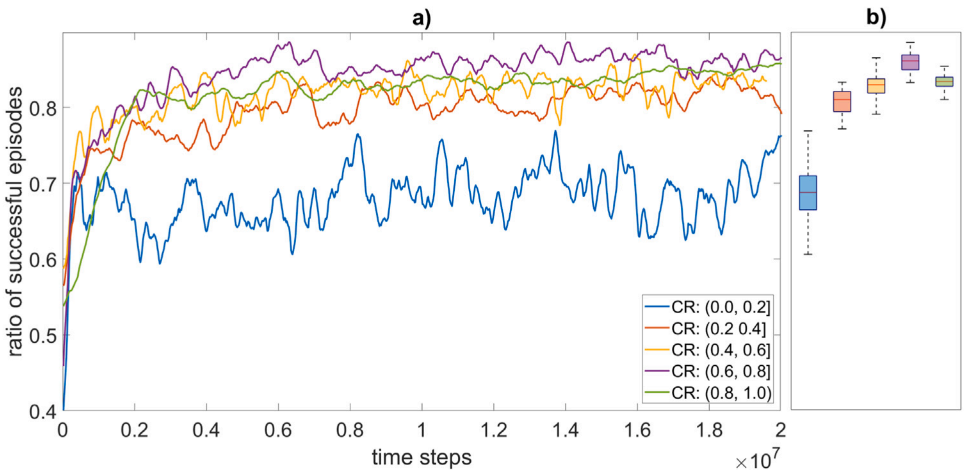

Fig. 7. (a) Training process for five different agents, the legend indicating the individual collision risk distribution in training. (b) Distribution of performance after 2×10^7 steps of training for ten agents in each category, initialized with different seeds.

the last scenario, this agent shows its capability to do complex collision avoidance maneuvers under consideration of multiple obstacles at once.

8. Generalizability study

After demonstrating the strength of the proposed training approach for specific agent dynamics, we will investigate how generally this approach can be applied to different domains. We, therefore, adopted the proposed training method in two common dynamic obstacle avoidance settings: First, in RL-based dynamic obstacle avoidance for mobile robots (Section 8.1), which, for example, has been studied in [21,65, 69], or [24]; and second, in RL-based maritime ship-collision avoidance (Section 8.2), which is a common field of scientific research as well [33, 35,70,71]. Therefore, we defined two further agent dynamics that both built upon the general definition of a 3-DoF model (Eqs. (1) and (2)) and can be interpreted as an extension of the model that we used in the previous study. The training results for both use cases are depicted in Section 8.3. Another part of the generalizability study is to analyze the robustness of the proposed training approach when experiencing sensor signal noise, detailed in Section 9.1.

8.1. Mobile robot collision avoidance

We use a two-dimensional model of a four-wheeled robot with control of the acceleration force τ_u and the steering angle of the front axis δ_r. For a schematic illustration, see Fig. 9. The resulting kinematic model can then be written as [72]:

$$\begin{aligned}
\dot{x} &= u\cos(\psi + \beta),\\
\dot{y} &= u\sin(\psi + \beta),\\
\dot{\psi} &= \frac{u}{l_r}\sin(\beta),\\
\dot{u} &= \frac{\tau_u}{m},\\
\dot{v} &= 0,
\end{aligned}$$

where β is the slip angle at the center of gravity G:

$$\beta = \arctan\left\{\tan(\delta_r)\frac{l_r}{L}\right\}.$$

As in the main study, we configured two-dimensional action $a_t \in [-1,1]^2$ that is linearly mapped to the acceleration force and the steering angle by using a maximum acceleration force $\tau_{u,\max}$ and a maximum steering angle $\delta_{r,\max}$. We further adopted the state space (6) and the update scheme (8)–(10) from the main study. To use the same training approach, the scaling parameters, as well as the ranges of scenario variables, had to be adjusted. All robot and environment-related parameters and variable ranges are depicted in Table A.1.

8.2. Maritime ship collision avoidance

Next to collision avoidance for a mobile robot, we aim to prove our proposed training approach for obstacle avoidance in a common maritime traffic application. Therefore, we consider the widely used KVLCC2 tanker with a length $L = 320\,\mathrm{m}$ and adopt the 3-DoF ship maneuvering model from [73]. Again, we use $\eta = (x,y,\psi)^\top \in \mathbb{R}^3$ and $\nu = (u,v,r)^\top \in \mathbb{R}^3$ to describe position and velocity vector. The ship dynamics follows the general 3-DoF model (2) and (1) as well, however, the resulting differential equations cannot be expressed explicitly:

$$\begin{aligned}
&(m+m_x)\,\dot{u} - (m+m_y)\,vr - x_G m r^2 = X,\\
&(m+m_y)\,\dot{v} + (m+m_x)\,ur + x_G m\dot{r} = Y,\\
&(I_{zG} + x_G^2 m + J_z)\,\dot{r} + x_G m(\dot{v} + ur) = N_m,
\end{aligned}$$

with ship mass m, added masses m_x and m_y in x and y direction, respectively, the longitudinal coordinate of the center of gravity x_G, the moment of inertia I_{zG} of the ship around the center of gravity, and the added moment of inertia J_z. The surge force X, lateral force Y, and the yaw moment N_m around midship can be decomposed in components acting on hull (H), rudder (R), and propeller (P):

$$\begin{aligned}
X &= X_H + X_R + X_P,\\
Y &= Y_H + Y_R,\\
N_m &= N_H + N_R.
\end{aligned}$$

The ship motion can be controlled by rudder angle δ_r and propeller revolutions n_P which directly impact surge force X lateral force Y, and the yaw moment N_m around midship. We refer to Yasukawa and Yoshimura [73] for a detailed explanation of each force component and the derivation of parameters describing the KVLCC2 tanker.

Again, we configured two-dimensional action $a_t \in [-1,1]^2$ that is linearly mapped to propeller revolutions $n_\mathrm{P} \in [0, n_{\mathrm{P},\max}]$ and the rudder angle $\delta_r \in [-\delta_{r,\max}, \delta_{r,\max}]$. Since, in contrast to the previously used models, the ship model is not restricted to lateral velocities $v = 0$, the state space from (6) is extended by the factor v_t/u_scale. All environment-related parameters with corresponding values and ranges are depicted in Table A.1.

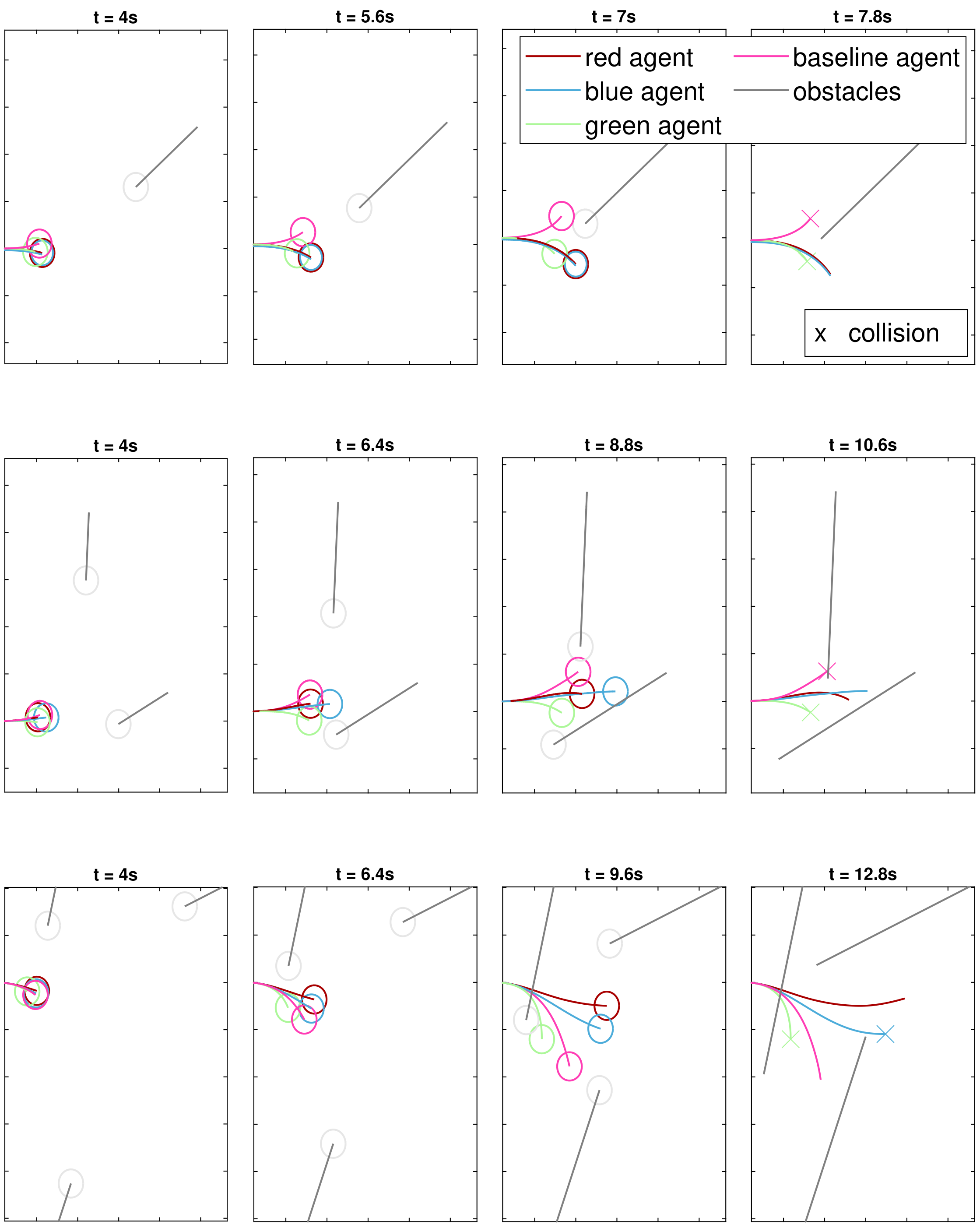

Fig. 8. Validation of our proposed approach in three scenarios with one (top row), two (middle row), and three (bottom row) obstacles by comparison of trajectories at different points in time.

8.3. Training results

We conducted two further trainings based on the previously defined agent dynamics. To evaluate the generalizability of our proposed approach, we used the same obstacle and collision risk distribution as in the main study (see Fig. 5 and Table 4). Figs. 10 and 11 depict the training process based on the robot and the ship dynamics, respectively. We used the same color coding as in the main study for obstacle and collision risk distributions. The general trend we observed before can be confirmed: Obstacles ratios towards more obstacles and collision risk distributions towards higher risks lead to a higher ratio of successful validation episodes. These findings show that instead of being limited to a specific problem, our training approach can be applied to many different domains of obstacle avoidance.

9. Robustness study

9.1. Sensor noise

In another experiment, we aim to analyze the robustness of the training approach against noisy sensor signals. Therefore, Gaussian noise is multiplied to all state variables (6) each time step. For an exemplary state variable s^*, this results in:

$$s^*_N = s^*\epsilon, \quad \text{with } \epsilon \sim \mathcal{N}(1, \sigma_N),$$

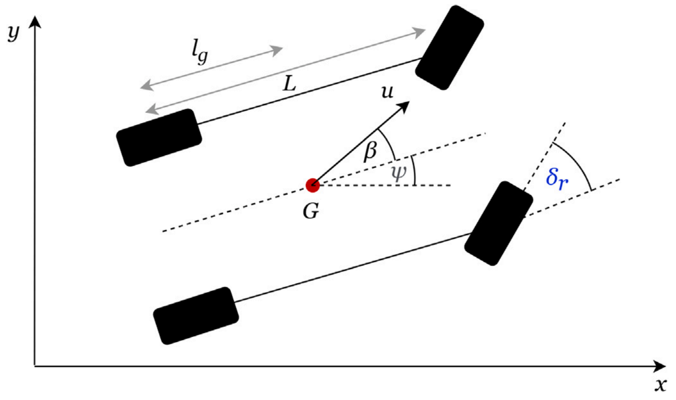

Fig. 9. Technical drawing of a four-wheeled mobile robot with steering of the front axis.

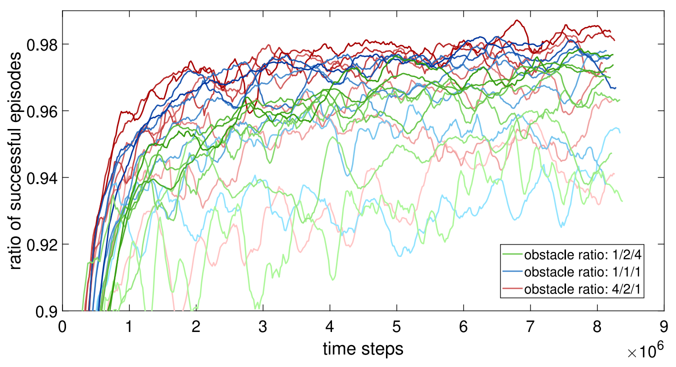

Fig. 10. Training process of the mobile robot in the proposed training environment, darker colors indicating a higher collision risk in training.

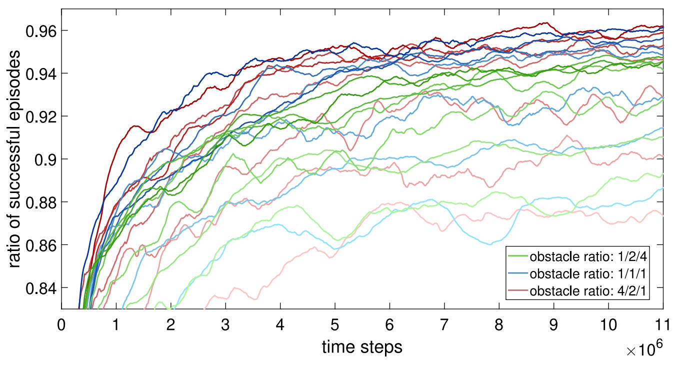

Fig. 11. Training process of the KVLCC2 tanker in the proposed training environment.

with standard deviation σ_N. Fig. 12 depicts the ratio of successful validation episodes for varying noise intensity. The results show that even for quite noisy signals with a standard deviation of $\sigma_N = 0.15$, the trained agent is still able to master a majority of the validation scenarios successfully. This demonstrates the robustness of the proposed training approach in situations where sensory information like positions and velocities of obstacles cannot accurately be measured.

9.2. Non-linear obstacle motion

To demonstrate the robustness of the proposed approach, we validate the agent for two types of non-linear trajectory behavior. First, we aim to design an obstacle motion behavior that is not as predictable as the linear motion used before, incorporating randomness using

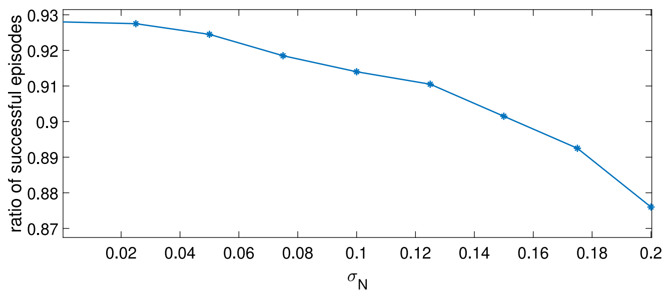

Fig. 12. Robustness of the trained agent against Gaussian sensor noise with standard deviation σ_N.

stochastic processes. Therefore, we add to the existing obstacle motion, defined by the position vector $v^i_{\text{obst,t}} = (x^i_{\text{obst,t}}, y^i_{\text{obst,t}})^\top$ over time t, a second component $v^i_{\text{stoch,t}}$ based on a stochastic two-dimensional AR(1) process [74], neglecting the third dimension ($\psi^i_{\text{obst,t}}$) for reasons of simplification:

$$v^i_{\text{stoch,t+1}} = \phi_{\text{AR1}} v^i_{\text{stoch,t}} + u_t, \quad \text{where} \quad u_t \sim \mathcal{N}_2\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}\sigma^2_{\text{AR1}} & 0\\ 0 & \sigma^2_{\text{AR1}}\end{pmatrix}\right), \tag{12}$$

with initial value $v^i_{\text{stoch,t}} = (0,0)^\top$, auto-regressive parameter $\phi_{\text{AR1}} = 1$, and variance $\sigma^2_{\text{AR1}} = 0.0532\ \text{m}^2$. For exemplary trajectory of the obstacle we refer to the left panel of Fig. 13.

Second, we validated the proposed approach for curved obstacle trajectories, which are defined as:

$$v^i_{\text{obst,t}} = \begin{pmatrix} x^i_{\text{R}} + R^i_{\text{obst}} \cos \psi^i_{\text{obst,t}} \\ y^i_{\text{R}} + R^i_{\text{obst}} \sin \psi^i_{\text{obst,t}} \\ \psi^i_{\text{obst,t}} \end{pmatrix}, \tag{13}$$

with the center point $(x^i_{\text{R}}, y^i_{\text{R}})^\top$ and the radius R^i_{obst}. The minimum radius was set to 40 m. The exemplary trajectory of the obstacle for this setup is depicted on the right panel of Fig. 13.

Similar to Sections 7 and 8, we validate our approach in scenarios with a uniform distribution of collision risk. Therefore, we again created a pool of scenarios for both defined non-linear obstacle motion models from which we can draw scenarios with a predefined value of collision risk, as described in Section 5.2. It is again to underline that the proposed collision risk metric is also suitable for non-linear trajectories. Fig. 13 illustrates two obstacle avoidance scenarios with obstacle motion according to (12) and (13) and possible agent maneuvers leading to a successful avoidance or a collision. For the validation of the trained agents, we sampled, similar to previous sections, 2000 scenarios that show equally distributed collision risk and number of obstacles. Fig. 14 shows the ratio of successful validation scenarios with linear, stochastic, and curved obstacle trajectories. Although there is a slight decrease in performance for stochastic and curved obstacle trajectories compared to linearly moving obstacles, our approach generally demonstrates to be robust against different types of obstacle behavior.

10. Conclusion

A naturally inspired and intuitive approach for designing a training environment for dynamic obstacle avoidance is to randomly position the agent and obstacles within the environment. Following this approach, the ERB as a storage for experiences to learn from is filled with random obstacle encounters. However, research has shown that learning in the real world does not follow a uniform encounter situation but more a highly skewed distribution, known as Zipf's law. Whereas we as humans have evolved sophisticated mechanisms to learn from

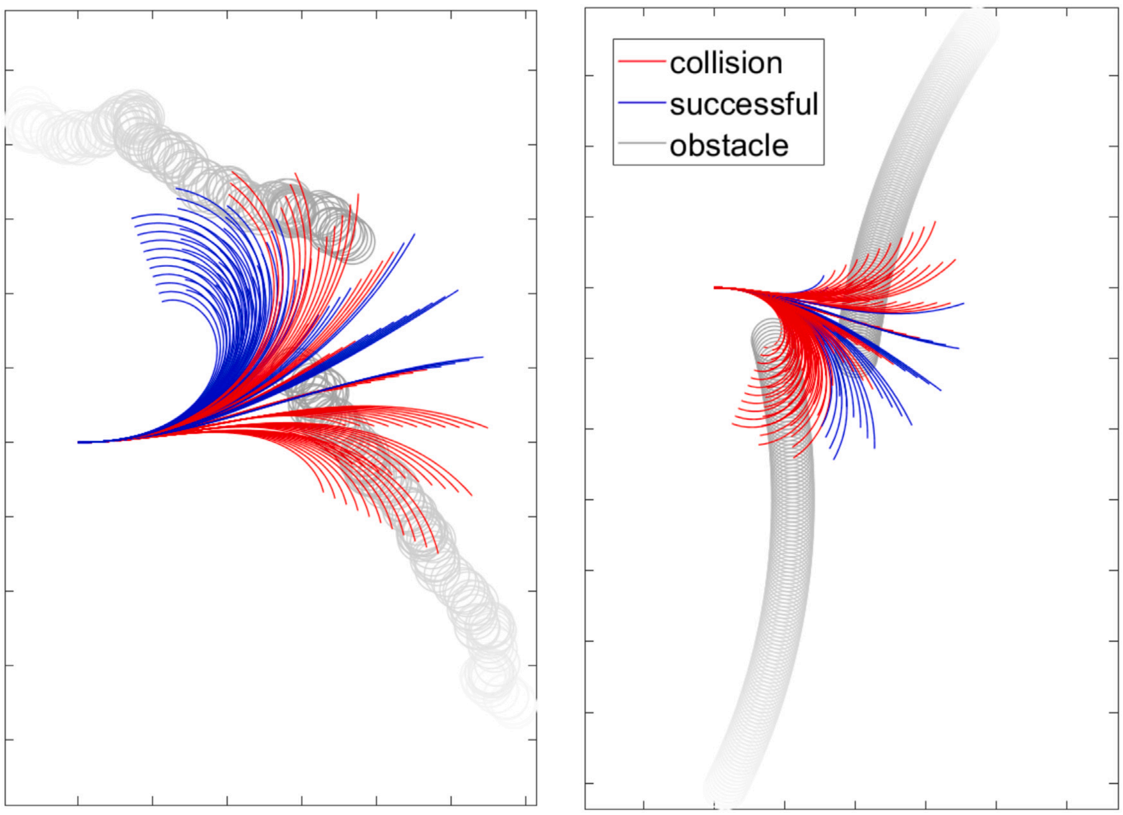

Fig. 13. Obstacle motion based on stochastic processes (12) and curved obstacle motion (13) with possible agent maneuvers leading to a successful avoidance or a collision.

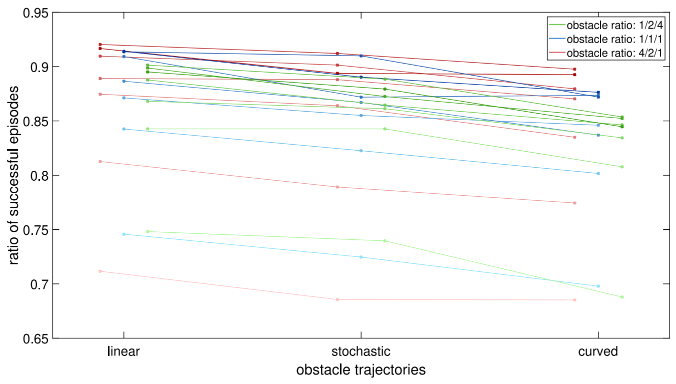

Fig. 14. Success rate of 21 agents, trained with all combinations of collision risk distributions (Fig. 5) and obstacle ratios (Table 4) for different types of obstacle motion models.

rare events, research has demonstrated that RL algorithms lack such mechanisms.

Building upon these insights, our study investigated the distribution of obstacle encounters in traditional RL-based dynamic obstacle avoidance training environments and confirmed that they indeed align with Zipf's law. Furthermore, we observed that learning from this highly skewed distribution resulted in poor final performance in obstacle avoidance tasks. To address this limitation, we developed a novel training approach that allows direct control over the distribution of experiences in the ERB.

Our approach involves manipulating two key factors: the number of threatening obstacles and an existing collision risk metric. We demonstrated that shifting the difficulty of training situations, assessed by the number of obstacles involved and a collision risk metric, towards greater difficulty resulted in higher performance on obstacle avoidance. We defined different ratios of obstacles and multiple distributions of collision risk to define scenarios ranging from simple obstacle avoidance with one obstacle and a low probability of collision up to high collision risk scenarios with three threatening obstacles. The training of an initial agent with simple dynamics indicates that training scenarios covering more obstacles and higher initial collision probabilities lead to better final performance of obstacle avoidance.

Another feature of our proposed method is its suitability for different application domains of obstacle avoidance, regardless of the agent's maneuverability. This advantage results from the used collision risk metric that explicitly considers the agent's dynamics. We proved the generalizability in two common applications of dynamic obstacle avoidance: A mobile robot and a maritime ship under the threat of approaching obstacles that can be considered as other traffic participants. In both applications, we can confirm the advantage of using training scenarios shifted towards greater difficulty. To further prove the robustness of the trained agent, we added Gaussian noise to the sensor signal, resulting in just a marginal degradation of performance.

However, there are still several issues that need to be addressed. Instead of using the number of threatening obstacles and the adopted collision risk metric, one should think about developing more sophisticated methods to assess the collision probability in a training scenario. A disadvantage coming along with the collision risk metric from [48] is the assumption of a constant control vector to compute the collision probability.

Despite those drawbacks and generalizing our proposed approach with regard to other obstacle avoidance applications, we believe that employing scenario-based training with increased coverage of high collision risk scenarios, whatever collision risk metric might be used, generally results in a better final performance. Our findings highlight the importance of considering the distribution of experiences and difficulty levels in training scenarios, providing valuable insights for enhancing RL-based obstacle avoidance.

CRediT authorship contribution statement

Fabian Hart: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Ostap Okhrin:** Methodology, Validation, Writing – review & editing, Project administration, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Fabian Hart reports financial support was provided by Bundesanstalt für Wasserbau.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was partially funded by BAW - Bundesanstalt für Wasserbau (Mikrosimulation des Schiffsverkehrs auf dem Niederrhein).

Appendix

Algorithm 1: TD3 algorithm following Fujimoto et al. [58].

```
Randomly initialize critics $Q^{\omega_1}, Q^{\omega_2}$ and actor $\mu^{\theta}$
Initialize target critics $Q^{\omega_1'}, Q^{\omega_2'}$ and target actor $\mu^{\theta'}$ with
  $\omega_1' \leftarrow \omega_1$, $\omega_2' \leftarrow \omega_2$, $\theta' \leftarrow \theta$
Initialize experience replay buffer $\mathcal{D}$
Receive initial state $s_1$ from environment
for $t = 1,T$ do
    Acting
    Select action with exploration noise: $a_t = \mu^{\theta}(s_t) + \epsilon$,
      $\epsilon \sim \mathcal{N}(0, \sigma)$
    Execute $a_t$, receive reward $r_{t+1}$, new state $s_{t+1}$, and done
      flag $d_t$
    Store transition $(s_t, a_t, r_{t+1}, s_{t+1}, d_t)$ to $\mathcal{D}$

    Learning
    Sample random mini-batch of transitions
      $(s_i, a_i, r_{i+1}, s_{i+1}, d_i)_{i=1}^{N}$ from $\mathcal{D}$
    Calculate targets:
      $\tilde{a}_{i+1} = \mu^{\theta'}(s_{i+1}) + \tilde{\epsilon}, \quad \tilde{\epsilon} \sim \text{clip}\{\mathcal{N}(0, \tilde{\sigma}), -c, c\}$,
      $y_i = r_{i+1} + \gamma(1 - d_i) \min_{j=1,2} Q^{\omega_j'}(s_{i+1}, \tilde{a}_{i+1})$.
    Update critics: $\omega_j \leftarrow \min_{\omega_j} N^{-1} \sum_{i=1}^{N} \left\{y_i - Q^{\omega_j}(s_i, a_i)\right\}^2$
    if $t$ mod $d$ then
        Update actor: $\theta \leftarrow \max_{\theta} N^{-1} \sum_{i=1}^{N} Q^{\omega_1}\left\{s_i, \mu^{\theta}(s_i)\right\}$
        Update target networks via (4)
    end

    End of episode handling
    if $d_t$ then
        Reset environment to an initial state $s_{t+1}$
    end
end
```

Table A.1
Mobile robot and maritime ship environment parameters and variable ranges.

Mobile robot		Maritime ship	
m	5 kg	–	
l_r	1 m	–	
L	2 m	–	
R_{coll}	6 m	R_{coll}	300 m
$\tau_{u,\max}$	1 N	$n_{\text{P},\max}$	2 s^{-1}
$\delta_{r,\max}$	5°	$\delta_{r,\max}$	20°
u_{scale}	6 m/s	u_{scale}	8 m/s
r_{scale}	0.2 s^{-1}	r_{scale}	0.005 s^{-1}
d_{scale}	40 m	d_{scale}	2000 m
t_{scale}	20 s	t_{scale}	1400 s
Δt	0.1 s	Δt	7 s
$u^i_{\text{obst},0}$	$[0,4]$ m/s	$u^i_{\text{obst},0}$	$[0,8]$ m/s
u_0	$[2,4]$ m/s	u_0	$[4,8]$ m/s
–		r_0	$[-4,4]$ 10^{-3} s^{-1}

References

[1] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh, et al., A review: On path planning strategies for navigation of mobile robot, Def. Technol. 15 (4) (2019) 582–606.
[2] N.J. Nilsson, Principles of Artificial Intelligence, Springer Science & Business Media, 1982.
[3] M. Brand, M. Masuda, N. Wehner, X.-H. Yu, Ant colony optimization algorithm for robot path planning, in: 2010 International Conference on Computer Design and Applications, Vol. 3, IEEE, 2010, pp. V3–436.
[4] S.M. LaValle, et al., Rapidly-exploring random trees: A new tool for path planning, TMathematics (1998).
[5] V.J. Lumelsky, T. Skewis, Incorporating range sensing in the robot navigation function, IEEE Trans. Syst. Man Cybern. 20 (5) (1990) 1058–1069.
[6] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robot. Res. 5 (1) (1986) 90–98.
[7] J. Borenstein, Y. Koren, Real-time obstacle avoidance for fast mobile robots, IEEE Trans. Syst. Man Cybern. 19 (5) (1989) 1179–1187.
[8] J. Borenstein, Y. Koren, et al., The vector field histogram-fast obstacle avoidance for mobile robots, IEEE Trans. Robot. Autom. 7 (3) (1991) 278–288.
[9] I. Ulrich, J. Borenstein, VFH+: Reliable obstacle avoidance for fast mobile robots, in: Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146), Vol. 2, IEEE, 1998, pp. 1572–1577.
[10] O. Brock, O. Khatib, High-speed navigation using the global dynamic window approach, in: Proceedings 1999 Ieee International Conference on Robotics and Automation (Cat. No. 99CH36288C), Vol. 1, IEEE, 1999, pp. 341–346.
[11] J. Minguez, L. Montano, Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios, IEEE Trans. Robot. Autom. 20 (1) (2004) 45–59.
[12] R. Simmons, The curvature-velocity method for local obstacle avoidance, in: Proceedings of IEEE International Conference on Robotics and Automation, Vol. 4, IEEE, 1996, pp. 3375–3382.
[13] S. Quinlan, O. Khatib, Elastic bands: Connecting path planning and control, in: [1993] Proceedings IEEE International Conference on Robotics and Automation, IEEE, 1993, pp. 802–807.
[14] V. Kunchev, L. Jain, V. Ivancevic, A. Finn, Path planning and obstacle avoidance for autonomous mobile robots: A review, in: Knowledge-Based Intelligent Information and Engineering Systems: 10th International Conference, KES 2006, Bournemouth, UK, October 9-11, 2006. Proceedings, Part II 10, Springer, 2006, pp. 537–544.
[15] K. Zhu, T. Zhang, Deep reinforcement learning based mobile robot navigation: A review, Tsinghua Sci. Technol. 26 (5) (2021) 674–691.
[16] A.T. Azar, A. Koubaa, N. Ali Mohamed, H.A. Ibrahim, Z.F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A.M. Khamis, I.A. Hameed, et al., Drone deep reinforcement learning: A review, Electronics 10 (9) (2021) 999.
[17] L. Hu, H. Hu, W. Naeem, Z. Wang, A review on COLREGs-compliant navigation of autonomous surface vehicles: From traditional to learning-based approaches, J. Autom. Intell. 1 (1) (2022) 100003.
[18] Y. Wang, H. He, C. Sun, Learning to navigate through complex dynamic environment with modular deep reinforcement learning, IEEE Trans. Games 10 (4) (2018) 400–412.
[19] W. Ding, S. Li, H. Qian, Y. Chen, Hierarchical reinforcement learning framework towards multi-agent navigation, in: 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), IEEE, 2018, pp. 237–242.
[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, 2013, arXiv preprint arXiv:1312.5602.
[21] C. Arvind, J. Senthilnath, Autonomous RL: Autonomous vehicle obstacle avoidance in a dynamic environment using MLP-SARSA reinforcement learning, in: 2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR), IEEE, 2019, pp. 120–124.

[22] X. Xu, P. Cai, Z. Ahmed, V.S. Yellapu, W. Zhang, Path planning and dynamic collision avoidance algorithm under COLREGs via deep reinforcement learning, Neurocomputing 468 (2022) 181–197.
[23] J. Yuan, H. Wang, H. Zhang, C. Lin, D. Yu, C. Li, AUV obstacle avoidance planning based on deep reinforcement learning, J. Mar. Sci. Eng. 9 (11) (2021) 1166.
[24] T. Fan, P. Long, W. Liu, J. Pan, Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios, Int. J. Robot. Res. 39 (7) (2020) 856–892.
[25] L. Kästner, T. Buiyan, L. Jiao, T.A. Le, X. Zhao, Z. Shen, J. Lambrecht, Arena-rosnav: Towards deployment of deep-reinforcement-learning-based obstacle avoidance into conventional autonomous navigation systems, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, pp. 6456–6463.
[26] G.K. Zipf, The Psycho-Biology of Language: An Introduction to Dynamic Philology, routledge, 1936.
[27] E.M. Clerkin, E. Hart, J.M. Rehg, C. Yu, L.B. Smith, Real-world visual statistics and infants' first-learned object names, Philos. Trans. R. Soc. B 372 (1711) (2017) 20160055.
[28] L.B. Smith, S. Jayaraman, E. Clerkin, C. Yu, The developing infant creates a curriculum for statistical learning, Trends Cogn. Sci. 22 (4) (2018) 325–336.
[29] A. Arenas, L. Danon, A. Diaz-Guilera, P.M. Gleiser, R. Guimera, Community analysis in social networks, Eur. Phys. J. B 38 (2004) 373–380.
[30] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Rev. Modern Phys. 74 (1) (2002) 47.
[31] J.L. McClelland, B.L. McNaughton, R.C. O'Reilly, Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory, Psychol. Rev. 102 (3) (1995) 419.
[32] S.C. Chan, A.K. Lampinen, P.H. Richemond, F. Hill, Zipfian environments for reinforcement learning, in: Conference on Lifelong Learning Agents, PMLR, 2022, pp. 406–429.
[33] D.-H. Chun, M.-I. Roh, H.-W. Lee, J. Ha, D. Yu, Deep reinforcement learning-based collision avoidance for an autonomous ship, Ocean Eng. 234 (2021) 109216.
[34] X. Lan, Y. Liu, Z. Zhao, Cooperative control for swarming systems based on reinforcement learning in unknown dynamic environment, Neurocomputing 410 (2020) 410–418.
[35] S. Xie, X. Chu, M. Zheng, C. Liu, A composite learning method for multi-ship collision avoidance based on reinforcement learning and inverse control, Neurocomputing 411 (2020) 375–392.
[36] L. Li, D. Wu, Y. Huang, Z.-M. Yuan, A path planning strategy unified with a COLREGS collision avoidance function based on deep reinforcement learning and artificial potential field, Appl. Ocean Res. 113 (2021) 102759.
[37] C. Zhang, O. Vinyals, R. Munos, S. Bengio, A study on overfitting in deep reinforcement learning, 2018, arXiv preprint arXiv:1804.06893.
[38] M. Hessel, H. Soyer, L. Espeholt, W. Czarnecki, S. Schmitt, H. van Hasselt, Multi-task deep reinforcement learning with popart, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2019, pp. 3796–3803.
[39] K. Cobbe, O. Klimov, C. Hesse, T. Kim, J. Schulman, Quantifying generalization in reinforcement learning, in: International Conference on Machine Learning, PMLR, 2019, pp. 1282–1289.
[40] R. Kirk, A. Zhang, E. Grefenstette, T. Rocktäschel, A survey of generalisation in deep reinforcement learning, 2021, arXiv preprint arXiv:2111.09794.
[41] M.B. Alatise, G.P. Hancke, A review on challenges of autonomous mobile robot and sensor fusion methods, IEEE Access 8 (2020) 39830–39846.
[42] Z.H. Munim, Autonomous ships: a review, innovative applications and future maritime business models, in: Supply Chain Forum: An International Journal, Taylor & Francis, 2019, pp. 266–279.
[43] D. Floreano, R.J. Wood, Science, technology and the future of small autonomous drones, nature 521 (7553) (2015) 460–466.
[44] M. Andretta, Some considerations on the definition of risk based on concepts of systems theory and probability, Risk Anal. 34 (7) (2014) 1184–1195.
[45] R. Zhen, M. Riveiro, Y. Jin, A novel analytic framework of real-time multi-vessel collision risk assessment for maritime traffic surveillance, Ocean Eng. 145 (2017) 492–501.
[46] Z. Liu, Z. Wu, Z. Zheng, A novel framework for regional collision risk identification based on AIS data, Appl. Ocean Res. 89 (2019) 261–272.
[47] R. Szlapczynski, P. Krata, J. Szlapczynska, Ship domain applied to determining distances for collision avoidance manoeuvres in give-way situations, Ocean Eng. 165 (2018) 43–54.
[48] Y. Huang, P. Van Gelder, Time-varying risk measurement for ship collision prevention, Risk Anal. 40 (1) (2020) 24–42.
[49] D. Wilkie, J. Van Den Berg, D. Manocha, Generalized velocity obstacles, in: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2009, pp. 5573–5578.
[50] M. Mahmoodi, K. Alipour, M.T. Masouleh, H.B. Mohammadi, Real-time safe navigation in crowded dynamic environments using Generalized Velocity Obstacles, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 46377, American Society of Mechanical Engineers, 2014, V05BT08A060.
[51] C.K. Peterson, J. Barton, Virtual structure formations of cooperating UAVs using wind-compensation command generation and generalized velocity obstacles, in: 2015 IEEE Aerospace Conference, IEEE, 2015, pp. 1–7.
[52] Y. Huang, L. Chen, P. Van Gelder, Generalized velocity obstacle algorithm for preventing ship collisions at sea, Ocean Eng. 173 (2019) 142–156.
[53] F. Leon, M. Gavrilescu, A review of tracking and trajectory prediction methods for autonomous driving, Mathematics 9 (6) (2021) 660.
[54] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[55] R. Bellman, The theory of dynamic programming, Bull. Amer. Math. Soc. 60 (6) (1954) 503–515.
[56] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.
[57] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[58] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, PMLR, 2018, pp. 1587–1596.
[59] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.
[60] H. Dong, H. Dong, Z. Ding, S. Zhang, Chang, Deep Reinforcement Learning, Springer, 2020.
[61] M. Waltz, O. Okhrin, Two-sample testing in reinforcement learning, 2022, CoRR abs/2201.08078. URL: https://arxiv.org/abs/2201.08078, arXiv:2201.08078.
[62] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: ICML 2010, ICML '10, Omni Press, Madison, WI, USA, 2010, pp. 807–814.
[63] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[64] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al., A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, Science 362 (6419) (2018) 1140–1144.
[65] L. Huang, H. Qu, M. Fu, W. Deng, Reinforcement learning for mobile robot obstacle avoidance under dynamic environments, in: Pacific Rim International Conference on Artificial Intelligence, Springer, 2018, pp. 441–453.
[66] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, J. Pan, Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 6252–6259.
[67] G. Tong, N. Jiang, L. Biyue, Z. Xi, W. Ya, D. Wenbo, UAV navigation in high dynamic environments: A deep reinforcement learning approach, Chin. J. Aeronaut. 34 (2) (2021) 479–489.
[68] M. Treiber, A. Kesting, Elementary car-following models, in: Traffic Flow Dynamics: Data, Models and Simulation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 157–180, http://dx.doi.org/10.1007/978-3-642-32460-4_10.
[69] M. Everett, Y.F. Chen, J.P. How, Motion planning among dynamic, decision-making agents with deep reinforcement learning, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 3052–3059.
[70] L. Zhao, M.-I. Roh, COLREGs-compliant multiship collision avoidance based on deep reinforcement learning, Ocean Eng. 191 (2019) 106436.
[71] X. Xu, Y. Lu, X. Liu, W. Zhang, Intelligent collision avoidance algorithms for USVs via deep reinforcement learning under COLREGs, Ocean Eng. 217 (2020) 107704, http://dx.doi.org/10.1016/j.oceaneng.2020.107704.
[72] R. Rajamani, Vehicle Dynamics and Control, Springer Science & Business Media, 2011.
[73] H. Yasukawa, Y. Yoshimura, Introduction of MMG standard method for ship maneuvering predictions, J. Mar. Sci. Technol. 20 (1) (2015) 37–52.
[74] R.S. Tsay, Analysis of Financial Time Series, John Wiley & Sons, New Jersey, 2010.