Reinforcement Learning-based Path Following for Robots: A Survey of Reward Functions

Jardel dos Santos Dyonisio
Centro de Ciências Computacionais
Universidade Federal de Rio Grande
Rio Grande, Brazil
0009-0003-7722-7203

Stephanie Loi Brião
Centro de Ciências Computacionais
Universidade Federal de Rio Grande
Rio Grande, Brazil
0000-0001-9345-2038

Kristofer Stift Kappel
Centro de Ciências Computacionais
Universidade Federal de Rio Grande
Rio Grande, Brazil
0000-0002-9124-8540

Rodrigo Silva Guerra
Centro de Ciências Computacionais
Universidade Federal de Rio Grande
Rio Grande, Brazil
0000-0003-4011-0901

Emanuel da Silva Diaz Estrada
Centro de Ciências Computacionais
Universidade Federal de Rio Grande
Rio Grande, Brazil
0000-0003-4088-5002

Paulo Lilles Jorge Drews-Jr
Centro de Ciências Computacionais
Universidade Federal de Rio Grande
Rio Grande, Brazil
0000-0002-7519-0502

***Abstract*—Reward functions define an agent's behavior in reinforcement learning by determining actions based on the feedback received. In the domain of path following, reward functions guide the agent toward successfully following the desired path. This paper presents a survey of reward functions focused on path-following tasks for mobile robots. The characteristics, strategies, and challenges associated with creating feedback mechanisms in various domains are highlighted. Components of the reward functions that can be controlled are explored and discussed. Furthermore, for each reward function, scenarios and mobile robots are indicated. Thus, this study provides insights into the reward components that influence reinforcement learning systems in robotic navigation.**

***Index Terms*—Reinforcement Learning, Reward Design, Motion Control, Autonomous Robots.**

I. Introduction

In recent years, autonomous robots have become increasingly prevalent in both everyday life and industrial applications [9], [10]. From self-driving vehicles [17] and mobile robots in warehouses to drones for delivery [13], these technologies are shaping the future of transportation, industrial automation, and various other sectors [33].

These advancements allow robots to make decisions in several environments [2], [18], [24]. The integration of Artificial Intelligence (AI) technologies, such as Deep Learning (DL) and Reinforcement Learning (RL), has enhanced the capabilities of mobile robots, enabling them to perceive and navigate in real time [14], [16]. RL involves training agents to make sequential decisions by maximizing cumulative rewards through interactions with their environment [12], [15]. An important aspect of RL applied to autonomous systems is the design of reward functions. These functions guide the agent's learning process by assigning positive or negative feedback based on the actions taken [6], [8]. However, the challenge lies in defining appropriate reward functions that represent the desired behavior while avoiding unintended consequences.

There is a considerable amount of literature on reward function design for general applications, offering insights into how to create effective reward functions and what factors to consider in the process [8], [11], [23], [27]. These articles discuss various aspects, including the trade-offs between different types of rewards, the challenges in designing rewards that encourage desirable behaviors, and the need to avoid unintended consequences. More specifically, the review [1] focuses on what is relevant in reward functions for autonomous vehicles. Furthermore, it discusses the factors to consider when designing reward functions, such as safety, comfort, progress, and compliance with traffic rules. Thus, the review examines how the literature addresses these reward components, highlighting their strengths, limitations, and challenges in reward function design, including the aggregation of conflicting objectives and the lack of contextual awareness in some formulations.

However, there is a noticeable lack of comprehensive reviews or surveys that focus on reward functions within specific domains. For this survey, the objective is to identify articles related to the application of RL to autonomous navigation, specifically focusing on the reward functions used in the RL algorithms for path-following tasks [7] (Figure 1). The systematic search was based on the following Boolean operators in the source Google Scholar: (”reinforcement learning”) AND (”path” OR ”trajectory” OR ”trajectories”) AND (”tracking” OR ”following”) published in the last ten years.

The main contributions of this paper are as follows:

- an overview of reward function strategies aimed at guiding the design of reward for mobile robot path-following control;
- a standardization of reward functions commonly employed in path-following tasks, with a focus on key components such as classical terms, progress-based distance,

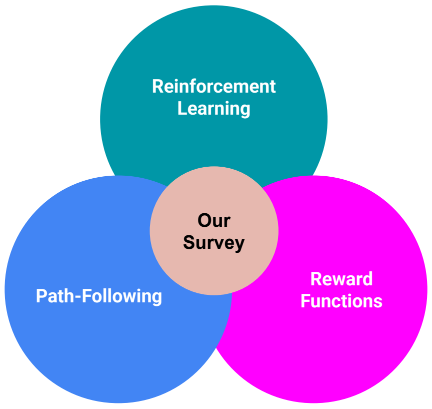

Fig. 1. Main key-words for this work and its connections

acceleration penalties, lateral deviation, steering control, angular deviation, and velocity regulation;
- a summary of characteristics, practical contexts, domain of applications, and challenges of reward functions.

The paper is organized as follows: Section II, which presents the categories and equations of the reward functions. In Section III, the challenges and applications of the reward functions are presented. Finally, Section IV shows the importance and justification of this review. Appendix A includes the equations used to better understand the reward functions.

II. Reward Functions for Path Following

The reward functions can be of various types based on the specific components of the agent's behavior they aim to control. In this section, we examine several reward functions commonly used in path-following. Each function considers different mobile robots, scenarios, applications, features, advantages, and limitations (see Table I).

A. Classic Rewards (CR)

In several situations, agents should receive feedback for simple actions, such as when they complete a task or encounter a specific event. The classic reward r_c for reaching the goal is defined by [3], [19]:

$$R_{\mathrm{c}} = k_{c_1}. \tag{1}$$

In other words, assign positive feedback when the agent reaches a waypoint. Otherwise, the robot may receive penalties, such as [3]:

$$R_{\mathrm{c}} = -k_{c_2}. \tag{2}$$

In this case, the function r_c applies negative feedback when the agent crashes or fails in some way.

B. Progress-based Reward (PR)

This component of the reward functions aims to encourage the agent to progress along the desired path or toward specific way-points. The agent's progress p_y can be quantified by calculating the variation in distance (14). The corresponding progress reward function R_p is computed using the distance variation and a scaling coefficient k_p, as expressed [3], [19]:

$$R_p = k_p \times p_y. \tag{3}$$

When the agent advances along the path, the reward is positive. In contrast, when the agent moves backwards, the reward becomes negative or decreases, motivating the agent to stay on course.

C. Acceleration-based Reward (AR)

The acceleration reward function R_a penalizes rapid changes in acceleration, encouraging the agent to avoid sudden jerks or oscillations in its movement as follows [31]:

$$R_a = -k_a \times a_y. \tag{4}$$

The term a_y represents the lateral acceleration of the agent along the path. The constant k_a controls the magnitude of the penalty for rapid acceleration changes. A negative value of R_a is assigned when the agent accelerates, with a higher penalty for faster acceleration rates, discouraging sudden or jerky movements.

D. Lateral error-based Reward (LER)

The lateral error e_y quantifies the agent's deviation from the reference path ((17), (18) and (19). By penalizing large lateral errors, this reward function encourages the agent to stay as close as possible to the desired path. The lateral error reward function R_e is of the form [25], [34]:

$$R_e = e^{-k_{e_y}|e_y|} \tag{5}$$

utilizes a Gaussian function, providing smooth and differentiable penalties, which are advantageous for gradient-based optimization. This design reinforces the need for the agent to minimize deviations from the desired path. Another reward function R_e, that also provides a Gaussian reward based on the lateral error e_y is [4], [21], [22]:

$$R_e = \begin{cases} k_{e_y} e^{-\frac{e_y^2}{2\sigma}} & \text{if } |\tilde{\psi}| < \frac{\pi}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The reward is highest when the error is small and decreases exponentially as the lateral error increases. However, the key difference in this approach lies in the added condition based on the yaw error $\tilde{\psi}$, which represents the difference between the robot's yaw angle (15) and the path heading. Suppose that the yaw error $|\tilde{\psi}| < \pi/2$. In this case, the reward given ensures that the agent is rewarded for staying close to the path while considering the agent's orientation relative to the goal. However, if the yaw error is greater than $\pi/2$, the reward is set to zero, penalizing the agent for deviating too far from the ideal path in terms of both lateral error and orientation.

Additionally, the difference lies in the 2σ term. The original equation uses $\sigma = 10$ m, chosen based on the scale of the navigation task. For agents operating in large environments, this value of σ is appropriate, as it provides the agent with

TABLE I
CONTEXTS AND CHARACTERISTICS OF EACH REWARD FUNCTION

Context	**CR function** R_c	**PR function** R_p	**AR function** R_a	**LER function** R_e	**SCR function** R_s	**ADR function** R_d	**VR function** R_v
Features	Fixed reward when the goal is reached	Rewards forward motion along the path	Penalizes high accelerations or jerk	Penalizes perpendicular distance to path	Penalizes jerky steering changes	Penalizes deviation from the desired angle	Rewards maintaining target speed or penalizing stops
Advantages	Easy to implement and interpret	Encourages efficient movement	Promotes smooth and stable motion	Ensures precise tracking	Encourages smooth transitions	Improves alignment, reduces oscillations	Keeps motion consistent, avoids stalls
Limitations	Sparse feedback; no path guidance	Ignores lateral deviation	Low responsiveness in dynamic scenes	Ignores orientation	Needs tuning for different dynamics	Sensitive to noise or curvature	May cause instability if not coupled properly
Scenarios	Simulated or structured environments	Obstacle-free or semi-structured settings	Uneven terrain or dynamic weather	Marked lanes or guided corridors	Sharp turns or curved trajectories	Orientation-critical conditions	Long-range or time-sensitive missions
Mobile Robots	Differential, Omnidirectional, UAV, AUV	Ackermann, Differential, Fixed-wing UAV, USV	Legged, Multirotor UAV, ASV	Differential, Omnidirectional, Fixed-wing UAV, AUV	Ackermann, Differential, VTOL, USVs with rudder	Omnidirectional, Docking AUV, Indoor UAV	High-speed terrestrial, Fixed-wing UAV, endurance USV
Applications	Navigation success, docking detection	Trajectory following, corridor inspection	Smooth navigation in harsh environments	Pipeline following, indoor navigation	Stable maneuvering and curve execution	Docking alignment, payload positioning	Racing, area coverage, urgent delivery

an acceptable error margin of up to 10 meters before being significantly penalized. An alternative approach to measuring the reward for cross-track error is the following equation [30]:

$$R_e = -k_{e_y}|e_y|, \tag{7}$$

which uses a linear reward function instead of the Gaussian function. This equation provides a constant reward proportional to the magnitude of the error, without the smooth exponential decay characteristic of the Gaussian function. The fourth approach in this category is based on the average tracking error $\bar{e}_y$ over a certain period or set of samples, as shown [31]:

$$R_e = -k_{e_y}|\bar{e}_y|. \tag{8}$$

Then, this method smooths out the reward function over time, making it less sensitive to short-term fluctuations.

E. Steering Control-based Reward (SCR)

This category penalizes abrupt or rapid steering changes, which can lead to instability or erratic behavior. The first approach is the standard deviation reward function R_s in (9). This function is based on the most recent N steering command values, calculated using (20). Designed to penalize rapid, oscillatory, or chattering behavior in steering control. The reward function is an exponential Gaussian function of the form [34]:

$$R_s = e^{-k_{\sigma_\delta}\sigma_\delta}. \tag{9}$$

An alternative approach is to calculate the reward based on the change in the steering rate R_s, as shown [21], [22]:

$$R_s = -k_\delta|\dot{\delta}| \tag{10a}$$

$$R_s = -k_{\dot{\delta}}\dot{\delta}^2. \tag{10b}$$

These rewards are derived from the steering angle change rate $\dot{\delta} = \frac{d(\delta(t))}{dt}$ for $\delta(t)$ the steering angle at time t. The function R_s is used to penalize excessive changes in the steering angle. For a linear approach in this category, (10a) is used. Furthermore, in (10b), the term $k_{\dot{\delta}}$ is a constant that determines the magnitude of the reward applied to rapid changes in steering rate. To make the system more sensitive to rapid steering adjustments, the square derivative is used to amplify the penalty. In this equation, rewards are calculated based on the steering change rate, making the system easily adaptable to other data, such as angular velocity, linear velocity, or another metric that helps prevent abrupt changes and encourages smoother motion.

F. Angle Deviation-based Reward (ADR)

Angle deviation rewards R_d are designed to control the agent's heading and orientation along the path. These rewards penalize the difference between the agent's current heading and the desired direction of travel. Thus, in path following, the agent must maintain a consistent orientation along the path, especially in environments where precise control over direction is necessary. One common approach is the use of angle deviation reward, as follows [34]:

$$R_d = \begin{cases} e^{-k_{\theta_r}|\theta_r|} & \text{if } |\theta_r| < 90^\circ, \\ -e^{-k_{\theta_r}(\theta_r - 180)} & \text{if } \theta_r \geq 90^\circ, \\ -e^{-k_{\theta_r}(\theta_r + 180)} & \text{if } \theta_r \leq -90^\circ. \end{cases} \tag{11}$$

This function penalizes deviations in the agent's heading relative to the desired path direction. In (11), θ_r represents the angle deviation. In other words, θ_r is the difference between the agent's current direction and the desired path direction. The coefficient k_{θ_r} is the scaling factor that adjusts the penalty for the deviation from the reference value. Furthermore, the function provides an exponential decay in the reward as the angle deviation increases, with different behavior for deviations within and beyond 90°.

G. Velocity-based Reward (VR)

In certain applications, when agents are constrained by time or need to complete tasks efficiently, encouraging higher speeds is crucial. To promote it, the reward function for velocity is defined [19]:

$$R_v = k_v v, \tag{12}$$

which is based on the agent velocity v and on the coefficient factor k_v that adjusts the magnitude of the reward. An alternative approach in this category, which involves using velocity to calculate progress, is represented by [30]:

$$R_v = k_v v_T, \tag{13}$$

whose basis is the projected velocity onto the x-axis of the tangential frame of reference v_T (see (16)). The velocity reward function R_v provides a positive reward when the agent is moving forward along the path and a negative reward when it moves in the opposite direction based on its velocity.

III. Challenges of Reward Design in Different Domains

The main challenge in designing reward functions for path following is ensuring that the reward correctly guides the agent along the desired path. However, there are different challenges for each applied environment; a summary of this context is given in Figure 2. When multiple components are considered as distance error, angle deviation, travel time, velocity, acceleration, stability, and smoothness of movements, a function that best fits the domain [5], [29] (see Table I).

In maritime environments, Autonomous Underwater Vehicles (AUVs) [26], [35] and Unmanned Surface Vehicles (USVs) [28], [36] have more flexibility in path following, as these environments typically present minimal collision risks [20]. In contrast to urban or high-density areas, maritime and aerial environments often have fewer obstacles, allowing these vehicles to deviate from the ideal path without significantly impacting overall performance or safety. This increased freedom of movement reduces the need to maintain a low lateral path error, as the potential for collisions or other hazards is minimal.

However, both maritime and aerial environments pose specific challenges. In maritime scenarios, the environment is often featureless and homogeneous, which can hinder precise localization due to the lack of visual or geometric landmarks. Additionally, water currents and wave dynamics introduce slow but persistent disturbances. In aerial environments, while similarly sparse in obstacles, vehicles are operated in three-dimensional space. This added degree of freedom, along with susceptibility to fast-changing disturbances such as wind and turbulence. As a result, reward functions in these environments can prioritize other actions while ensuring deviations from the path do not compromise mission objectives.

In the autonomous driving domain [32], maintaining a path following with a low lateral error is relevant, as even slight deviations can lead to safety issues, especially in urban settings where there is a need to account for obstacles such as pedestrians, other vehicles, and traffic signals. Autonomous vehicles are typically required to stay within narrow lanes and follow specific routes. As such, path-following reward functions in this domain often emphasize minimizing lateral error and ensuring the vehicle's velocity and angular velocity are adjusted smoothly to avoid abrupt changes that could compromise passenger comfort or vehicle stability [19].

In warehouse navigation and automated material handling within logistics environments, vehicles or robots often operate in structured settings with predefined paths, surrounded by restricted areas dictated by industry norms. These areas are designated for robots, people, or specific operations, such as narrow corridors filled with shelves. These systems must minimize lateral error and ensure smooth path adjustments without prioritizing passenger comfort. However, they must still avoid abrupt changes in velocity and steering. The presence of narrow corridors adds a layer of complexity, requiring careful navigation to prevent collisions with shelves, obstacles, and people. In minimizing task duration, the agent must reach the target location quickly while avoiding collisions. Reward functions in this domain typically account for factors such as time to destination, travel distance, and the number of obstacles encountered while allowing some flexibility in the path following, provided that deviations remain within acceptable limits.

IV. Conclusions

The proper definition and adaptation of reward functions play an important role in reinforcement learning, providing feedback that shapes the agent's behavior and actions. While RL is an emerging technology with widespread applications, the literature on reward function surveys remains limited. The lack of review studies is particularly noticeable when focused on specifically path-following tasks. These gaps highlight the need for further research on the various approaches and challenges associated with reward functions in this domain. Given this, the current paper presents foundational reward functions. These equations have been adapted for better understanding and applicability to different categories and scenarios. Thus, the paper contributes to the review of reward functions, particularly in the domain of path-following tasks that utilize the RL method. Based on the insights gained from this work, future research can focus on developing a taxonomy of reward functions that considers both the agent's objective and the environment in which it is applied. Additionally, develop generalizations of the functions based on the robot's objective or the operating scenario.

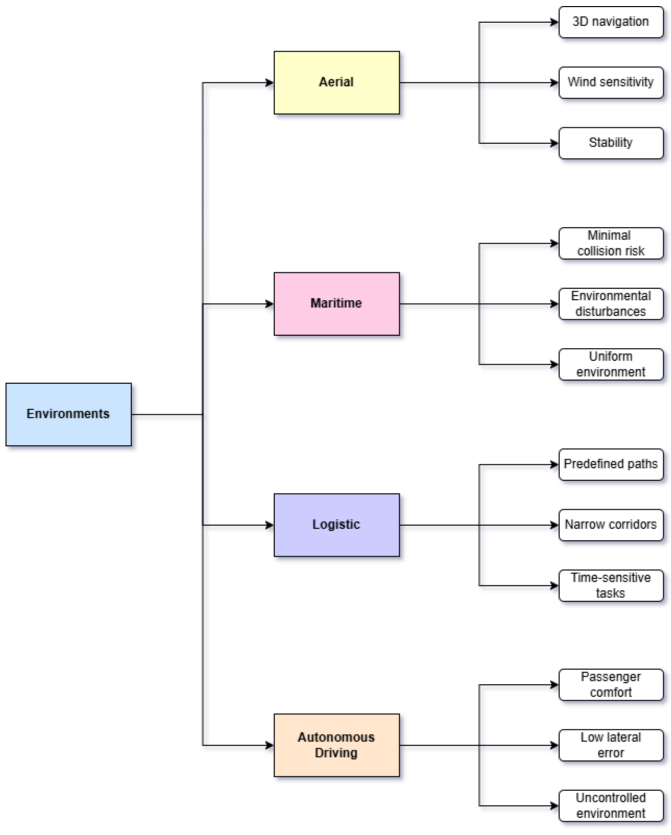

Fig. 2. Summarized challenges for each environment.

Acknowledgment

This work was partly supported by FINEP, FAURG, FAPERGS, and CNPQ.

References

[1] Abouelazm, A., Michel, J., Zöllner, J.M.: A review of reward functions for reinforcement learning in the context of autonomous driving. In: IEEE IV. pp. 156–163 (2024)

[2] Alatise, M.B., Hancke, G.P.: A review on challenges of autonomous mobile robot and sensor fusion methods. IEEE Access **8** (2020)

[3] Aljalbout, E., Krinner, M., Romero, A., Scaramuzza, D.: Accelerating model-based reinforcement learning with state-space world models. In: ICLRw (2025)

[4] Cao, Y., Ni, K., Kawaguchi, T., Hashimoto, S.: Path following for autonomous mobile robots with deep reinforcement learning. Sensors **24**(2) (2024)

[5] Chunyang, S., Hao, A., Jun, N., Haixia, W., Xiao, L.: Improved m-dqn with ϵ-UCB action selection policy and multi-goal fusion reward function for mobile robot path planning. IEEE TVT (2024)

[6] Drid, A.H., Debilou, A., Chouchane, A., Lahmar, O., Ghiloubi, I.B.: Reward function analysis for deep reinforcement learning agent in goal-oriented navigation. In: ISNIB. pp. 1–6 (2025)

[7] Dyonisio, J.S., Milczarek, L.F., Dias, N.F., Brião, S.L., Dorneles, G.A., Maurell, I.P., Pinheiro, P.M., Guerra, R.S., Drews-Jr, P.L.J.: Teach and repeat for path planning using bézier curves. In: SBR/WRE (2024)

[8] Eschmann, J.: Reward Function Design in Reinforcement Learning. Springer International Publishing (2021)

[9] Grando, R.B., de Jesus, J.C., Kich, V.A., Kolling, A.H., Bortoluzzi, N.P., Pinheiro, P.M., Neto, A.A., Drews-Jr, P.L.J.: Deep reinforcement learning for mapless navigation of a hybrid aerial underwater vehicle with medium transition. In: IEEE ICRA. pp. 1088–1094 (2021)

[10] Grando, R.B., de Jesus, J.C., Kich, V.A., Kolling, A.H., Drews-Jr, P.L.J.: Double critic deep reinforcement learning for mapless 3d navigation of unmanned aerial vehicles. JIRS **104**(2), 29 (2022)

[11] Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S.J., Dragan, A.: Inverse reward design. NeurIPS **30** (2017)

[12] Hong, L., Chenyan, X., Hengyu, L.: Research on geomagnetic sensing navigation method based on deep reinforcement learning+ simulated annealing. In: ICMEE. pp. 63–68 (2024)

[13] de Jesus, J.C., Kich, V.A., Kolling, A.H., Grando, R.B., Guerra, R.S., Drews, P.L.: Depth-cuprl: Depth-imaged contrastive unsupervised prioritized representations in reinforcement learning for mapless navigation of unmanned aerial vehicles. In: IEEE/RSJ IROS. pp. 10579–10586 (2022)

[14] de Jesus, J.C., Kich, V.A., Kolling, A.H., Grando, R.B., da Silva Guerra, R., Drews-Jr, P.L.J.: Image-based mapless navigation of a hybrid aerial-underwater vehicle using prioritized deep reinforcement learning. JIRS **111**(1), 27 (2025)

[15] bin Kamarulariffin, A., Ibrahim, A.b.M., Bahamid, A.: Improving deep reinforcement learning training convergence using fuzzy logic for autonomous mobile robot navigation. IJACSA **14**(11) (2023)

[16] Kich, V.A., Kolling, A.H., de Jesus, J.C., Heisler, G.V., Jacobs, H., Bottega, J.A., Da S. Kelbouscas, A.L., Ohya, A., Grando, R.B., Drews-Jr, P.L.J., Gamarra, D.F.T.: Parallel distributional deep reinforcement learning for mapless navigation of terrestrial mobile robots. In: ICCAS. pp. 978–983 (2024)

[17] Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A.A.A., Yogamani, S., Pérez, P.: Deep reinforcement learning for autonomous driving: A survey. IEEE T-ITS **23**(6), 4909–4926 (2022)

[18] Kolling, A.H., Kich, V.A., de Jesus, J.C., Da Silva, A.C., Grando, R.B., Drews, P.L.J., Gamarra, D.F.: Parallel distributional prioritized deep reinforcement learning for unmanned aerial vehicles. In: LARS/SBR. pp. 95–100 (2023)

[19] Liu, J., Cui, Y., Duan, J., Jiang, Z., Pan, Z., Xu, K., Li, H.: Reinforcement learning-based high-speed path following control for autonomous vehicles. IEEE TVT **73**(6), 7603–7615 (2024)

[20] Malviya, A., Rajendran, S.: Multi-agent reinforcement learning for collision avoidance and path following of autonomous surface vehicles. In: IEEE UT. pp. 1–6 (2025)

[21] Martinsen, A.B., Lekkas, A.M.: Curved path following with deep reinforcement learning: Results from three vessel models. In: MTS/IEEE OCEANS. pp. 1–8 (2018)

[22] Martinsen, A.B., Lekkas, A.M.: Straight-path following for underactuated marine vessels using deep reinforcement learning. IFAC-PapersOnLine **51**(29), 329–334 (2018)

[23] Mataric, M.J.: Reward functions for accelerated learning. In: Cohen, W.W., Hirsh, H. (eds.) ICML, pp. 181–189. San Francisco (CA) (1994)

[24] Mateus, M.G., Grando, R.B., Drews-Jr, P.L.: Active perception applied to unmanned aerial vehicles through deep reinforcement learning. In: LARS/SBR/WRE. pp. 1–6 (2022)

[25] Meyer, E., Robinson, H., Rasheed, A., San, O.: Taming an autonomous surface vehicle for path following and collision avoidance using deep reinforcement learning. IEEE Access **8**, 41466–41481 (2020)

[26] Moe, S., Pettersen, K.Y., Fossen, T.I., Gravdahl, J.T.: Line-of-sight curved path following for underactuated usvs and auvs in the horizontal plane under the influence of ocean currents. In: Mediterranean Conference on Control and Automation (MED). pp. 38–45 (2016)

[27] Niekum, S., Spector, L., Barto, A.G.: Evolution of reward functions for reinforcement learning. GECCO (2011)

[28] Paulig, N., Okhrin, O.: Robust path following on rivers using bootstrapped reinforcement learning. Ocean Engineering **298**, 117207 (2024)

[29] Rubí, B., Morcego, B., Pérez, R.: A deep reinforcement learning approach for path following on a quadrotor. In: ECC (2020)

[30] Rubí, B., Morcego, B., Pérez, R.: Deep reinforcement learning for quadrotor path following with adaptive velocity. Autonomous Robots **45**(1) (2021)

[31] Shan, Y., Zheng, B., Chen, L., Chen, L., Chen, D.: A reinforcement learning-based adaptive path tracking approach for autonomous driving. IEEE TVT **69**(10), 10581–10595 (2020)

[32] Shan, Y., Zheng, B., Chen, L., Chen, L., Chen, D.: A reinforcement learning-based adaptive path tracking approach for autonomous driving. IEEE TVT **69**(10), 10581–10595 (2020)

[33] Singh, B., Kumar, R., Singh, V.P.: Reinforcement learning in robotic applications: a comprehensive survey. Artificial Intelligence Review **55**(2), 945–990 (2022)

[34] Woo, J., Yu, C., Kim, N.: Deep reinforcement learning-based controller for path following of an unmanned surface vehicle. Ocean Engineering **183**, 155–166 (2019)

[35] Yu, R., Shi, Z., Huang, C., Li, T., Ma, Q.: Deep reinforcement learning based optimal trajectory tracking control of autonomous underwater vehicle. In: CCC. pp. 4958–4965 (2017)

[36] Zhao, Y., Qi, X., Ma, Y., Li, Z., Malekian, R., Sotelo, M.A.: Path following optimization for an underactuated usv using smoothly-convergent deep reinforcement learning. IEEE T-ITS **22**(10), 6208–6220 (2020)

Appendix

The Euclidean distance d in path-following algorithms calculates the distance between the agent and the waypoint or two points along the desired path as:

$$d = \sqrt{(y_a - y_b)^2 + (x_a - x_b)^2}. \tag{14}$$

The yaw angle θ between two discrete points can be calculated using ($\operatorname{atan2}$, being the 2-argument arctangent):

$$\theta = \operatorname{atan2}(y_a - y_b, x_a - x_b). \tag{15}$$

If the path is parameterized, the yaw angle is calculated based on its derivatives at a specific point, and the path is represented as a continuous function.

The velocity projected on the x-axis of the tangential frame of reference v_T is calculated by:

$$v_T = v_x \cos(\theta) + v_y \sin(\theta). \tag{16}$$

Considering the current position (x, y) of the agent, the current waypoint (x_k, y_k), and the yaw angle θ, between the next and current waypoint, the lateral error normal e_y is:

$$e_y = -(x - x_k)\sin(\theta) + (y - y_k)\cos(\theta). \tag{17}$$

An alternative method for calculating the lateral error e_y, considering the path segment between the way-points, is:

$$e_y = \sin(\theta_r) d_{W_{k-1}} \tag{18}$$

for $d_{W_{k-1}}$ representing the Euclidean distance between the agent and the previous waypoint. The longitudinal error e_x represents the deviation from the desired path and is used to track the agent's progress along the path:

$$e_x = \cos(\theta_r) d_{W_{k-1}}. \tag{19}$$

The standard deviation of steering commands is used to quantify the variability in the agent's steering behavior, as:

$$\sigma_\delta = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\delta_i - \bar{\delta})^2}, \tag{20}$$

where δ_i represents the individual steering command values and $\bar{\delta}$ is the average of the steering commands over the last N readings.