Aquatic Navigation: A Challenging Benchmark for Deep Reinforcement Learning

Davide Corsi
dcorsi@uci.edu
Department of Computer Science
University of California, Irvine

Davide Camponogara
davide.camponogara@studenti.univr.it
Department of Computer Science
University of Verona

Alessandro Farinelli
alessandro.farinelli@univr.it
Department of Computer Science
University of Verona

Abstract

An exciting and promising frontier for Deep Reinforcement Learning (DRL) is its application to real-world robotic systems. While modern DRL approaches achieved remarkable successes in many robotic scenarios (including mobile robotics, surgical assistance, and autonomous driving) unpredictable and non-stationary environments can pose critical challenges to such methods. These features can significantly undermine fundamental requirements for a successful training process, such as the Markovian properties of the transition model. To address this challenge, we propose a new benchmarking environment for aquatic navigation using recent advances in the integration between game engines and DRL. In more detail, we show that our benchmarking environment is problematic even for state-of-the-art DRL approaches that may struggle to generate reliable policies in terms of generalization power and safety. Specifically, we focus on PPO, one of the most widely accepted algorithms, and we propose advanced training techniques (such as curriculum learning and learnable hyperparameters). Our extensive empirical evaluation shows that a well-designed combination of these ingredients can achieve promising results. Our simulation environment and training baselines are freely available to facilitate further research on this open problem and encourage collaboration in the field.

1 Introduction

In recent years, Deep Reinforcement Learning (DRL) methods have advanced rapidly and achieved impressive results in various domains. For instance, modern DRL algorithms, such as TD3 (Fujimoto et al., 2018), SAC (Haarnoja et al., 2018a), PPO (Schulman et al., 2017), or Rainbow (Hessel et al., 2018), have demonstrated remarkable capabilities in solving highly complex problems, ranging from video games (Mnih et al., 2013) to complex decision-making tasks and robotic applications (Kober et al., 2013; Rolf et al., 2023). However, even the state-of-the-art algorithms struggle when dealing with unpredictable and non-stationary environments, where the basic Markovian properties may be violated (Marchesini et al., 2021).

In this direction, the limited availability of challenging benchmarking environments, where even state-of-the-art algorithms fail to achieve optimal performance, makes it difficult to evaluate the impact of new DRL methods and advanced learning approaches. This is especially relevant in the field of robotics, where issues related to safe control are often impossible to separate from the hardware and therefore not readily available to the community as a benchmark (Aractingi et al., 2023; Akkaya et al., 2019). Moreover, a common limitation shared by almost all the DRL algorithms

lies in their data efficiency (Lillicrap et al., 2015; Haarnoja et al., 2018b). In the context of robotics, this limitation assumes particular significance due to the challenging process of data collection. Collecting real-world data on the actual robot can be slow and dangerous, especially when factors such as human safety or the use of expensive hardware are involved. Consequently, the development of realistic simulators for the training process has emerged as a priority (Attanasio et al., 2020; Pore et al., 2021; Amir et al., 2023a).

Against this background, the first contribution of this paper involves developing a simulator designed for aquatic navigation, that considers both surface and underwater scenarios (see Fig.1). In this type of environment, many of the aforementioned issues related to the complex and unpredictable evolution of water can arise, potentially compromising training performance. Specifically, our focus is on autonomous navigation, which is increasingly used for important tasks such as exploration, cable monitoring, security, and seabed mapping in oceans and lakes (Carreras et al., 2018; Wynn et al., 2014). The simulator is tailored specifically for this purpose and addresses several critical requirements. First, it is designed to be lightweight and high-performing, enabling multiple executions to collect the substantial volume of data necessary for effective training. Second, it allows a wide range of customization possibilities to replicate different real-world environments. Finally, it strongly emphasizes realism, crucial for training autonomous agents in environments that mirror real-world challenges. Mapless navigation problems can generally be addressed with Deep Reinforcement Learning (DRL) approaches (Zhu et al., 2017; Bojarski et al., 2016; Marchesini & Farinelli, 2021; 2022); nevertheless, our results show that the unpredictable nature of the environment makes the task much more challenging for the DRL agent.

Our second contribution is a pipeline for training and validating a DRL agent to provide a stable baseline for comparison with future work and algorithmic improvements. We rely on Proximal Policy Optimization (PPO) (Schulman et al., 2017), a state-of-the-art reinforcement learning algorithm that has shown groundbreaking results across a wide range of tasks. While PPO offers a general approach for reinforcement learning problems, achieving satisfactory results demands careful consideration of problem-specific configurations (Engstrom et al., 2020; Corsi et al., 2024) and additional optimization tricks and implementation details (Schulman et al., 2015; Marchesini & Amato, 2023; Liang et al., 2022). Throughout this paper, we present a comprehensive set of ablation studies that support our ultimate design choices, highlighting the limitations that even state-of-the-art algorithms can have on such a complex problem. This work emphasizes the critical problem of safety in the domain of autonomous navigation. The involvement of expensive equipment and the inherent challenges associated with potential rescue operations make safety a particularly relevant concern (Fossen, 2011). Our benchmark includes additional safety requirements, making it suitable for research in the field of safe deep reinforcement learning (Corsi et al., 2021; Ray et al., 2019; Yerushalmi et al., 2022).

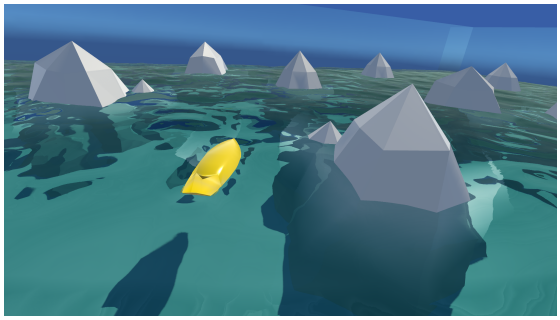

Figure 1: The figures depict two environments within our simulator. The first figure shows our Autonomous Underwater Vehicle (AUV) navigating a 3D model of Porth yr Ogof marine cave, while the second figure shows our surface drone in one of the scenarios from our marine benchmark. Although the two environments differ in their objective, they share the main challenges introduced by the aquatic environment.

Finally, to further validate the effectiveness of our agent, we extensively test our baseline on a cave navigation scenario that is created using real-world data. In detail, we recreate the *Porth yr Ogof* cave, located in South Wales; where we then deploy our trained agent without any prior knowledge of the cave's structure. The results demonstrate that the agent is able to explore the entire cave while avoiding catastrophic collisions. However, a more detailed analysis showed some limitations in terms of generalization power and safety against specific corner cases, even employing state-of-the-art solutions. For this reason, we believe that this challenging environment can be of great value to the DRL community, and not strictly limited to applications in water navigation domains. Our analysis shows that solving these tasks – taking into account the safety and generalization aspects – is still an open problem, and we believe that it can be considered a challenging benchmark to validate novel learning tools and algorithms. Crucially, to promote further research and collaboration in this domain, we offer open access to our simulation environment and training algorithms[1][2].

2 Related Work

In the previous section, we discuss a critical problem in DRL, the high amount of data necessary for an effective learning process (Lillicrap et al., 2015; Haarnoja et al., 2018b). Collecting all these experiences can be hard in a robotic context, where expensive equipment is involved and failures can be barely tolerated. A common solution is the exploitation of realistic simulation engines. Historically, the robotic community has relied on software such as RViz (for visualization) and Gazebo (for simulation); however these solutions are not designed to support fast computation and parallel execution, both necessary requirements in a DRL context (Zhao et al., 2020; Azar et al., 2023). In contrast, standard benchmarks for DRL rely on libraries such as MuJoCo, Bullet, or PyGame to approximate the real-world dynamics, often sacrificing the accuracy of the physics simulation to obtain faster computation (Gronauer, 2022; Ray et al., 2019). To bridge this gap modern approaches propose the use of 3D simulation engines typically developed for video games such as Unreal (De Melo et al., 2019), Coppelia (Nogueira, 2014), or Unity3D (Juliani et al., 2020). In this work, we focus on the latter, which has been recently successfully employed as a simulation engine for robotic research (Technologies, 2020). Unity offers unique capabilities to fasten the simulation, such as a *server mode* that allows computing the simulation without the rendering part, *time acceleration*, and the *synchronous execution* to allow easier integration with the state-of-the-art DRL libraries. In fact, a crucial advantage of Unity3D with respect to other engines is the built-in package *Unity ML-Agents*, which provides full compatibility with Gym, a standardized set of API for DRL research (Juliani et al., 2020). We believe this is a critical asset to foster the wider use of a DRL benchmark in the community as Gym is the de facto standard interface for the most popular DRL implementation (e.g., *Stable-Baselines*, *SpinningUP*, *CleanRL*, and more). There are already underwater navigation simulators that accurately simulate the physical characteristics of these scenarios (Lončar et al., 2022; Cieślak, 2019), but they are not simulators designed for DRL, but rather for data collection and dataset generation from simulations. This represents a significant limitation for DRL practitioners, as it does not allow for a straightforward integration of the learning algorithm. For example, DRL algorithms are designed to solve variations of a Markov Decision Process, which requires a discretization of the time. This is particularly challenging in the context of complex physics simulations (e.g., water). Moreover, our environment allows for easy access to the reward (and cost) function and a clear and explicit definition of the state and action spaces. Finally, the results presented in this paper provide a fair baseline for future algorithm and approach development.

Navigation and Mapless Navigation

We focus on the problem of navigating a robot through an environment, to reach a specific target position. Typically, the agent should adhere to additional constraints, that may include finding the shortest path, avoiding obstacles, or optimizing energy consumption. In the last years, this problem

[1] https://github.com/dadecampo/aquatic_navigation_envs
[2] https://github.com/dadecampo/SafeRLAUV

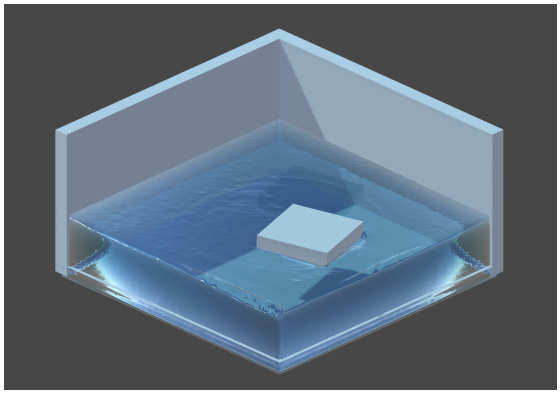

Figure 2: Viscous liquid.

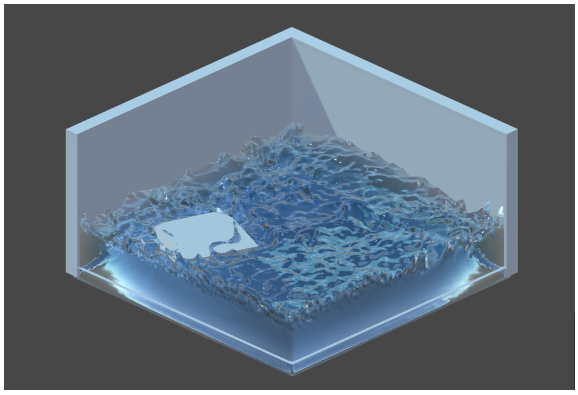

Figure 3: Runny liquid.

has gathered increasing attention, particularly due to its relevance in the context of autonomous vehicles (Pan et al., 2017), and it is today considered one of the classical problems in robotics. Robotic Navigation has been extensively studied over the years, resulting in various algorithmic solutions such as planning and search-based approaches (LaValle, 2006; Latombe, 2012). Nevertheless, a variant of robotic navigation, known as *mapless navigation*, has recently emerged as a popular problem, and a standard benchmark for DRL, that presents additional unique challenges (Zhu et al., 2017; Marchesini & Farinelli, 2022). In mapless navigation, the robot operates within the environment without using a map, relying solely on its local observations. This configuration introduces additional complexities, as the absence of a map hinders the use of conventional planning-based methods. Moreover, limited sensor information makes the problem *partially observable*, giving rise to additional challenges such as sensor noise and the uncertainty of action outcomes (Marchesini & Farinelli, 2022). State-of-the-art solutions for mapless navigation suggest exploiting DRL techniques to generate policies capable of controlling the autonomous vehicle; these solutions have demonstrated exceptional performance (Bojarski et al., 2016) and are nowadays considered a clear example of the DRL's potential. Moreover, recent works show that these kinds of problems can be accomplished by employing relatively simple and small DNN architectures, which is essential for enabling on-board control of the robot, where resource constraints require compact models.

3 Simulation Environment

In this section, we introduce our environment, presenting the first contribution of the paper: a realistic underwater simulator based on the Unity3D game engine. Unity has emerged as a powerful tool for the development of Reinforcement Learning agents in simulated scenarios, especially in the domain of robotics research (Juliani et al., 2020). As a fundamental building block for our experiments, we developed a virtual environment tailored to closely emulate the challenges of aquatic environments. To replicate the hydrodynamic aspects of water, we rely on ZibraAI Liquids (ZibraAI, 2021), a state-of-the-art solution for real-time 3D liquid simulation. This versatile tool provides us with the flexibility to manipulate a wide array of parameters, encompassing liquid physics settings and interaction dynamics with other physical objects in the environment.

3.1 Simulation of Fluid Behavior

Simulating the behavior of fluids has always been considered a hard challenge, especially due to the multitude of intricate physical forces involved (Lončar et al., 2022; Cieślak, 2019). Moreover, precisely reproducing all these forces in real time remains impractical, necessitating a significant degree of approximation. ZibraAI enables the creation of water zones that can be precisely parameterized to adjust viscosity, surface tension, and other essential characteristics.

Through rigorous experimentation with various settings, we identified an optimal configuration that aligned perfectly with our research objectives, paving the way for the subsequent development of our aquatic environment. An illustrative example of Zibra's plugin capabilities can be found in Fig.2 and Fig.3, where we show the interaction between a simple object and a liquid of varying viscosity. The underlying mechanics of ZibraAI's operation involve a novel approach to encoding a 3D object into concise vectors, subsequently decoded by a compact neural network to regenerate the original Signed Distance Field (SDF). This innovative technique finds practical application in gaming physics, particularly in particle simulations. The Zibra Liquids Pro plugin represents a collaborative synergy between proprietary physical solvers and machine learning-based neural representations of objects (ZibraAI, 2021). Additionally, ZibraAI has internally developed fluid simulation technology utilizing the Moving Least Squares Material Point Method (Hu et al., 2018). Early experiments have demonstrated the remarkable efficiency of this approach, capable of simulating 300,000 particles in only 7 milliseconds on a GTX 1050, even without extensive optimizations. The ZibraAI plugin is publicly available, further contributing to the advancement of fluid simulation research, resulting in a critical asset also for robotics and deep learning.

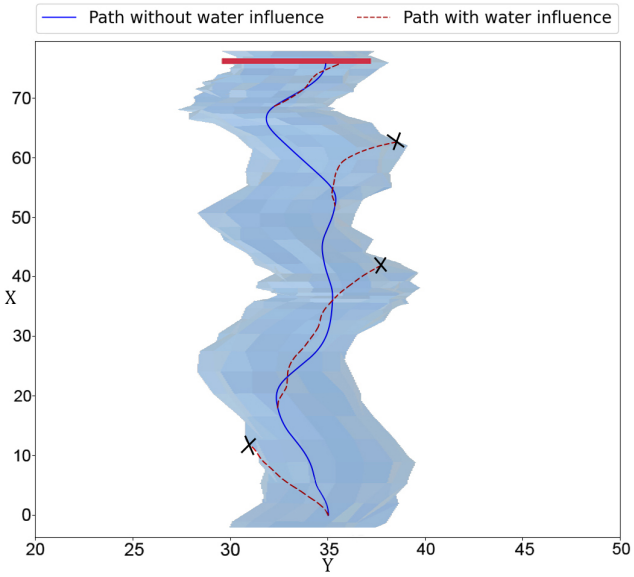

Figure 4: Illustration of the influence that water exerts on the AUV as it attempts to follow an ideal path.

Impact of Water Physics We now delve into the influence of the water in our aquatic environment, which represents the critical challenge for our learning agent. In particular, we consider an underwater navigation scenario where a rover must safely navigate in an underwater cave. Understanding this detail is crucial for quantifying the level of unpredictability in the underwater environment and, consequently, assessing the challenges the agent faces in making decisions in this non-stationary scenario. To demonstrate how marine currents can influence AUV trajectories we conducted an additional experiment, illustrated in Fig.4. We created an ideal trajectory (blue line) by manually moving an AUV unaffected by water forces. Subsequently, we instructed a second AUV to follow the same sequence of actions as the first, with the additional challenge of marine currents (red dashed line). As shown in the figure, this second rover collided with the cave walls a total of three times. This result underscores the critical importance of generating an intelligent agent capable of dynamically correcting unexpected trajectories that could potentially bring the rover to operate too close to cave walls.

4 Training Approach

In this section, we introduce our deep reinforcement learning pipeline, showing the various strategies we employed to develop a safe and reliable agent that serves as a stable baseline for our benchmarking environment. In the following sections, we discuss different approaches describing their respective strengths and weaknesses. Our comparative analysis has been performed to meet the standard requirements for an empirical DRL evaluation (Henderson et al., 2018); in particular, we report the average reward with the standard deviation from different random initializations for the neural networks (i.e., 10 different random seeds for each set of experiments). In the following sections, we conduct a comparative analysis focusing solely on the underwater cave exploration sub-domain available in our simulator. The choice is motivated by the fact that underwater navigation is more

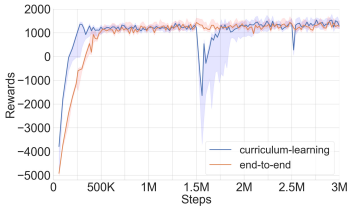

Figure 5: Comparison between curriculum learning and E2E.

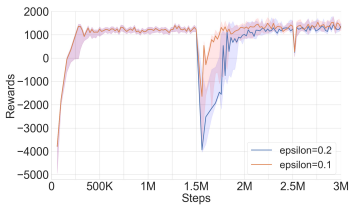

Figure 6: Ablation study on the PPO-clip hyperparameter.

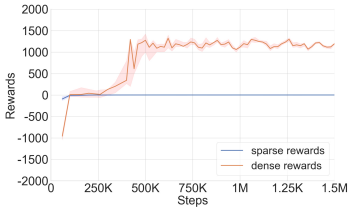

Figure 7: Comparison between sparse and dense reward functions.

complex than surface navigation. Indeed, additional factors such as controlling the diving motion and adjusting pressure based on the depths reached need to be considered. These additional difficulties allow us to conduct a more meaningful analysis. Nevertheless, in Sec. 5 we perform a validation step on the surface navigation problem, confirming our findings.

In both our benchmarks, the goal for the agent is to reach a target destination without colliding with obstacles. Our agent is equipped with 28 sensors arranged in the 180-degree frontal field, allowing it to observe the immediate surrounding environment directly. It is aware of the direction of the target point in a straight line and also knows its own linear and angular velocity. At each time step, the model determines which actions to take by selecting them from a discrete action space, enabling the agent to move forward, rotate, or adjust its depth when submerged. The configuration of sensors and actuators results in a vector observation of 31 real values; while the action space can be tuned by the user and consists of a variable set of discrete actions.

Cave Environments for Training and Testing

The primary objective of our agent is to navigate through a cave and safely reach the target point without colliding with rocks and walls. Crucially the agent is provided with the coordinates of the destination in terms of polar coordinates to its position; this setup is widely adopted in the literature and constitutes a challenging benchmark for the training agent (Ray et al., 2019; Amir et al., 2023a; Marchesini & Farinelli, 2021). Achieving this goal demands high capabilities in obstacle avoidance and the ability to counter the unpredictable movements induced by water currents. To comprehensively evaluate our model's performance, we have designed various cave models. Some of these caves serve for the training phase, while others are used for testing purposes. The idea behind this diversity is to expose the agent to a broad spectrum of scenarios, each posing unique challenges. All the training caves exhibit distinct characteristics that serve as robust evaluative metrics for our model. *The first cave* features larger dimensions compared to the others, moreover, it does not present any additional forces due to the currents of the water. The agent can thus focus completely on the simple control aspect. *The second cave* comprises a sequence of narrow passages interspersed with wider areas. This configuration introduces the additional challenge of navigating through tunnels of varying difficulty. *The third cave* presents a long series of curves, each posing different levels of difficulty; crucially, the parameters related to the velocity of the water particles are raised significantly, posing a significant challenge for the agent. Moreover, in addition to the custom caves for training and testing, in Sec.5 we perform an additional evaluation using a 3D model built from data from a real cave. This test aims to assess the ability of the agent to navigate safely in complex and realistic environments. All these caves and settings are available for testing in our simulation engine. The set of hyperparameters employed for the training is reported in the public repository[3]; these values have been empirically tuned through a grid search process over a set of common configurations.

[3] https://github.com/dadecampo/aquatic_navigation_envs

Curriculum Learning

We start by comparing two distinct training approaches: *curriculum learning (CL)* and *end-to-end (E2E)* training. The E2E approach involves training an Artificial Neural Network (ANN) from start to finish on the entire task, without decomposing it into separate subtasks learned sequentially, this leads to simplifying features and reward engineering. Another improvement we made among the different phases of the curriculum regards the PPO-clip value. Specifically, we reduce the clip value from 0.2 (as recommended in the literature) to 0.1. More details on the approaches and a detailed comparison between the end-to-end approach and our suggested curriculum learning method can be found in Appendix A of the supplementary materials. To summarize, from our experiments, the curriculum learning approach only slightly reduces the convergence time but it does not provide a substantial improvement in performance (see Fig.5). Interestingly, however, the curriculum learning approach results in a significant improvement from a safety perspective (e.g., the number of collisions with rocks); demonstrating a higher generalization capability in previously unseen environments, as can be seen in Sec.5.

Conclusion: For our final experiments we adopted a curriculum learning strategy. According to the literature in the field, our findings suggest that a more structured training pipeline strongly supports a faster and more effective training process (Morad et al., 2021).

Reward Engineering

Reward engineering is a pivotal component of a successful deep reinforcement learning (DRL) process. However, the formulation of effective reward functions is often non-trivial, demanding meticulous consideration of various factors. In Appendix C of the supplementary materials we report a detailed comparison between a sparse and a dense reward function, showing the strengths and the weaknesses of both methodologies. To summarize, the training with sparse rewards did not yield success, with the rover notably failing to reach the final goal (see Fig.7). Conversely, training conducted with dense rewarding has proven to be fruitful, demonstrating the ability to achieve convergence without excessive difficulty.

Conclusion: For our final experiments we employed a dense function; formally the reward at time t is calculated as follows:

$$r_t = \begin{cases} R_{\text{goalReached}} & \text{if the goal is reached} \\ R_{\text{movement}_t} + R_{\text{timestep}} + R_{\text{collision}} & \text{if a collision occurs} \\ R_{\text{movement}_t} + R_{\text{timestep}} & \text{otherwise} \end{cases} \tag{4.1}$$

The parameters within the reward function have been defined through empirical testing and can be found in the appendix C.

The Safety Aspect

Although we obtained promising results with the approaches proposed in this section, we focused only on the pure performance of the agent, while in this section, we consider an additional requirement, the overall safety. In particular, we focus on two aspects: (i) the number of collisions with rocks and (ii) the average distance between the agent and the walls of the cave. More details and results about the safety-oriented reward can be found in Appendix B of the supplementary materials. Our findings demonstrate the positive impact of our explicit safety-centric reward function in terms of reducing the number of collisions and increasing the average safety distance from the cave walls. However, it is worth emphasizing that our approach does not entirely eliminate unsafe behaviors, highlighting the need for future research in this direction.

Conclusion: To enhance the agent's safety, we implemented a reward function that considers the distance from the walls as a cost to minimize during training. Crucially, the final training setup used for our real-world evaluations, reported in Sec. 5, consists of the previous three training techniques in combination.

5 Evaluation in scenarios built from real-world data

In this section, we exploit all the techniques and methodologies discussed throughout this paper to assess the agent's performance within the three-dimensional representation of a real cave. This validation step is made possible through the application of photogrammetry, a technique that enables the creation of highly accurate three-dimensional models using sequences of images or videos. In our simulator, we recreated a detailed portion of Porth Yr Ogof, a cave situated in South Wales (Wilton-Jones, 2023). Porth Yr Ogof is a cave of particular interest due to its unique characteristics. The complex formation of this cave is due to the frequent floods caused by the overflowing of the adjacent Afon Mellte River (the reconstruction of the cave is based on a 3D scan of the area). In this environment, the agent's goal is to reach the end of the cave and thus explore the entire map. What makes the task challenging is that the drone does not have access to the map of the environment, the agent must rely entirely on local observations, exploiting in the decision-making process the policy learned by exploring the caves used for training.

In Fig.8 we show a screenshot from our simulator and a plot of the results obtained by our trained agent; for the experimental evaluation, we deploy our agent multiple times starting from a random position of the cave and collecting the average success rate (i.e., the number of time the AUV manages to exit the caves normalized by the number of experiments). We note that, for this evaluation phase, in order to highlight when safety constraints are violated, we considered a single collision as a complete failure.

Extention to Surface Navigation In the last few sections, we have focused on underwater cave exploration because our preliminary experiments have shown it to be the most challenging environment. However, surface navigation presents similar interesting challenges due to the non-stationary and dynamic nature of water. In this last evaluation, we repeat the analysis of the previous problem in this second benchmark. In this environment, the goal of the agent is to reach the target position while avoiding collisions with rocks and reefs. The agent can only rely on observations from local sensors, which include a GPS and compass for computing heading and distance to the target position, and a proximity sensor for detecting obstacles in specific directions. At the initialization of each episode, the map and the positions of the target and agent are randomly generated. Our results are shown in Fig.9, although overall simpler, our results confirm that even in this scenario PPO struggles to find an optimal policy, especially from the safety perspective.

6 Conclusion

This paper has presented a challenging benchmark to stimulate the advancement of DRL methods for robot control. Our contributions span three key areas: i) we developed a realistic simulator tailored to the unique challenges of underwater cave exploration and surface navigation; ii) we provided a

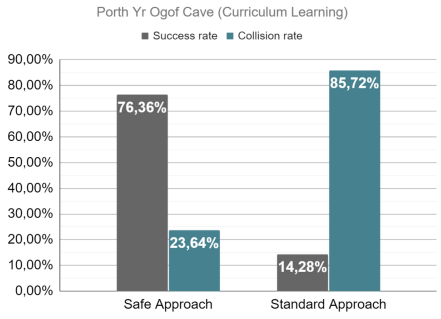

Figure 8: On the left is a screenshot of the Porth Yr Ogof cave; on the right are the results obtained by the trained agent.

Figure 9: On the left are the results obtained by the trained agent; on the right: a screenshot of the surface navigation benchmark.

comprehensive pipeline for training autonomous agents using Deep Reinforcement Learning (DRL); iii) we addressed safety through two critical aspects: collision avoidance and maintaining a safe distance from cave walls. To demonstrate the effectiveness of our approach, we finally conducted an extensive testing phase in a simulation of the real-world cave environment of "Porth yr Ogof" in South Wales where our trained agent successfully explored the cave, avoiding catastrophic collisions with rocks and maintaining a safe distance from cave walls. These contributions together serve to introduce a novel benchmark for deep reinforcement learning in a challenging and realistic scenario and a series of techniques to provide a stable and reproducible result that provides an initial baseline for future development.

We believe this work paves the way for several future directions, including the exploration of alternative approaches to ensure the safety of our trained agents, such as shielding (Alshiekh et al., 2018), constrained reinforcement learning (Achiam et al., 2017), explainability (Bassan et al., 2023), and formal verification tools (Marzari et al., 2023; Corsi et al., 2021; Katz et al., 2019; Amir et al., 2021). Additionally, a natural direction is to move from the simulated environment to real robotic platforms so to gain insights into how the agent interacts with the environment in the real world with the idea of enhancing our simulator. In this last direction, many new challenges arise, such as generalization to unseen situations (Amir et al., 2023b) and possible delays in the communication between the drone and the controller (Karamzade et al., 2024). Crucially, our simulation tool is freely available for future research and collaborations.

Acknowledgments

The work was carried out within the Interconnected Nord-Est Innovation Ecosystem (iNEST) and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.5 – D.D. 1058 23/06/2022, ECS00000043).

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization, 2017.

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand, 2019.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding, 2018.

Guy Amir, Michael Schapira, and Guy Katz. Towards scalable verification of deep reinforcement learning. In *2021 formal methods in computer aided design (FMCAD)*, 2021.

Guy Amir, Davide Corsi, Raz Yerushalmi, Luca Marzari, David Harel, Alessandro Farinelli, and Guy Katz. Verifying learning-based robotic navigation systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 607–627, Paris, 2023a. Springer.

Guy Amir, Osher Maayan, Tom Zelazny, Guy Katz, and Michael Schapira. Verifying generalization in deep learning. In *International Conference on Computer Aided Verification*, 2023b.

Michel Aractingi, Pierre-Alexandre Léziart, Thomas Flayols, Julien Perez, Tomi Silander, and Philippe Souères. Controlling the solo12 quadruped robot with deep reinforcement learning, 2023.

Aleks Attanasio, Bruno Scaglioni, Matteo Leonetti, Alejandro F Frangi, William Cross, Chandra Shekhar Biyani, and Pietro Valdastri. Autonomous Tissue Retraction in Robotic Assisted Minimally Invasive Surgery - A Feasibility Study, 2020.

Ahmad Taher Azar, Muhammad Zeeshan Sardar, Saim Ahmed, Aboul Ella Hassanien, and Nashwa Ahmad Kamal. Autonomous robot navigation and exploration using deep reinforcement learning with gazebo and ros, 2023.

Shahaf Bassan, Guy Amir, Davide Corsi, Idan Refaeli, and Guy Katz. Formally explaining neural networks within reactive systems. In *2023 Formal Methods in Computer-Aided Design (FMCAD)*, 2023.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to End Learning for Self-Driving Cars, 2016. Technical Report. http://arxiv.org/abs/1604.07316.

Marc Carreras, Juan David Hernández, Eduard Vidal, Narcís Palomeras, David Ribas, and Pere Ridao. Sparus ii auv—a hovering vehicle for seabed inspection, 2018.

Patryk Cieślak. Stonefish: An advanced open-source simulation tool designed for marine robotics, with a ros interface. *OCEANS 2019 - Marseille*, pp. 1–6, 2019. URL https://api.semanticscholar.org/CorpusID:204701708.

Davide Corsi, Enrico Marchesini, and Alessandro Farinelli. Formal verification of neural networks for safety-critical tasks in deep reinforcement learning. In *Uncertainty in Artificial Intelligence*, 2021.

Davide Corsi, Guy Amir, Guy Katz, and Alessandro Farinelli. Analyzing adversarial inputs in deep reinforcement learning. *arXiv preprint arXiv:2402.05284*, 2024.

Mirella Santos Pessoa De Melo, José Gomes da Silva Neto, Pedro Jorge Lima Da Silva, João Marcelo Xavier Natario Teixeira, and Veronica Teichrieb. Analysis and comparison of robotics 3d simulators, 2019.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo, 2020.

Thor Fossen. Handmisc of marine craft hydrodynamics and motion control, 2011.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018.

Kelam Goutam, S. Balasubramanian, Darshan Gera, and R. R. Sarma. Layerout: Freezing layers in deep neural networks, 2020.

Sven Gronauer. Bullet-safety-gym: A framework for constrained reinforcement learning, 2022.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications, 2018b.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters, 2018.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning, 2018.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling, 2018.

Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping, 2020.

Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents, 2020.

Armin Karamzade, Kyungmin Kim, Montek Kalsi, and Roy Fox. Reinforcement learning from delayed observations via world models. *arXiv preprint arXiv:2403.12309*, 2024.

Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, et al. The marabou framework for verification and analysis of deep neural networks, 2019.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey, 2013.

Jean-Claude Latombe. *Robot motion planning*. Springer Science & Business Media, 2012.

SM LaValle. Planning algorithms. *Cambridge University Press google schola*, 2006.

Litian Liang, Yaosheng Xu, Stephen McAleer, Dailin Hu, Alexander Ihler, Pieter Abbeel, and Roy Fox. Reducing variance in temporal-difference value estimation via ensemble of deep networks. In *International Conference on Machine Learning*, 2022.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2015.

Ivan Lončar, Juraj Obradović, Natko Kraševac, Luka Mandić, Igor Kvasić, Fausto Ferreira, Vladimir Slošić, Đula Nađ, and Nikola Mišković. Marus - a marine robotics simulator. In *OCEANS 2022, Hampton Roads*, pp. 1–7, 2022. doi: 10.1109/OCEANS47191.2022.9976969.

Enrico Marchesini and Christopher Amato. Improving deep policy gradients with value function search. *International Conference on Learning Representations, ICLR*, 2023.

Enrico Marchesini and Alessandro Farinelli. Centralizing state-values in dueling networks for multi-robot reinforcement learning mapless navigation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4583–4588, Prague, 2021. IEEE.

Enrico Marchesini and Alessandro Farinelli. Enhancing deep reinforcement learning approaches for multi-robot navigation via single-robot evolutionary policy search. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 5525–5531, Philadelphia, 2022. IEEE.

Enrico Marchesini, Davide Corsi, and Alessandro Farinelli. Benchmarking safe deep reinforcement learning in aquatic navigation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5590–5595, Prague, 2021. IEEE.

Luca Marzari, Davide Corsi, Ferdinando Cicalese, and Alessandro Farinelli. The #dnn-verification problem: Counting unsafe inputs for deep neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, 2013. Technical Report. https://arxiv.org/abs/1312.5602.

Steven D Morad, Roberto Mecca, Rudra PK Poudel, Stephan Liwicki, and Roberto Cipolla. Embodied visual navigation with automatic curriculum learning in real environments. *IEEE Robotics and Automation Letters*, 2021.

Lucas Nogueira. Comparative analysis between gazebo and v-rep robotic simulators, 2014.

Xinlei Pan, Yurong You, Ziyan Wang, and Cewu Lu. Virtual to real reinforcement learning for autonomous driving, 2017.

Ameya Pore, Davide Corsi, Enrico Marchesini, Diego Dall'Alba, Alicia Casals, Alessandro Farinelli, and Paolo Fiorini. Safe reinforcement learning using formal verification for tissue retraction in autonomous robotic-assisted surgery. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning, 2019.

Benjamin Rolf, Ilya Jackson, Marcel Müller, Sebastian Lang, Tobias Reggelin, and Dmitry Ivanov. A review on reinforcement learning algorithms and applications in supply chain management, 2023.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017. Technical Report. http://arxiv.org/abs/1707.06347.

Thiago Simão, Nils Jansen, and Matthijs Spaan. Alwayssafe: Reinforcement learning without safety constraint violations during training, 2021.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods, 2020.

Unity Technologies. Unity robotics hub, 2020. URL https://github.com/Unity-Technologies/Unity-Robotics-Hub.

Mark Wilton-Jones. Porth yr ogof - uk caves database, 2023. URL http://www.ukcaves.co.uk/cave-porthyrogof.

Russell B Wynn, Veerle AI Huvenne, Timothy P Le Bas, Bramley J Murton, Douglas P Connelly, Brian J Bett, Henry A Ruhl, Kirsty J Morris, Jeffrey Peakall, Daniel R Parsons, et al. Autonomous underwater vehicles (auvs): Their past, present and future contributions to the advancement of marine geoscience, 2014.

Raz Yerushalmi, Guy Amir, Achiya Elyasaf, David Harel, Guy Katz, and Assaf Marron. Scenario-assisted deep reinforcement learning. In *10th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2022*, pp. 310–319. Science and Technology Publications, Lda, 2022.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey, 2020.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning, 2017.

ZibraAI. Zibraai and its ml-powered toolset to boost the game development industry, 2021.

A Curriculum Learning

We start by comparing two distinct training approaches: *curriculum learning (CL)* and *end-to-end (E2E)* training.

The E2E approach involves training an Artificial Neural Network (ANN) from start to finish on the entire task, without decomposing it into separate subtasks learned sequentially. End-to-end training offers the advantage of simplifying feature and reward engineering and reducing the need for manually designing intermediate steps. However, it often necessitates a substantial amount of training data and can pose challenges in terms of interpretability. Curriculum learning, on the other hand, consists of training the agent in a sequence of problems, typically of increasing difficulty; this enables the agent to leverage knowledge and skills acquired in simpler tasks to enhance learning and performance when facing the complete problem. Our curriculum learning pipeline starts with the simplest environment, denoted as *Cave_Train1* (Fig.10), where the agent learns a set of fundamental skills such as keeping a safe distance from walls and navigating toward the target while executing gentle turns. Subsequently, the weights learned in the initial task are transferred to the new neural network as we progress to the *Cave_Train2* environment. Here, the agent exploits the already learned capabilities in a more intricate environment. Additionally, the rover encounters water currents for the first time, albeit at half the strength intended for the evaluation phase. The third and final phase of our training process introduces the rover to water currents at the intended evaluation strength. This phase can be regarded as a refinement stage, building upon the model developed during the initial two phases. To prevent any loss of knowledge acquired throughout the entire training process, we lowered the learning rate for this final step.

Another improvement we made among the different phases of the curriculum regards the PPO-clip value. Specifically, we reduce the clip value from 0.2 (as recommended in the literature) to 0.1. The intuition behind this modification is that during the initial *Cave_Train1* phase, the rover learns fundamental movement policies, which we aim to preserve as the training progresses to the more complex stages. By lowering the clip value, we aim to maintain policy stability. However, reducing this parameter requires a tradeoff that lies in a loss in data efficiency (i.e., slowing the acquisition of new behaviors). Typically, knowledge transfer across phases is accompanied by the freezing of layers trained in the previous stage. However, given that the fundamental objective of our agent remains unchanged despite the changing environments, we opted to avoid this technique, allowing for continuous learning and adaptation throughout the curriculum Goutam et al. (2020).

Results:

Fig.5 presents the results of our comparison between the curriculum learning and the E2E approach. Noticeable drops in rewards correspond to the transitions between lessons, occurring at the 1.5 million and 2.5 million timestep marks. However, in both cases, the reward graphs quickly recover and converge to approximately 1200. To ensure a fair comparison, we assign to E2E training the cumulative number of timesteps across all phases of curriculum learning required to achieve a satisfactory result in the analyzed environment.

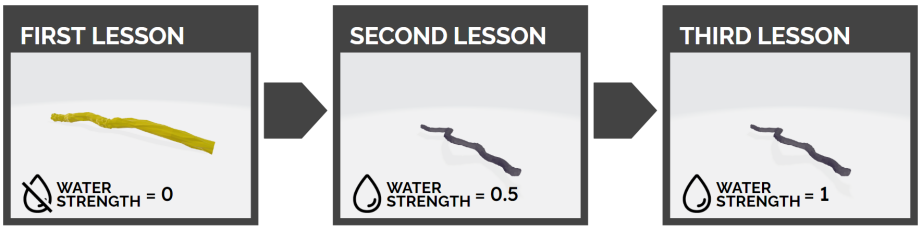

Figure 10: Curriculum Learning lessons plan. From left to right the pictures represent the training caves ordered by growing difficult.

The curriculum learning approach slightly reduces the convergence time required but it does not provide a substantial improvement in performance. Interestingly, however, the curriculum learning approach results in a significant improvement from a safety perspective (e.g., number of collisions with rocks); demonstrating a higher generalization capability (Sec.B provides more detail about this result). Moreover, Fig.6 shows an ablation study to motivate our choice of using a smaller PPO-clip value with respect to the standard setting; our results clearly show the obtained improvements, confirming our design choice.

B The Safety Aspect

Although we obtained promising results with the approaches proposed in Sec. 4, we focused only on the pure performance of the agent, while in this section, we consider an additional requirement, the overall safety. In particular, we focus on two aspects: (i) the number of collisions with rocks and (ii) the average safe distance between the agent and the walls of the cave. In the literature, numerous approaches exist to improve the safety of a learning agent, such as Constrained Deep Reinforcement Learning Stooke et al. (2020), Safe Exploration Simão et al. (2021), Shielding Alshiekh et al. (2018), and Formal Verification Katz et al. (2019); however, these techniques go beyond the benchmarking scope of this work, and we leave the analysis of these approaches for future research. In contrast, in this paper, we focus on the concept of "reward engineering". In particular, we propose to modify the reward function presented in Sec. 4, refining it to encourage more cautious behaviors. Through this rewarding mechanism, we emphasize the role of the proximity sensors, treating them not only as part of the observation space but also as key components in the calculation of the reward function. By doing so, we expect our agent to exhibit safer behavior, actively attempting to maintain a safe distance from the cave walls.

$$r_t = \begin{cases} R_{\text{goalReached}} & \text{goal} \\ R_{\text{movement}_t} + R_{\text{timestep}} + R_{\text{collision}} + R_{\text{sensors}} & \text{collision} \\ R_{\text{movement}_t} + R_{\text{timestep}} + R_{\text{sensors}} & \text{otherwise} \end{cases}$$

The term R_{sensors} is calculated based on the measurements taken at each timestep by the rover's sensors. When contact occurs through the lidar sensor, a value is returned indicating the height at which the ray was intercepted. This value is normalized to 1, and then, by using $-(1 - \text{rayInterceptionValue})$, subsequently multiplied by a constant. This operation is repeated for each of the 28 rays at every single timestep. The way this reward has been adjusted, based on the multiplication by the chosen constants, limits the range of R_{sensors} to $(-0.6, 0.0]$. Figure 11 provides a visual explanation of the lidar sensor readings of our robot.

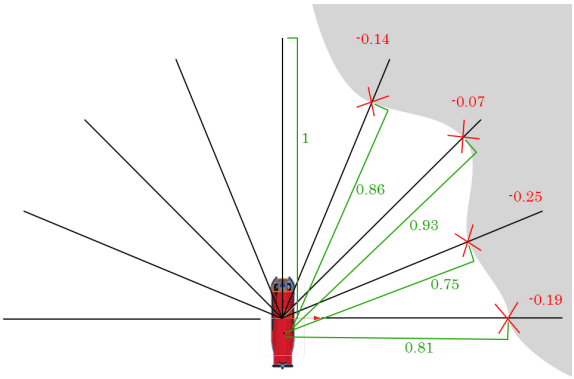

Figure 11: This image illustrates the concept of "interceptionValue", highlighted in green, while the penalties imposed on the agent are indicated in red.

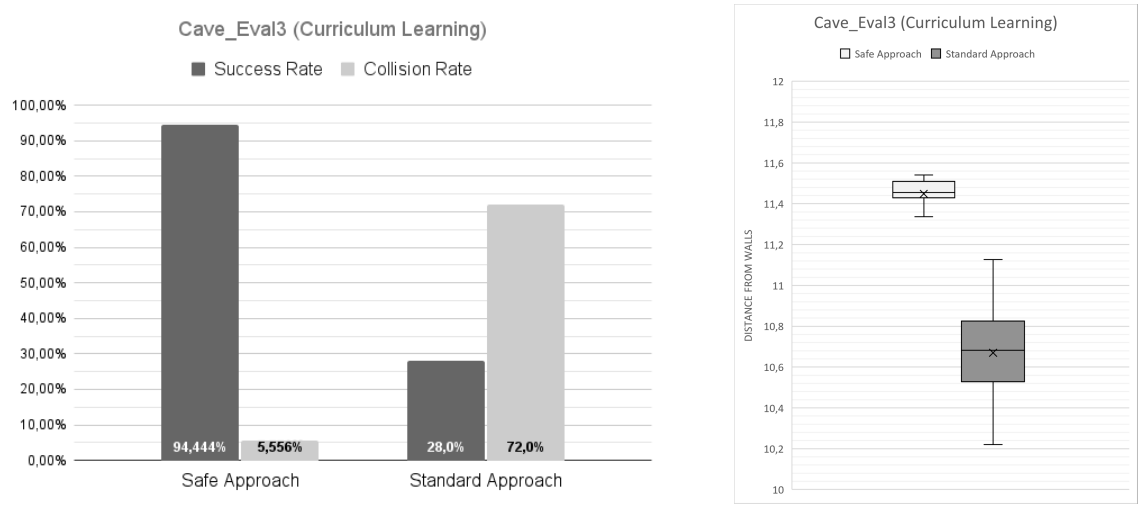

Figure 12: Analysis of the agent's performance from a safety perspective.

Results:

In this section, we delve into the results of our safety-oriented analysis by focusing on the results obtained in the most challenging scenario, referred to as *"cave_test3"*. Results are presented in Figure 12. Our findings demonstrate the positive impact of our safety-centric reward function in terms of reducing the number of collisions and increasing the average safety distance from the cave walls. However, it is worth emphasizing that our approach does not entirely eliminate unsafe behaviors, highlighting the need for future research in this direction.

C Reward Engineering

Reward engineering is a pivotal component of a successful deep reinforcement learning (DRL) process. However, the formulation of effective reward functions is often non-trivial, demanding meticulous consideration of various factors. For example, an important challenge revolves around achieving the delicate balance between shaping the agent's behavior and avoiding inadvertent side effects. An excessively simplistic or sparse reward may prevent an effective learning process. Conversely, an overly intricate function may result in an agent incapable of generalizing beyond the training environment Hu et al. (2020). Another well-known problem pertains to reward hacking, i.e., whereby an agent exploits loopholes or biases in the reward function to maximize rewards without genuinely fulfilling the intended task. This underscores the imperative need for crafting reward functions that are both informative and resilient to manipulation. In literature, reward functions are typically subdivided into two main categories: sparse and dense. In this section, we conduct an analysis comparing the efficacy of these two reward paradigms applied to our case study.

Sparse Rewards

Sparse rewards consist of a structure where the agent receives a reward signal only upon accomplishing specific operations (e.g., avoiding an obstacle) or achieving particular objectives (e.g., reaching a target position). However, for the majority of the learning process, the agent encounters limited to no feedback. Sparse rewards present notable challenges for RL agents due to their nature, offering minimal guidance during the learning process. For example, an agent may struggle to understand which actions or states contribute to their success or failure, given the sporadic feedback. Consequently, this can lead to a slow learning loop or hindered convergence, particularly in complex

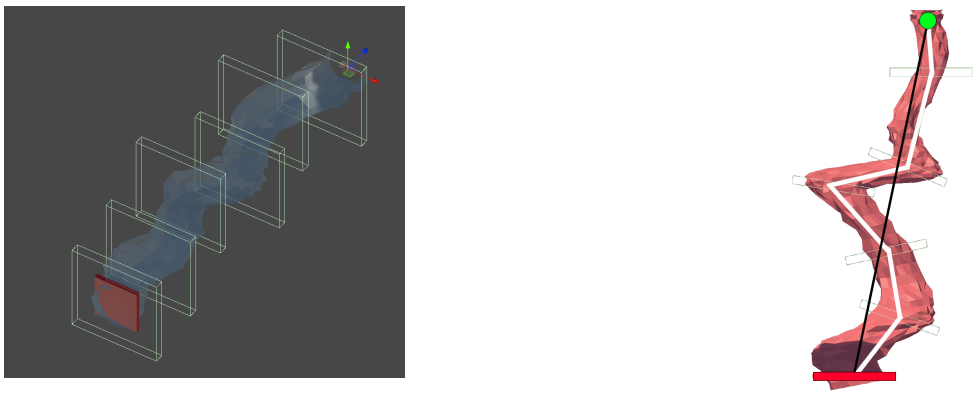

Figure 13: Comparison between *DistanceRewarder* (white) and Euclidean distance (black line).

environments. Despite these limitations, sparse rewards are easy to design and are more robust against reward hacking.

We now introduce the discrete reward function we employed in our training process. This is a straightforward reward system that grants rewards to the rover under specific conditions while imposing penalties for undesirable outcomes (e.g., collisions with rocks or cave walls).

$$r_t = \begin{cases} R_{\text{goalReached}} & \text{if the goal is reached} \\ R_{\text{collision}} & \text{if the rover collides} \\ R_{\text{fail}} & \text{timeout} \end{cases} \tag{C.1}$$

In this reward function, $R_{\text{goalReached}}$ represents the reward granted when the rover successfully reaches its goal, $R_{\text{collision}}$ signifies the penalty for collisions, and R_{fail} denotes the penalty imposed if the rover fails to achieve the goal within the stipulated time frame. In our setup, we impose $R_{\text{goalReached}} = 10$, $R_{\text{collision}} = -10$, and $R_{\text{fail}} = -1$.

Dense Rewards

Dense rewards represent an alternative structure where the agent receives a signal more frequently, typically at each time step of an episode. These rewards exhibit a higher level of continuity and furnish constant feedback to the agent as it progresses through the environment. We now introduce the continuous function we proposed in our work. This reward incentivizes movement toward the cave exit while penalizing proximity to cave walls. During training, an accurate calculation of the distance from the goal is crucial, and to achieve this, we introduce a system of panels that trace the tunnel's curvature (Fig. 13). By measuring the distance between these panels, along with the distance from the last panel to the rover, we obtain a more precise approximation of the distance from the goal. This system, termed "DistanceRewarder", is a specific and fundamental component we developed to furnish a more precise reward to the agent.

Moreover, the reward function we employed for our standard training includes multiple additional components designed to incentivize specific behaviors. First, a specific signal that encourages the rover to progress along the path toward the goal. At each timestep, the agent is rewarded with a value proportional to the distance traveled toward the cave exit. It's important to note that moving away from the goal results in a penalty. Second, to promote behavior that takes the environment into account, a penalty is applied when the agent collides with a cave wall. The third component of the function is a reward bonus, which is obtained upon reaching the cave exit. During training, the reward at time t is calculated as follows:

$$r_t = \begin{cases} R_{\text{goalReached}} & \text{if the goal is reached} \\ R_{\text{movement}_t} + R_{\text{timestep}} + R_{\text{collision}} & \text{if a collision occurs} \\ R_{\text{movement}_t} + R_{\text{timestep}} & \text{otherwise} \end{cases} \tag{C.2}$$

For our experiment we set $R_{goalReached} = 500$, $R_{movement_t} = (distanceToGoal_{t-1} - distanceToGoal_t) * 10$, $R_{timestep} = -0.01$ and $R_{collision} = -0.01$. $R_{movement_t}$ rewards the displacement of the rover in the direction of the goal. In our environment, the combination of rover speed and water friction puts this reward in a range from -0.03 to 0.03, so we can conclude $R_{movement_t} \in (-0.03, 0.03)$.

Results:

We now provide a comparative analysis of dense and sparse rewarding approaches. As depicted in Fig.7, training with sparse rewards did not yield success, with the rover notably failing to reach the final goal. Conversely, training conducted with dense rewarding has proven to be fruitful, demonstrating the ability to achieve convergence without excessive difficulty. In the next section, we will focus on the safety aspects, showing how the reward engineering process is crucial also for this latter aspect.