Inteligencia Artificial 27(73), 163-176
doi: 10.4114/intartif.vol27iss73pp163-176

INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Investigating the Impact of Curriculum Learning on Reinforcement Learning for Improved Navigational Capabilities in Mobile Robots

Alaa Iskandar[1], Béla Kovács[2]

[1] PhD Student, University of Miskolc, Institute of Mathematics 3515 Miskolc, Miskolc-Egyetemváros, e-mail: iskandar.alaa@student.uni-miskolc.hu
[2]Associate professor, University of Miskolc, Institute of Mathematics 3515 Miskolc, Miskolc-Egyetemváros, e-mail: bela.kovacs@uni-miskolc.hu

Abstract This paper proposes a method for finding the shortest path of a mobile robot using deep reinforcement learning with utilizing Proximal policy optimization algorithm (PPO) enhanced with curriculum learning. By modelling the environment in 3D space using the Webots simulator, we extend the PPO algorithm's capabilities to handle continuous states from 8 IR sensors and control the velocities of two motors of E-puck robot. Our study uniquely integrates curriculum learning into the PPO framework, aiming to improve adaptability and training efficiency in complex environments. A comparative analysis is conducted between the modified PPO, the original PPO, and the deep deterministic policy gradient algorithm, highlighting the strengths of our approach The results demonstrate that our curriculum-augmented PPO algorithm not only accelerates the training process but also shows superior adaptability and generalization in new environments. This work underscores the significant potential of curriculum learning in enhancing the performance of deep reinforcement learning algorithms for robust and efficient robotic navigation.

Keywords: Reinforcement learning, Proximal Policy Optimization, Path Planning, E-puck robot, Webots simulator, Deep deterministic policy gradient algorithm, Curriculum learning.

1 Introduction

Path planning for mobile robots is an area of active research and it presents a significant challenge to researchers seeking to develop optimal algorithms, tools, and equipment for finding safe and efficient paths between two points. This task involves computing the path from a starting point to a target point while considering the avoidance of obstacles and optimizing parameters such as distance, time, energy consumption, and others [1].
Path planning algorithms are categorized based on a robot's environmental knowledge. If a robot has full knowledge of its environment, it can construct a map and compute the optimal path to reach its target using algorithms such as A*, Dijkstra, or artificial potential field. On the other hand, algorithms like random rapidly exploring tree (RRT), fuzzy logic, and reinforcement learning (RL) are better suited for situations where the robot has little or no prior knowledge of the environment, and need to explore and learn the environment in order to find an optimal path [2].
RRT is designed to quickly explore the space and build a tree structure of possible paths while RL is designed to learn a policy for making decisions in a complex and uncertain environment through trial and error. RRT is suitable for solving path planning problems in high-dimensional spaces and dynamic environments due to its ability to rapidly acknowledge the space and find a feasible path. However, RRT may not always find the optimal path and can be sensitive to the initial conditions and obstacles in the environment [2].
Fuzzy logic is a rule-based approach that can handle imprecise and uncertain information in the environment making it suitable for path planning in uncertain and dynamic environments. However, fuzzy logic may not be able to learn and adapt to the environment as well as RL and its performance may depend on the accuracy of the predefined rules

[3]. RL is a powerful approach for path planning that can learn from experience and adapt to the environment over time [4].

RL may require significant computational resources and a large amount of data to achieve high performance [5]. Therefore, the choice between RRT, fuzzy logic, and RL for path planning depends on the specific requirements of the problem and the available resources.

In the field of machine learning, the concept of curriculum learning (CL) was introduced by [6], highlighting the importance of organizing training examples in an orderly manner, from simpler to more complex tasks. This methodology has been shown to substantially improve generalization and speed up training, particularly in non-convex training scenarios. In path planning, CL is indispensable for systematically enhancing a robot's navigation skills, thus ensuring greater adaptability and efficiency in varied and complex environments.

This approach tackles the integration challenges of various perception tasks such as mapping and localization with optimal path-planning and control [7]. A notable example of this application is the study by [8], which employed RL for two-dimensional path planning and obstacle avoidance in unmanned aerial vehicles. By using CL in a simulated environment, the study notably improved learning efficiency, achieving maximum goal rates of 71.2% and 88.0% with two different reward models.

In this paper, we utilize PPO considered one of the most popular policy-based methods of RL algorithms. It is a promising approach for path planning due to its ability to handle complex environments, explore effectively, and incorporate continuous action spaces. PPO is used in this study with CL which is a technique that can overcome some of challenges such as adapting with complex environments. In addition, it improves the performance of PPO by providing a better initialization for the policy and enabling the robot to learn more effectively.

Webots simulator was used to construct a 3D environment and the Open AI gym with Deepbots-Python libraries to adapt the RL algorithm with the simulator. The paper is structured as follows: introducing RL and its applications in robotics for path planning focusing on PPO, applying PPO with CL approach, analyzing the results and comparing with DDPG algorithm and original PPO, and finally concluding the results with recommendations for further research.

2 Reinforcement Learning

RL is a decision-making approach related to machine learning. It is based on two main components; the agent and the environment where the agent learns by interacting with it. Firstly, it explores the environment to build knowledge about it and then exploits it to solve the given problem by choosing the optimal actions according to the current and next states.

Any RL problem is formulated as:

- State space S: Environment is divided into many states. The agent executes an action to move from one state S_t to another S_{t+1}. For example, distance and light sensors reading can be grouped in a frame each time step as states for a robot interacting with the environment
- Action space A: It includes the actions that the agent can perform, like turning right or left for a robot. The actions might be discrete or continuous.
- Reward function R: According to the behavior produced by the agent, actions are evaluated using the reward function as feedback. Good ones are rewarded, and bad ones are punished. As a result, RL aims to learn a policy that maps each state to the best action by maximizing the received cumulative rewards.

In many problems, different situations call for different actions and the chosen actions affect the future reward. These two aspects are considered by formulating the problem as a Markov Decision Process MDP [9]. The diagram in figure 1 illustrates the RL problem as MDP.

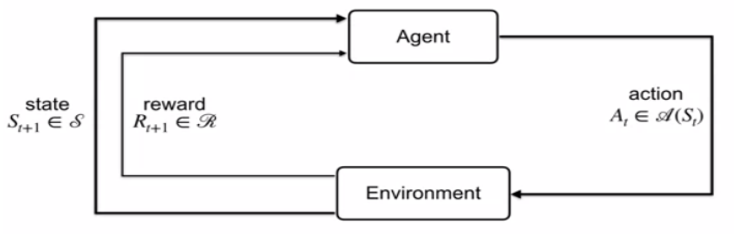
Agent
state $S_{t+1} \in \mathcal{S}$
reward $R_{t+1} \in \mathcal{R}$
action $A_t \in \mathcal{A}(S_t)$
Environment

Figure 1. Representing RL approach as Markov decision process.

RL aims to maximize the future reward received each time. The cumulative received rewards are called Return G, and because of the dynamics of MDP, it is considered a random variable. It represents an essential value to construct the equations used to find the optimal policy called function values in equation 1. There are many approaches to make the agent learn how to make decisions according to Return. It is achieved by computing the function values and choosing the corresponding actions as in equation 2, [9].

$$Q(s,a) = E[G_t | S_t = s, A_t = a] \tag{1}$$

$$a^* = \underset{a \in A}{\operatorname{argmax}}\, Q(s,a) \tag{2}$$

Q (s, a): State- Action value
G_t: Return at time t, S_t: state at time t, A_t: Action at time t
a^*: Best action
Generally, the used approaches are model-based like Dyna or model-free like Q-learning, SARSA, expected SARSA, and others [10]. RL methods have become more efficient, and less computations by incorporating it with deep learning techniques [11].
Moreover, deep learning introduces new methods for RL. Instead of depending on function values to obtain the optimal policy which called value-based methods, neural network is employed to learn the optimal policy directly called policy-based methods, such as Soft Actor-Critic (SAC), Trust Region Policy Optimization (TRPO) [12], Deep Deterministic Policy Gradient (DDPG) [13]. For instance, DDPG is an actor-critic algorithm designed for continuous action spaces. It learns an actor network to select actions and a critic network to estimate the value of the selected actions. Proximal Policy Optimization (PPO) is a family of on-policy. It is also based on actor critic and policy gradient methods that optimize the objective function by clipping the policy update. It is designed to be computationally efficient and stable [14]. Figure 2 illustrates the main diagram of RL algorithms.

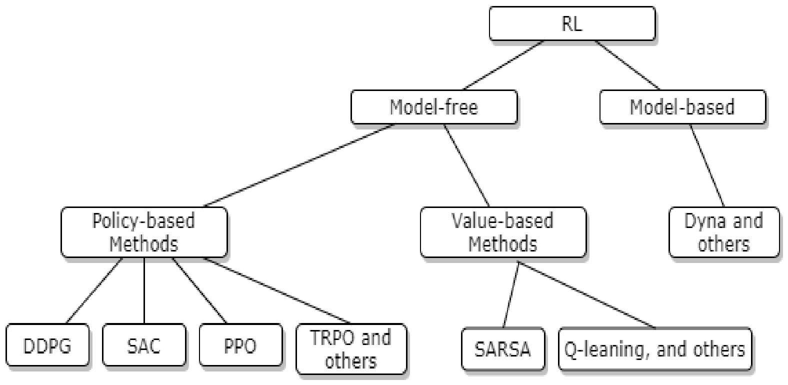

Figure 2. RL algorithms diagram.

3 Reinforcement learning for path planning of mobile robots

RL has shown a promise approach in path planning for mobile robots especially in complex and dynamic environments due to its ability to deal with uncertainty and no knowledge about the working place. RL algorithms offer several advantages for path planning of mobile robots including adaptability, optimization, flexibility, and potential for autonomy [15]. These advantages make RL an encouraging direction for developing efficient and effective path planning solutions for mobile robots in various real-world scenarios [16].
Researchers are actively developing new RL algorithms that are more sample-efficient, generalizable, safe, robust, and less computationally demands. These factors represent the main challenges for this field. Many improvements have been introduced for RL to work effectively in navigations of mobile robotics by proposing solutions to deal with huge amount of data such as continuous state and action spaces. As it was shown in [17], the researcher used DQN with conventional neural network to adapt the color images as an input so the robot becomes able to avoid the obstacles successfully and reach to the target. Another study [18] depended on discretization of the state space to finite states for representing the reading of distance sensors with shaping rewards. Thus, RL algorithm was able to solve the problem and showed the ability of self-learning. An improved DQN method with reward function combined with artificial potential field is used to speed up finding the path. It increases the reward and, accordingly, convergence becomes faster by comparing with traditional DQN [19].
A double DQN was proposed in [20] to avoid collision with obstacles and find the path to target. All previous mentioned studies have used value-based methods. However, policy-based methods are more preferable because of

their ability to deal with continuous state and action spaces. Moreover, they map directly between action and state instead of choosing the actions according to the function values in value-based methods. For example, DDPG is used to find the path of a robot in different environments and compare it with DDQN to prove that it is faster than it in convergence [21]. There are many other studies depended on PPO algorithms to find the optimal policy which enable the robot to follow the optimal path. It is less computational and more stable by comparing to many other policy-based algorithms. It is used surrogate objective function optimized by using gradient decent. As a result, it needs less computations by comparing to TRPO [14]. PPO updates the policy in multiple epochs using the same data with each epoch. This allows the algorithm to learn from the same data multiple times which helps to reduce the impact of outliers or noise in the data. It is considered one of many famous algorithms in the applications that are based on path planning of mobile robots. Table 1 illustrates some of the recent studies.

Table 1: Applications of PPO in robot's path planning.

Research	Application	Environment	Contributions	Validation
Author [22]	Path planning of cleaning robot	2D Tkinter, Python Discrete, Simple 20×20 states	Reward shaping Transfer learning	PPO, Dueling DQN, and DQN
Author [23]	Path planning of Indoor mobile robot	2D Pygame, Python Discrete, Simple 400×400 states Static and dynamic obstacles	Improved reward function by using static-dynamic normalization and priority replay buffer	SAC, PPO
Author [15]	Path planning of Indoor mobile robot	3D with 13×13 m^2 area Gazebo simulator with ROS	Using probabilistic Roadmap with TD3 as novel path planner	TD3, SAC, PPO, DDPG
Author [24]	Lunar rover	3D Gazebo with ROS to simulate the moon's terrain	End to end path planning system	PPO, DQN, A*
Author [25]	Path planning of Indoor mobile robot	2D-6 simple, static environments and real-world experiment	Combine A* with PPO	PPOA*, DWAQ, DDQNP, DBPQ, EPRQL

Table 1 shows many improvements that can modify PPO, and other RL algorithms like DQN, DDPG. Some of these improvements are related to reward function as in [22],[23], the input/output operation [23], or deep learning techniques like transfer learning [22],[24]. The validation methods depend on comparing the suggested algorithm with other standard or improved algorithms; the best performance should be read from left to right for each cell in validation columns.

In this paper, PPO with CL is used to find the path of robot in four 3D indoor gradual complex environments built by Webots simulator with Deepbots. PPO with CL is also tested by a generalized in environment based on training ones. Shaping reward is employed, and the results are compared to PPO without CL and DDPG.

4 Implementation of path planning of mobile robot based on PPO

The simulation will be held in a 3D environment to be more realistic using the Webots simulator. It supports AI and controls algorithms' development, testing, and validation. It contains a library with many professional models for sensors, actuators, and robots [26]. E-puck is used for path planning. It is a mobile robot developed by GCtronic and EPFL. It is small and equipped with various sensors, figure 3. It is also integrated with Webots which facilitates programming and testing [27]. To code PPO algorithm, Open AI Gym is used [28]. DeepBots framework interfaces Deep RL with Webots. It contains the required libraries and packages to apply abstract deep RL algorithms in the Webots simulator [29].

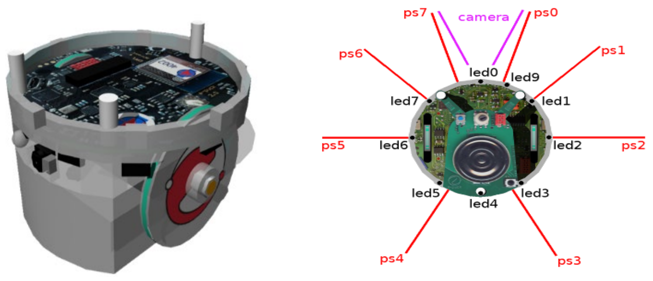

Figure 3. E-puck2 robot [13].

4.1 The environment

In this study, we aim to investigate the effectiveness of using PPO with CL to solve path planning problem in proposed environments; each measuring $0.7 \times 0.7\ m^2$. The first training environment is a simplest one without obstacles showed in figure 4-a, while the second environment is more complex with multiple small boxes and narrow passages in figure 4-b. The third training one is designed to be a long obstacle as in figure 4-c. The final training environment is combined of previous ones as shown in figure 4-d measuring $1 \times 1\ m^2$.

PPO was used for training the robot in each training environment using CL, where gradually the complexity of the environments was increased by introducing new obstacles through every training level. After training, the performance was evaluated on a separate testing environment with area $1 \times 1\ m^2$, figure 4-e, which was designed to has new distribution of obstacles as new area that the agent has not encountered during training taking into consideration the fact that the testing environment is a combination of training environments.

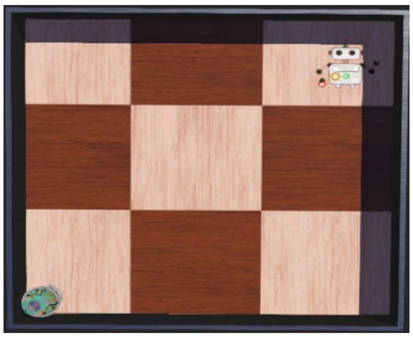
Figure 4-a. Training environment 1 $\mathbf{0.7 \times 0.7\ m^2}$

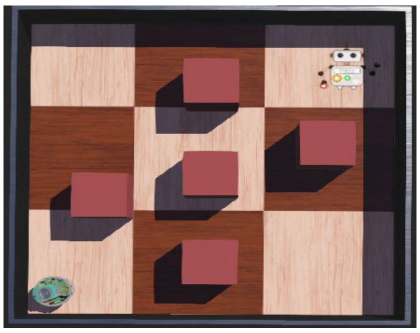
Figure 4-b. Training environment 2 $\mathbf{0.7 \times 0.7\ m^2}$

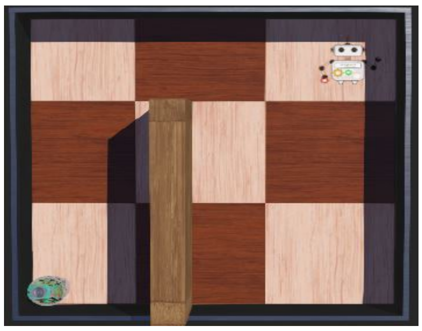
Figure 4-c. Training environment 3 $\mathbf{0.7 \times 0.7\ m^2}$

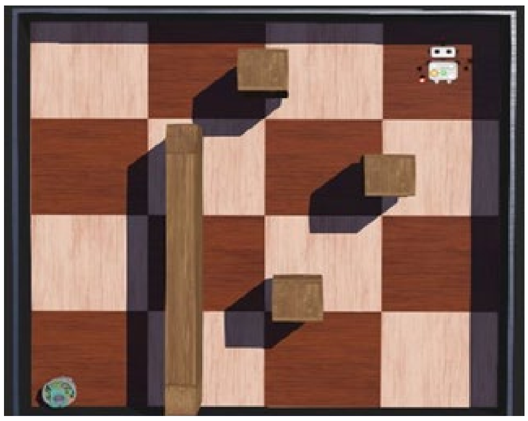
Figure: 4-d. Training environment 3, $\mathbf{1 \times 1\ m^2}$

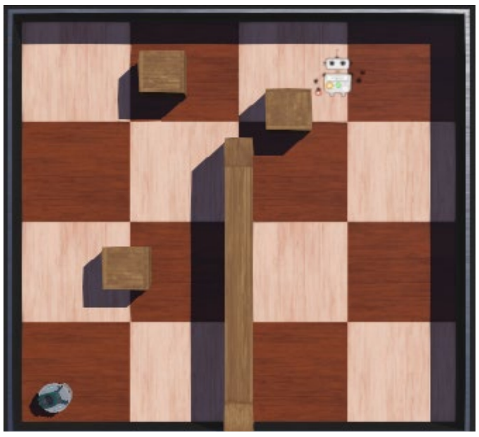
Figure: 4-e. Testing environment, $\mathbf{1 \times 1\ m^2}$

Figure 4. The proposed environments.

4.2 RL algorithm architecture and parameters

1- Observation space: It contains eight readings of IR sensors in addition to the distance between the robot's current location and the angle of the robot's deviation from the target. So, state space at each time t contains ten values to describe the environment.

2- Action space: PPO algorithm deals with continuous action space. The output is two continuous values reflecting the velocities of the robot's motors to go straight or turn.

3- Reward function: Reward shaping is a technique used to enhance the performance of an agent. It involves modifying the reward function of the agent to provide additional incentives that encourage desirable behavior. Sparse rewards where the agent only receives a reward at the end of a long sequence of actions can make learning very difficult. Reward shaping can help overcome this challenge by providing intermediate rewards at each step. It makes the learning process faster and more efficiently.
A shaping reward function is used, as shown in Equation 3, with the penalty $P = -0.001$, if there are collisions with obstacles or the robot deviates from the goal in case of free-obstacles environment. $P = 0$ when there is no collide with obstacles.
The shaping idea comes from comparing the current and previous position of the robot. Accordingly, if it progresses towards to the target, this comparison produces a positive value as a reward. Otherwise, the value is negative as a punishment. It is compared to spares rewards given in equation 4.

$$Penality = \begin{cases} -0.001 & ,If\ there\ are\ obstcales\ in\ the\ range\ of\ IR\ sensors. \\ 0 & ,If\ the\ is\ no\ obstcales\ in\ the\ range\ of\ IR\ sensors. \end{cases} \tag{3}$$
$$Reward\ = prvious\ ditance - current\ \ distance + penality.$$

$$Reward = \begin{cases} -0.001 & ,If\ there\ are\ obstcales\ or\ robot\ fails\ reach\ the\ goal. \\ +0.05 & ,If\ the\ robot\ reachs\ the\ goal. \end{cases} \tag{4}$$

4- D-flag: Binary value is used to terminate the episode. It happens when the maximum length reaches 1500-time steps or when it arrives at the target area.

Table2: The parameters of the PPO algorithm.

Parameter	Value
max training timesteps	1000000
max timesteps per episode	1500
state space dimension	10
action space dimension	2
discount factor (gamma)	0.99
PPO epsilon clip	0.2
PPO K epochs	80
optimizer learning rate actor	0.0003
optimizer learning rate critic	0.001
starting std of action distribution	0.6
the decay rate of std of action distribution	0.05
minimum std of action distribution	0.1
decay frequency of std of action distribution	250000 timesteps

4.3 PPO architecture with curriculum learning approach

Many considerations are taken into account by the proposed PPO to adapt with supposed environments to enhance convergence.

1- Reward function: The chosen reward function reflects the desired behavior which is reaching to the goal as quickly as possible while avoiding obstacles.

2- Exploration: ε-Greedy algorithm is used for balancing between exploration and exploitation the environment.

3- Value function: Advantage A, which is the difference between state value and action-state, is used in critic module to enhance the optimization process.

4- Network architecture: It is shown in fig for actor and critic, and hyperparameters as shown in table 2.
5- Regularization: It was achieved by weights decay to prevent overfitting.
6- CL: It is an approach where the agent is trained on progressively more difficult tasks to help it learn more efficiently. In the context of path planning for mobile robots, this could mean starting with simple environments without obstacles and gradually increasing the complexity with small distributed obstacles until it is trained to avoid a long obstacle. Towards testing, the robot can generalize the learnt behavior in a new environment which it is combined of training environments. This approach can help the agent learn faster and avoid optimization from getting stuck in local minima.

figure5 illustrates the main architecture of actor and critic neural networks of PPO, and how it is combined with CL according to proposed environments. In addition to that table 2 shows the chosen values of parameters which are chosen according to [29], and adapting some other values like max training timesteps, max timesteps per episode, optimizer learning rate actor, and optimizer learning rate critic empirically.

In the DDPG algorithm, the architecture for the actor network was mirrored in the critic network. This approach ensured consistency in their structural design. Furthermore, identical values of hyperparameters were applied to both networks.

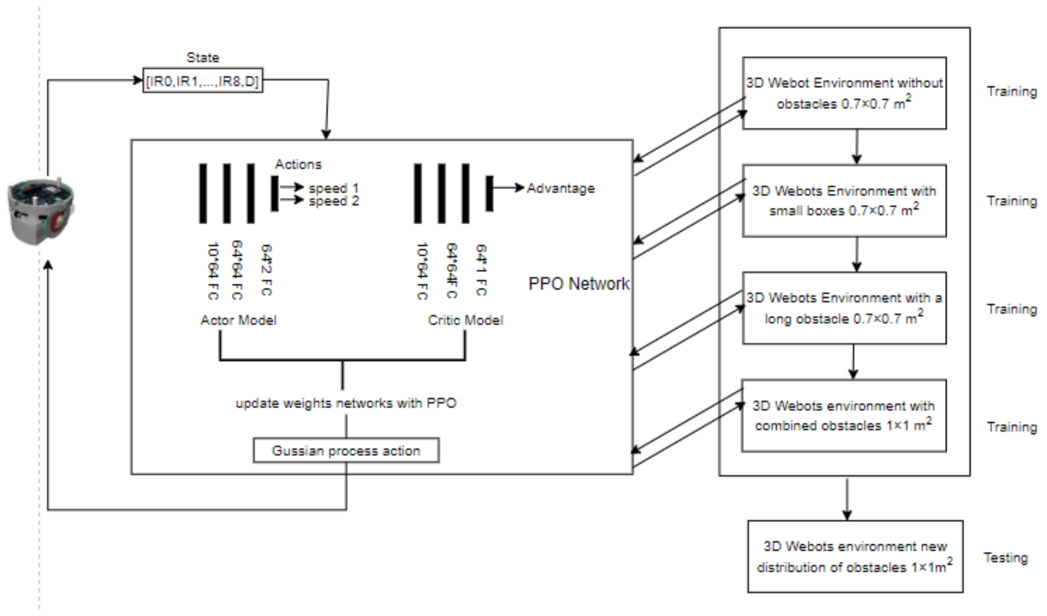
State
[IR0,IR1,...,IR8,D]
Actions
speed 1
speed 2
10*64 FC
64*64 FC
64*2 FC
Actor Model
Advantage
10*64 FC
64*64F C
64*1 FC
Critic Model
PPO Network
update weights networks with PPO
Gussian process action
3D Webot Environment without obstacles 0.7×0.7 m²
Training
3D Webots Environment with small boxes 0.7×0.7 m²
Training
3D Webots Environment with a long obstacle 0.7×0.7 m²
Training
3D Webots environment with combined obstacles 1×1 m²
Training
3D Webots environment new distribution of obstacles 1×1m²
Testing

Figure 5. Proposed PPO with curriculum architecture.

5 Results

In this study, we highlighted the crucial role of the reward function in achieving effective path planning for the mobile robot. PPO model using CL in four different environments of increasing difficulty was trained gradually. The trained model was subsequently tested in a novel environment with a new distribution of obstacles. In order to evaluate the effectiveness, and generalization of PPO with CL, the results were compared with both of PPO without CL and DDPG algorithm.

5.1 Shaping and Sparing rewards

The robot was trained in the Env_1 to reach the goal. RL learnt with shaped and sparse rewards given in equation 3, and equation 4. The metrics were used to evaluate the performance are Reaching to the goal each episode and the reward curve.

Reaching to the goal is a metric that take a value 1 when robot reaches the goal, and zero if it fails to reach each episode.

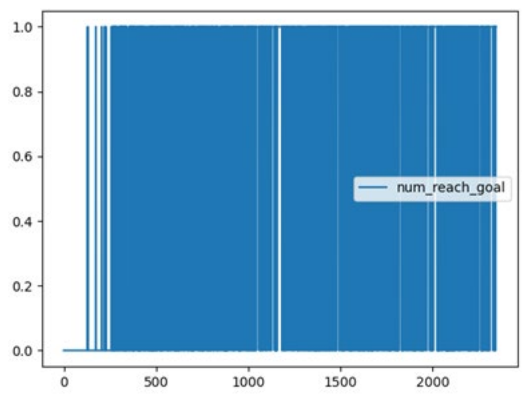

Figure.6-a. shaping rewards.

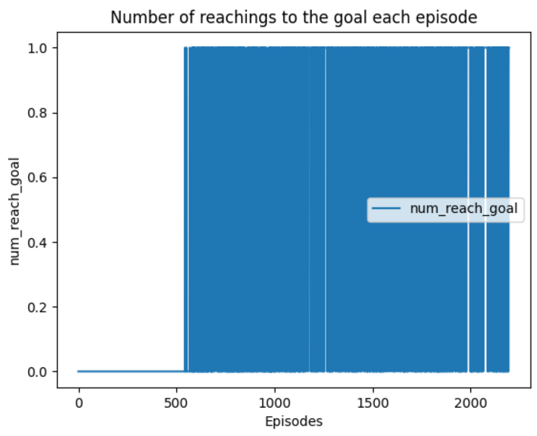

Figure.6-b. Sparse rewards.

Figure 6. Reaching the goal metric.

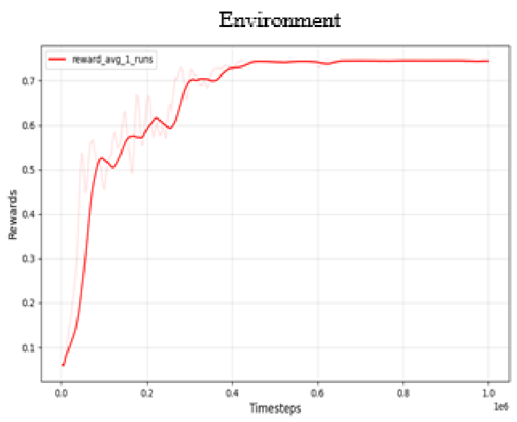

Figure.7-a. Shaping rewards.

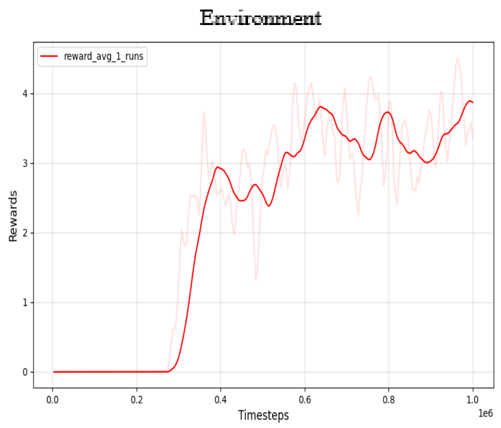

Figure.7-b. Spares rewards.

Figure 7. obtained rewards.

By comparing shaping rewards in figure 6-a and spares in figure 6-b, it is notable that for shaping method, the robot began to reach the goal earlier. As a result, shaping reward is better in performance as it makes the learning process faster. Figure 7-a provides more frequent and incremental feedback to the robot based on its actions. The curve starts at a lower reward value and then steadily increases, reaching a plateau. This suggests that the robot was learning over time and receiving shaping rewards as it gets closer to the desired behavior or outcome. The curve in figure 7-b is more volatile, with various peaks and troughs, indicating that the robot received rewards irregularly. This can be a more challenging environment for the robot to learn from because it must discover which actions lead to rewards through exploration, with less immediate feedback from the environment.

5.2 Robot's path planning

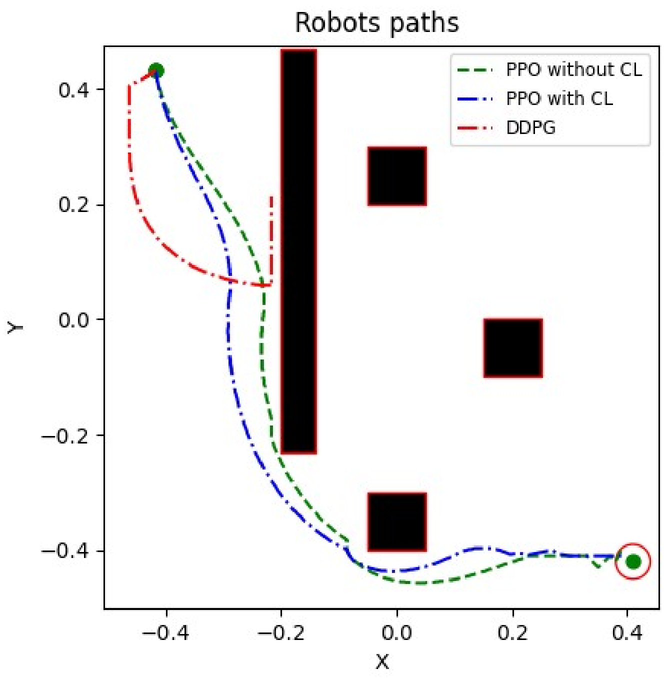

Figure 8. Robot's paths by proposed algorithms.

Figure 8 shows the result of path planning by three proposed algorithms. In the scenario where PPO is utilized without CL, the robot initially moves away from direction of the goal and seems to be exploring the space. Remarkably, it manages to avoid obstacles without any collisions, albeit not in the most direct manner. indicating that it has acquired obstacle avoidance skills, though its path is not the most straightforward. On the other hand, when equipped with the advantages of CL, PPO guides the robot along a more refined trajectory. This approach results in a more direct route to the goal, showcasing an improved navigational strategy post initial exploration.
The path generated by PPO without CL, while ultimately effective in avoiding obstacles, is less direct, hence less efficient. Conversely, DDPG fails to achieve the desired outcome in this context due to its inability to learn obstacle avoidance. Its performance is highly contingent on the fine-tuning of hyperparameters and the intricate balance within its actor-critic architecture. For this comparative study, the hyperparameters and structure tailored for PPO were adopted, which may not align well with the requirements of DDPG.
by starting simple and gradually increasing complexity, PPO with CL can avoid overfitting to specific scenarios. This can lead to a more generalized policy that performs well in various situations, suggesting why the path avoids obstacles more effectively. CL can help stabilize the learning process by breaking down the learning task into manageable stages. This can be particularly helpful in RL, where exploration can sometimes lead to large policy updates that destabilize learning. The smoother path of PPO with CL suggests a more stable learning progression.

5.3 Training efficiency

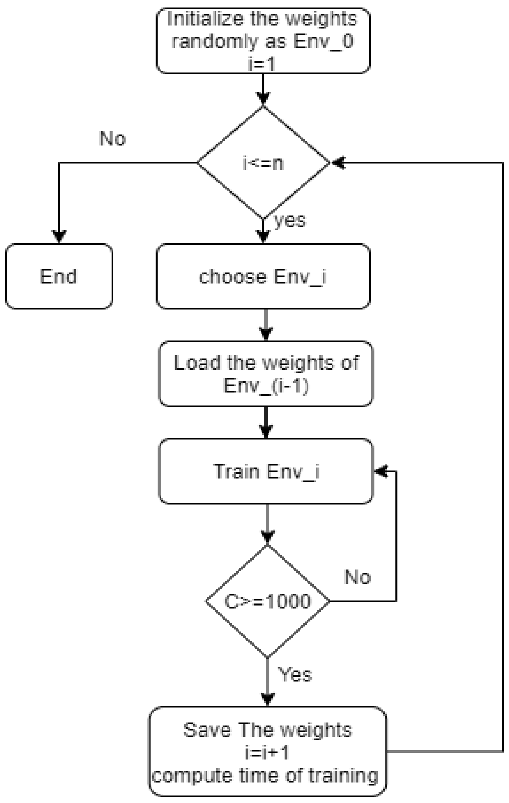
Initialize the weights randomly as Env_0 i=1
i<=n
No
yes
End
choose Env_i
Load the weights of Env_(i-1)
Train Env_i
C>=1000
No
Yes
Save The weights i=i+1 compute time of training

Figure 9: Curriculum learning.

The diagram in figure 9 outlines a CL process for the PPO model, where the weights are initially randomized and the model is trained sequentially on a series of incrementally challenging environments (Env_i). After each training phase on environment Env_i, using the weights from the previous environment (Env_(i-1)), When the robot records 1000 success attempts to reach the goal, the model is learnt and the weights is saved and the training time is computed as (t_Env1, t_Env2, t_Env3, t_Env4) and the process iterates to the next environment. It is called C criteria. The cycle continues until the model has been trained on all environments, at which point the training process concludes, and the time of training for PPO with CL is computed as in equation:

$$Training\ time = t_{Env1} + t_{Env2} + t_{Env3} + t_{Env4} \quad (5)$$

The results depicted in the figure 10 indicates a progression of the robot's learning and adaptability across various environments. In the simplest environment, env_1, the robot achieved the goal after about 124,000 timesteps, demonstrating the initial learning curve necessary to navigate without obstacles, as shown in figure 10-a-1.

In the subsequent environment, env_2, which introduced small boxes as new obstacles, the robot required approximately 236,000 timesteps to successfully reach the goal, as illustrated in figure 10-a-2. The increase in required timesteps is attributed to the penalties incurred from collisions, necessitating additional learning to avoid obstacles effectively.

Moving to a different challenge, env_3 featured a long obstacle, which the robot learned to manuver in a significantly shorter period, taking only about 32,000 timesteps, as shown in figure 10-a-3. This quick adaptation suggests that the transfer learning from previous environments enabled the robot to generalize its avoidance strategies more efficiently.

Finally, in env_4, the robot faced what appears to be a simpler task compared to the previous environments due to the absence of complex obstacles or the robot's accumulated learning experiences. It mastered this environment in roughly 44,000 timesteps, as displayed in figure 10-a-4. This indicates that previous training in more complex environments may have expedited the robot's learning process for new, but less challenging, scenarios.

Figure 10-5 illustrates the performance of the robot in environment env_4 using PPO without CL, showing a gradual increase in rewards . They idicates to reaching gola attempts near 452000 timesteps with some fluctuations, indicating a more protracted and potentially less efficient learning process.

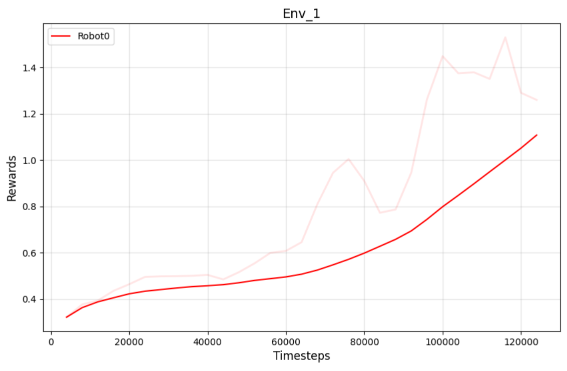

Figure 10-1

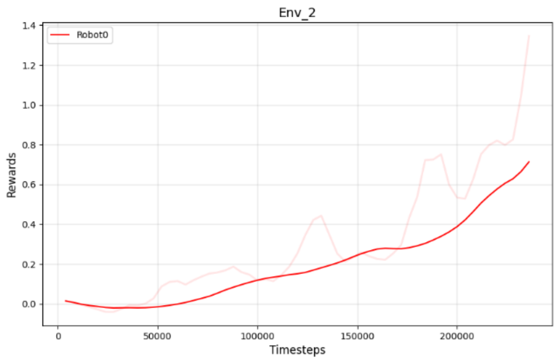

Figure 10-2

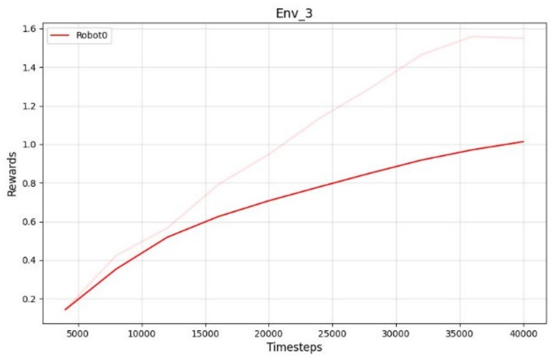

Figure 10-3

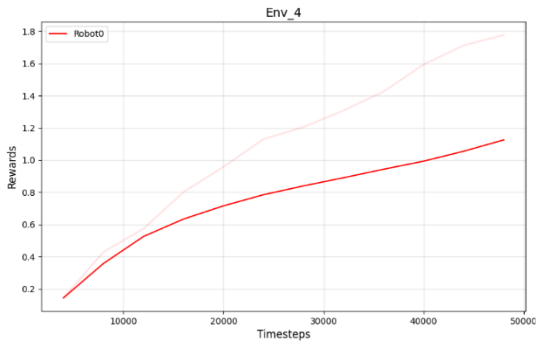

Figure 10-4

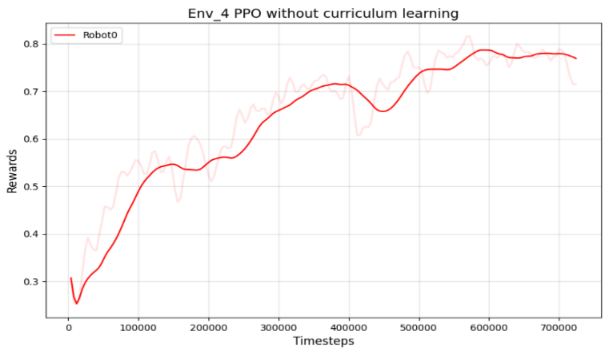

Figure 10-5

Figure 10. Robot's learning process with curriculum learning

The table 3 compares the training efficiency of DDPG, PPO, and PPO with CL. DDPG did not successfully learn any of the environments, as indicated by infinite training time. PPO with CL shows progressive learning across environments with a cumulative training time while standard PPO's slower learning in the final, most complex environment Env4 indicates it requires more extensive exploration and training time to adapt without the structured progression that CL provides.

Table 3. Training efficiency.

	$t1\_Env1$	$t2\_Env2$	$t3\_Env3$	t_Env4	Training time
DDPG	-	-	-	-	∞
PPO	-	-	-	452×10^3	452×10^3
PPO+CL	128×10^3	209×10^3	32×10^3	44×10^3	413×10^3

5.4 Generalization

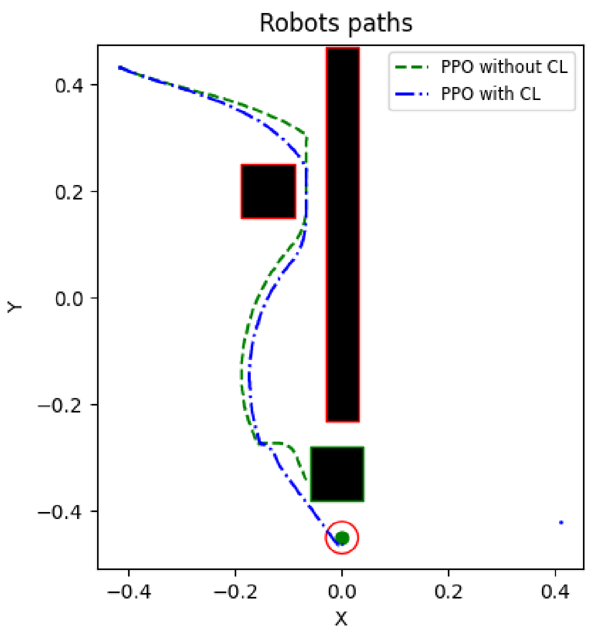

Figure 11. Ability to generalize.

The image displays the paths taken by a robot in a new environment where the positions of the obstacles have been altered, PPO with CL succeeds in generalizing to a new environment due to its incremental learning approach, which provides a diverse range of experiences that build a robust policy capable of adapting to changes. In contrast, PPO without CL fails to generalize because it may overfit to the specific training scenarios, lacking exposure to the varied situations necessary for developing a flexible strategy. The staged complexity introduced by CL results in a policy that understands the underlying principles of navigation and obstacle avoidance, rather than memorizing specific paths.

6 Conclusion

The study presented an evaluation of the impact of CL on the RL approach based on PPO algorithm for mobile robot navigation. The results demonstrated the superiority of incorporating CLand shaping rewards, with the robot achieving more efficient path planning in complex environments. The research also compared the performance of PPO with and without CL, indicating the significant benefits of the incremental learning approach. The enhanced performance of PPO with CL was evident in the robot's ability to generalize its learning to a new environment with altered obstacle configurations. The study also highlighted the limitations of the DDPG algorithm when applied to the same task due to high sensitivity to fine-tune the parameters, which underlines the importance of selecting appropriate RL strategies. The PPO with CL model showed promising results in terms of training efficiency and operational effectiveness. Future work should explore the applicability of modifying the CL by making it more adaptively to increasing the efficiency of training process. Further research could also investigate the systematic approach to increase the complexity of training environments to enhance both efficiency and flexibility. The study sets a foundation for more advanced autonomous robotic systems capable of adapting to and navigating within diverse and dynamic environments.

7 References

[1] S.Ibáñez, J.Ricardo, C. J.-P.-Pulgar, and A. G.-Cerezo. "Path Planning for Autonomous Mobile Robots: A Review." *Sensors* 21, no. 23, p 7898 (2021). https://doi.org/10.3390/s21237898

[2] K., K., N. Sharma, C. Dharmatti, and J. E. Siegel. "A survey of path planning algorithms for mobile robots." *Vehicles* 3, no. 3, p 448-468 (2021). https://doi.org/10.3390/vehicles3030027

[3] G., F., W. Rahiman, and S.S.N. Alhady. "A comprehensive study for robot navigation techniques." Cogent Engineering 6, no. 1,p: 1632046, (2019). https://doi.org/10.1080/23311916.2019.1632046

[4] Z., Y., Y. Zhang, and S. Wang. "A Review of Mobile Robot Path Planning Based on Deep Reinforcement Learning Algorithm." In Journal of Physics: Conference Series, vol. 2138, no. 1, p. 012011. IOP Publishing, (2021). https://doi.org/10.1088/1742-6596/2138/1/012011

[5] S., B., R. Kumar, and V. P. Singh. "Reinforcement learning in robotic applications: a comprehensive survey." Artificial Intelligence Review 55, no. 2,p 945-990 (2022). https://doi.org/10.1007/s10462-021-09997-9

[6] Bengio, Y., Louradour, J., Collobert, R., & Weston, J. "Curriculum Learning." In Proceedings of the 26th Annual International Conference on Machine Learning, pp. 41-48, (2009, June). https://doi.org/10.1145/1553374.1553380

[7] B., K., P. Chakravarty, and S. Shrivastava. "An A* curriculum approach to reinforcement learning for RGBD indoor robot navigation." arXiv preprint arXiv:2101.01774 (2021).https://doi.org/10.48550/arXiv.2101.01774

[8] P., J., S. Jang, and Y. Shin. "Indoor path planning for an unmanned aerial vehicle via curriculum learning." 2021 21st International Conference on Control, Automation and Systems (ICCAS). IEEE, (2021). https://doi.org/10.1109/ICTC55196.2022.9952572

[9] S., Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, (2018).

[10] Ç., S., and M. K. Pehlivanoğlu. "Model-free reinforcement learning algorithms: A survey." In *2019 27th Signal Processing and Communications Applications Conference (SIU)*, p. 1-4. IEEE, 2019. https://doi.org/10.1109/SIU.2019.8806389

[11] S., Y., Y. Liu, G. Wang, and H. Zhang. "Deep learning for plant identification in natural environment." *Computational intelligence and neuroscience* 2017 (2017). https://doi.org/10.1155/2017/7361042

[12] S., J., S. Levine, P. Abbeel, M. Jordan, and P. Moritz. "Trust region policy optimization." In *International conference on machine learning*, pp. 1889-1897. PMLR, (2015). https://doi.org/10.48550/arXiv.1502.05477

[13] C., N. "Deep deterministic policy gradient for urban traffic light control." *arXiv preprint arXiv:1703.09035* (2017). https://doi.org/10.48550/arXiv.1703.09035

[14] S., J., F. W., P. Dhariwal, A. Radford, and O. Klimov. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347* (2017). https://doi.org/10.48550/arXiv.1707.06347

[15] G., J., W. Ye, J. G., and Z. Li. "Deep reinforcement learning for indoor mobile robot path planning." *Sensors* 20, no. 19,p 5439, (2020). https://doi.org/10.3390/s20195493

[16] Z., W., J. P. Queralta, and T. Westerlund. "Sim-to-real transfer in deep reinforcement learning for robotics: a survey." In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 737-744. IEEE, (2020). https://doi.org/10.1109/SSCI47803.2020.9308468

[17] T., L., and M. Liu. "A robot exploration strategy based on q-learning network." In *2016 ieee international conference on real-time computing and robotics (rcar)*, pp. 57-62. IEEE, (2016). https://doi.org/10.1109/RCAR.2016.7784001

[18] L., Z., Q. Wang, and B. Yang. "Reinforcement Learning-Based Path Planning Algorithm for Mobile Robots." *Wireless Communications and Mobile Computing* 2022 (2022). https://doi.org/10.1155/2022/1859020

[19] W., W., Z. Wu, H. Luo, and B. Zhang. "Path Planning Method of Mobile Robot Using Improved Deep Reinforcement Learning." *Journal of Electrical and Computer Engineering* 2022 (2022). https://doi.org/10.1155/2022/5433988

[20] X., X., Z. Li, D. Zhang, and Y. Yan. "A deep reinforcement learning method for mobile robot collision avoidance based on double DQN." In *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, pp. 2131-2136. IEEE, (2019). https://doi.org/10.1109/ISIE.2019.8781522

[21] Y., J., Y. Su, and Y. Liao. "The path planning of mobile robot by neural networks and hierarchical reinforcement learning." *Frontiers in Neurorobotics* 14, p63, (2020). https://doi.org/10.3389/fnbot.2020.00063

[22] M., W., B. Park, S. H. Nengroo, T. Kim, and D. Har. "Path planning of cleaning robot with reinforcement learning." In 2022 IEEE International Symposium on Robotic and Sensors Environments (ROSE), pp. 1-7. IEEE, (2022). https://doi.org/10.48550/arXiv.2208.08211

[23] Y., L., J. Bi, and H. Yuan. "Dynamic Path Planning for Mobile Robots with Deep Reinforcement Learning." IFAC-PapersOnLine 55, no. 11 ,p: 19-24.(2022) https://doi.org/10.1016/j.ifacol.2022.08.042

[24] Y., X., P. Wang, and Z. Zhang. "Learning-based end-to-end path planning for lunar rovers with safety constraints." Sensors 21, no. 3, p: 796,(2021). https://doi.org/10.3390/s21030796

[25] J., X., and Z. Wang. "Proximal policy optimization based dynamic path planning algorithm for mobile robots." Electronics Letters 58, no. 1, p: 13-15,(2022). https://doi.org/10.1049/ell2.12342

[26] M., O. "Cyberbotics Ltd. Webots™: professional mobile robot simulation." *International Journal of Advanced Robotic Systems* 1, no. 1,p 5, (2004). https://doi.org/10.5772/5618

[27] M., F., M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli. "The e-puck, a robot designed for education in engineering." In *Proceedings of the 9th conference on autonomous robot systems and competitions*, vol. 1, no. CONF, pp. 59-65. IPCB: Instituto Politécnico de Castelo Branco, (2009).

[28] B., G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba."Openaigym." *arXiv preprint arXiv:1606.01540* (2016). https://doi.org/10.48550/arXiv.1606.01540

[29] K., M., K. Tsampazis, N. Passalis, and A. Tefas. "Deepbots: A webots-based deep reinforcement learning framework for robotics." In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pp. 64-75. Springer, Cham, (2020). https://doi.org/10.1007/978-3-030-49186-4_6