Effectiveness of Curriculum Learning in Reinforcement Learning Robotics

Daniel Dias, Lucas Santiago, Nuno Moreira, Rafael Conceição
MIA Students
Faculty of Engineering, University of Porto (FEUP)
Porto, Portugal
up202105076@fe.up.pt, up202104660@fe.up.pt, up202104873@fe.up.pt, up202006898@fe.up.pt

***Abstract*—This paper presents the implementation of Reinforcement Learning (RL) algorithms in an environment designed specifically for navigation/path planning tasks. Operating within the Webots Python framework and simulated in the Webots environment, we deploy a Curriculum Learning strategy in the training of said algorithms.**

Discrete and continuous RL algorithms are trained, tested and compared, namely PPO, DQN, SAC, and TD3. The simulated agent is an E-Puck robot equipped with several sensors mainly LiDAR, GPS, and Touch Sensor.

The approach involves training the robot in a simulated environment using a Curriculum Learning approach: different maps, different starting positions, and different starting actions. In simple terms, the approach consists of training the robot "level by level" starting on easy scenarios and doing transfer learning into increasingly more challenging scenarios.

Experimental results demonstrate that this method effectively enhances the training journey of the robot, both for discrete and continuous environments. Therefore, this solution has the potential to improve the autonomy and efficiency of robotic systems in various settings.

***Index Terms*—Reinforcement Learning, Path Planning, Navigation, Curriculum Learning, Simulation, Webots, E-Puck.**

I. Introduction

Path planning for robotic systems involves determining an optimal path for a robot to navigate from a starting point to a target location while avoiding obstacles. This task is critical in various domains, such as autonomous driving, warehouse automation, and service robotics. Traditionally, path planning has relied on deterministic algorithms, such as Dijkstra's and A*, but with the advent of more complex environments and the need for adaptive behaviors, reinforcement learning (RL) has emerged as a promising approach.

Automating path planning and navigation through reinforcement learning can lead to significant advances in the efficiency and adaptability of robotic systems. RL allows agents to learn optimal policies through interactions with their environment, enabling them to handle dynamic and unforeseen obstacles more effectively. However, the application of RL to path planning poses several challenges, including the need for efficient exploration strategies and the computational complexity of training models in high-dimensional spaces.

The focus of this study is the development of a robotic environment that uses RL algorithms for path planning. The robot is trained within the Webots simulation platform, which provides a versatile environment to test and validate robotic behaviors. The goal is to compare different RL algorithms while leveraging the Curriculum Learning approach designed by us.

Curriculum learning in RL robotics is a training strategy in which agents learn tasks progressively, starting from simple scenarios and gradually increasing complexity. Instead of jumping directly to the hardest environment, robots first master basic skills before tackling advanced challenges. This approach improves learning efficiency, reduces training time, and helps agents develop more robust behaviors by building foundational skills incrementally.

The proposed solution aims to achieve significantly faster training times and model convergence by reducing the amount of training time in over-complicated environments, while ensuring that the robot can navigate complex terrains. This is achieved by applying the Curriculum Learning approach explained in Section IV.

Section II provides a brief explanation of the simulation environment. Section III details our approach to implementing the reinforcement learning environment, including the detailed explanation of action spaces, observation spaces, rewards, etc. In Section IV, we present the implementation of the Curriculum Learning approach, highlighting the various steps of the process.

Section V analyzes the intermediate results and the robot behaviors seen between levels. Section VI and VII explain the training approach and display its results. Finally Section VIII compares the 4 RL algorithms used during training.

II. Simulation Environment

A. Setup

This work was carried out using the Webots Robot Simulator and its Python framework. The robot utilized for the experiments is the E-Puck robot, equipped with various sensors. We designed a custom map specifically for navigation and path planning that features challenging obstacles.

The simulation was carried out on a GPU equipped machine that allowed us to achieve fast training speeds reaching 50x faster than the real simulation time.

B. Robot

The robot used in the simulation is an E-Puck robot equipped with LiDAR, GPS and Touch sensors.

The LiDAR has an FOV of 2.62 rad (approximately 150º) with 25 horizontal rays equally spaced as can be seen on image 1.

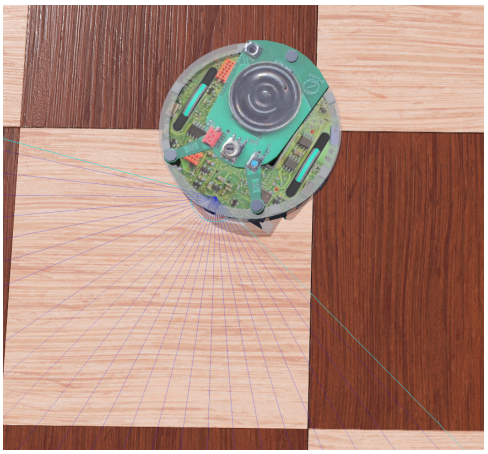

Fig. 1. Robot LiDAR rays

The robot's observation space consists of the LiDAR rays readings, the distance to the target, and the direction angle to the target.

The action space allows the robot to move forward and rotate in either direction. It is adapted according to the environment used (discrete/continuous).

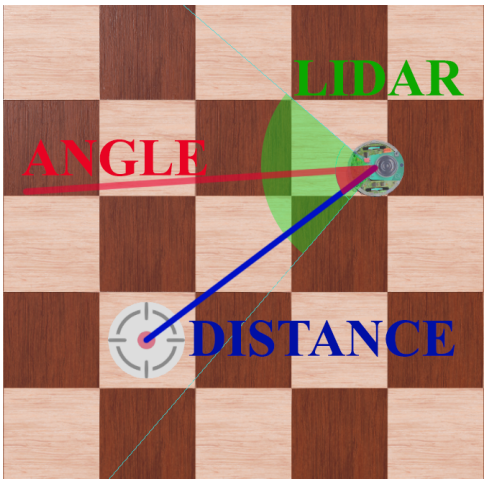

Fig. 2. Robot's observation space

C. Map

The 4 maps in image 3 were used to train the robot. We decided to have more than one map in order to allow the robot to experience diverse scenarios, adapting to whatever situation it encounters. Map 2 and map 3 have more than one target which allows for double target path planning scenarios.

These maps allow for different scenarios ranging from easy to hard which is very useful for our Curriculum Learning approach.

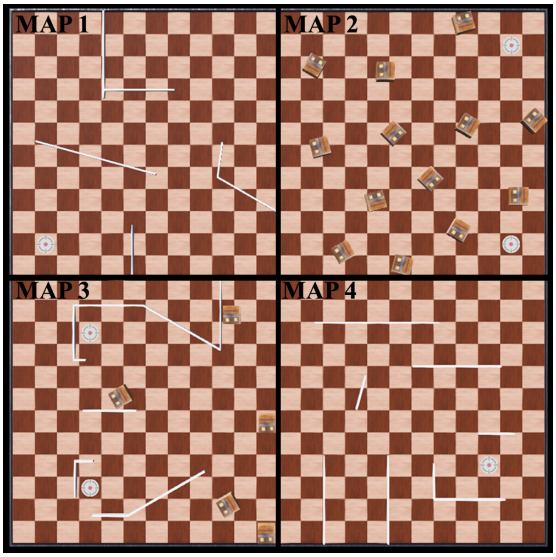

Fig. 3. Simulation Map

III. Reinforcement Learning Environment

A. Setup

The robotic navigation environment is implemented using Python's OpenAI Gymnasium framework, providing a standardized interface for reinforcement learning algorithms. The implementation consists of two variants: a discrete action-space version and a continuous action-space version.

The environment simulates an E-PUCK robot navigating through various obstacle configurations to reach target positions. The setup integrates several key components:

- **Simulation Platform**: Webots robotics simulator providing realistic physics
- **Robot Model**: E-PUCK robot equipped with LiDAR, GPS, compass, and touch sensors
- **RL Framework**: OpenAI Gymnasium for standardized RL interfaces
- **Algorithm Compatibility**: Full integration with Stable-Baselines3 library

B. Action and Observation Spaces

Discrete Action Space Version:

- **Action Space**: $\mathcal{A} = \{0, 1, 2\}$ representing forward movement, left turn, and right turn
- **Observation Space**: $\mathcal{O} \in \mathbb{R}^{27}$ containing:
  - 25 LiDAR distance readings $\in [0, 2]$
  - Distance to target $\in [0, \text{MAX\_DISTANCE}]$
  - Angle to target $\in [-\pi, \pi]$

Continuous Action Space Version:

- **Action Space**: $\mathcal{A} = [0, 1] \times [-1, 1]$ for $[v_{forward} \in [0, 1], \omega_{angular} \in [-1, 1]]$
- **Observation Space**: $\mathcal{O} \in \mathbb{R}^{27}$ containing:

- 25 LiDAR distance readings $\in [0, 2]$
- Distance to target $\in [0, \text{MAX\_DISTANCE}]$
- Angle to target $\in [-\pi, \pi]$

C. Metrics Collection System

The `MetricsTracker` class implements comprehensive data collection for training analysis:

Episode-Level Metrics:

- Episode rewards and step counts
- Outcome classification (success, collision, truncation)
- Exploration measure through unique locations visited
- Action distribution analysis (discrete version)

Training-Level Aggregates:

- Success rate: $\frac{\text{successful episodes}}{\text{total episodes}} \times 100\%$
- Collision and truncation rates
- Average episode metrics (length, reward, exploration)
- Training efficiency measures (total steps, time)

D. Reward Function Design

The reward function employs dense reward shaping with multiple behavioral incentives. The structure balances task completion with efficient navigation behaviors:

$r \leftarrow 0$
if collision detected **then**
  return $r = -500$, done = True
end if
if target reached **then**
  if final target **then**
    return $r = +500$, done = True
  else
    $r \leftarrow r + 300$
    remove target from list
  end if
end if
$d \leftarrow$ distance_to_target()
$r \leftarrow r + \min(3, \frac{0.3}{d})$ {Distance-based reward}
$r \leftarrow r + (d_{prev} - d) \times 2$ {Progress reward}
$r \leftarrow r - (0.5 + \frac{\text{steps}}{400})$ {Time penalty}
if new location visited **then**
  $r \leftarrow r + 0.1 \times$ exploration_bonus
end if
$r \leftarrow r + \|\Delta\text{position}\| \times 0.5$ {Movement reward}
for all LiDAR reading l_i **do**
  if $l_i \leq 0.2$ **then**
    $r \leftarrow r - (0.2 - l_i) \times 3$ {Wall proximity penalty}
  end if
end for
if stationary behavior detected **then**
  $r \leftarrow r -$ stationary_steps $\times 0.25$
end if
$\theta \leftarrow$ angle_to_target()
if clear path to target **then**
  $r \leftarrow r + \min(1.0, \frac{0.2}{|\theta|})$ {Directional bonus}
end if
return r, done = False

Key Reward Components:

- **Task Completion**: Large positive rewards for reaching targets (+300/+500)
- **Safety**: Collision penalties (-500) and predictive avoidance
- **Efficiency**: Progress rewards and time penalties
- **Exploration**: Bonuses for visiting new locations
- **Behavioral Shaping**: Anti-spinning penalties and wall avoidance

E. Predictive Collision Avoidance

To enhance safety and training efficiency, the environment implements a predictive collision avoidance mechanism within the step function. This system prevents the robot from executing actions that would likely result in immediate collisions, therefore reducing negative experiences and improving learning convergence.

The predictive system operates by analyzing LiDAR readings before performing the action. When the robot attempts significant forward movement (action value > 0.3 in continuous mode or forward action in discrete mode), the system examines the front-facing sensor readings to detect potential obstacles.

Implementation Details:

- **Sensor Analysis**: The system focuses on a 42° frontal cone by selecting the center LiDAR ray plus 3 rays on each side
- **Collision Threshold**: If the minimum distance among these front-facing rays is ≤ 0.085 meters, a potential collision is detected
- **Preventive Response**: Instead of executing the risky forward action, the system:
  - Applies a moderate penalty (-15 reward points)
  - Prevents action execution for up to 5 consecutive attempts
  - Returns early from the step function without robot movement
- **Adaptive Behavior**: After 5 consecutive preventions, the system allows action execution to avoid indefinite blocking.

This mechanism serves two purposes: reduces the frequency of collision-based episode terminations during training while teaching the agent to be more cautious in obstacle-dense environments. The moderate penalty encourages the agent to learn alternative navigation strategies without the severe disruption of episode termination.

IV. Curriculum Learning Approach

The training was divided into 4 main categories: EASY, MEDIUM, HARD, ALL. This approach is known as "Curriculum Learning." Curriculum Learning is a training strategy in machine learning where a model is progressively trained on increasingly difficult tasks. The idea is to first train the model on simpler categories (EASY and MEDIUM) and gradually increase the complexity (HARD and ALL).

This method can lead to better generalization and faster training times compared to training the model on difficult tasks from the start.

Two other categories are also present in the environment: START and RANDOM. These are useful because they have extreme scenarios:

- **START:** Allows the robot to understand that the target is rewarding. The robot has a clear path to the target. The robot is automatically rotated towards the target in the beginning of each episode.
- **RANDOM:** For algorithm generalization testing. The central divisions of the map are removed, creating a huge map. 3 target scenarios are introduced. This allows us to test the algorithms and how they can adapt to unseen scenarios.

The main categories implement normal scenarios with different difficulties. The robot can start in various positions and aim for either one or two targets. The positions can be seen in the image 4:

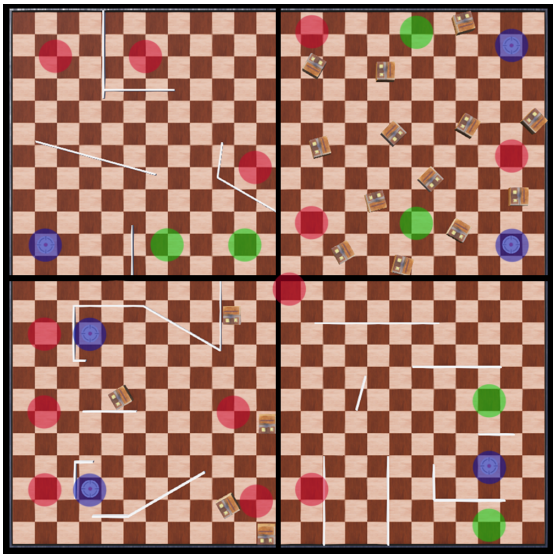

Fig. 4. Map positions

- **EASY:** In the beginning of every episode, the robot will start in a randomly selected green position and aim at a single blue target. Every single combination that can occur will always grant that between the robot and the target there is only one obstacle. The robot is also manually rotated into the target's direction.
  This simple setup was designed with the idea of briefly help the robot understand that going for the target is as good action, while going towards a wall is not a good idea.
- **MEDIUM:** Position and target are selected in the same way as the EASY level. The only difference is that in the MEDIUM setup, the robot will start with a random orientation at the beginning of each episode.
  The idea of this setup is to see if the robot is capable of autonomously rotate toward the target at the start of each episode.
- **HARD:** The initial and target positions are chosen by randomly selecting a red and a blue position, respectively. In the cases where the map has more than one target (map 2 and map 3), the robot will face a double-target path planning scenario. In this setup, the robot is also manually rotated into the closest target direction.
  The intention of the HARD level is to teach the robot that in not every scenario it is good to start by rotating towards the target, since this level contains many more scenarios where there are several obstacles directly between the robot and the target. In addition, it is used to introduce difficult scenarios and double-target navigation.
- **ALL:** The ALL level gathers all the start position/target combinations from the EASY and HARD levels. This time the robot is not hard coded into rotating to the target. The ALL level was the last step of the Curriculum Learning approach. Without our Curriculum Learning approach, the robot would start directly in this scenario.

The behaviors of the first "levels" of the Curriculum Learning approach allow the robot to train on the complex ALL setup, with much better performance, and achieve consistent results way faster than just starting in the ALL level without any previous information.

V. Curriculum Learning Results Analysis

The curriculum learning approach demonstrated significant advantages in training efficiency and behavioral development across all difficulty levels. This section analyzes the emergent behaviors observed at each training stage and compares the curriculum approach against traditional training methods.

A. Progressive Skill Development

The curriculum learning methodology enabled the systematic development of navigation capabilities, with each level building upon previously acquired skills. The following behavioral patterns emerged during training:

- **START:** The robot learned fundamental goal-seeking behavior within 10,000 timesteps. This initial stage established the basic understanding that reaching the target location yields positive rewards, forming the foundation for all subsequent learning.
- **EASY:** Building on the goal-seeking behavior, the robot developed basic obstacle avoidance skills within 50,000 time-steps. The pre-established target orientation from the START level accelerated convergence, as the robot could focus on navigation rather than re-learning goal identification.
- **MEDIUM:** The robot autonomously developed a strategic turning behavior at episode initialization. Without explicit programming, the agent learned to orient itself toward the target before proceeding, demonstrating the emergence of planning-like behavior in simple scenarios.

- **HARD:** Multi-target navigation capabilities emerged efficiently, leveraging the obstacle avoidance and goal-seeking skills from previous levels. While complex scenarios initially posed challenges, the accumulated knowledge enabled rapid adaptation and stable policy convergence.
- **ALL:** The integration of all previously learned behaviors resulted in robust navigation performance across diverse scenarios. The curriculum foundation significantly reduced the training time required for handling complex multi-obstacle environments.
- **RANDOM:** The final stage demonstrated successful generalization to completely novel configurations, including three-target scenarios. The learned policies proved robust enough to handle previously unseen obstacle arrangements without additional training.

B. Training Efficiency Analysis

Across all tested algorithms, the curriculum learning approach consistently demonstrated similar efficiency gains. Figure 5 presents the PPO training progression as a representative example, showing reward evolution across curriculum stages.

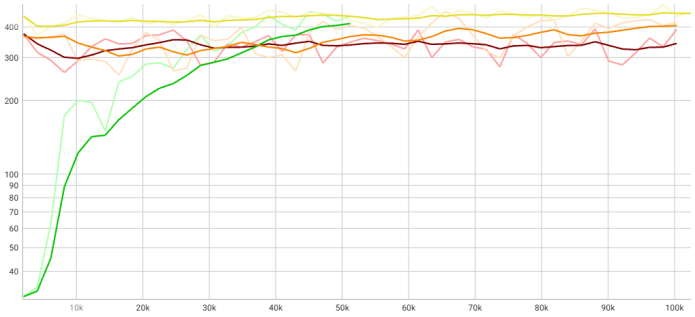

Fig. 5. PPO Curriculum Learning rewards by step

The progression reveals that fundamental navigation behaviors established during the EASY stage (green line) enabled near-instantaneous convergence in subsequent levels. The MEDIUM, HARD, and ALL stages (yellow, orange, and dark red lines respectively) exhibit remarkable stability throughout training, with minimal variance and rapid achievement of optimal performance. This stability indicates that the curriculum approach successfully transferred learned behaviors across increasing complexity levels.

C. Curriculum Learning vs. Direct Training Comparison

To quantify the benefits of curriculum learning, we conducted a comparative experiment training PPO directly on the most challenging "ALL" level without progressive difficulty scaling. Figure 6 illustrates the performance difference between curriculum-trained and directly-trained agents.

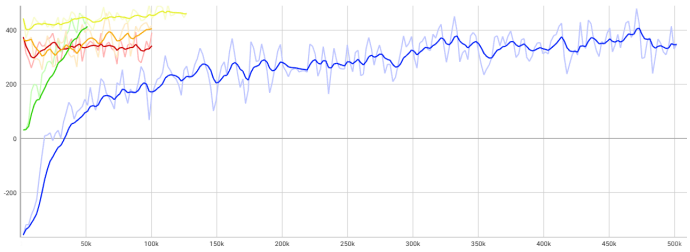

Fig. 6. PPO from scratch rewards by step

The direct training approach (blue line) required approximately 300,000 time-steps to achieve performance comparable to the curriculum learning endpoint, representing a 5x increase in training requirements. Moreover, the directly-trained agent never achieved the smooth, stable performance characteristic of the curriculum-trained agent (dark red line), exhibiting higher variance and less consistent reward accumulation.

Episode length analysis provides additional insight into the qualitative differences between training approaches. Figure 7 demonstrates that directly-trained agents consistently require more steps to complete episodes, indicating less efficient navigation strategies.

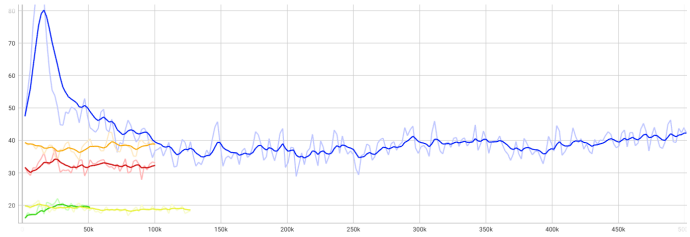

Fig. 7. PPO from scratch episode length by step

The curriculum-trained agent (dark red line) converges to significantly shorter episode lengths and maintains lower variance, suggesting the development of more direct and purposeful navigation policies. The directly-trained agent's consistently higher episode lengths indicate the presence of inefficient behaviors such as unnecessary exploration or suboptimal path planning.

D. Implementation Implications

The results suggest that curriculum learning with adaptive early stopping would provide optimal training efficiency. By implementing convergence detection at each curriculum stage, training time could be further reduced while maintaining the quality benefits of progressive learning. This approach would automatically advance to the next difficulty level upon achieving stable performance, minimizing unnecessary training on mastered skills while ensuring solid foundation development.

The consistent improvements observed across different algorithms indicate that curriculum learning benefits are algorithm-agnostic, making this approach broadly applicable to robotic navigation tasks regardless of the specific reinforcement learning method employed.

VI. Training

The training script serves as the orchestrator for the reinforcement learning experiments, integrating the environments with state-of-the-art algorithms from the Stable-Baselines3 library.

A. Algorithms and Setup

The training framework supports four distinct reinforcement learning algorithms through Stable-Baselines3 (SB3): Proximal Policy Optimization (PPO), Deep Q-Network (DQN), Soft Actor-Critic (SAC), and Twin Delayed Deep Deterministic Policy Gradient (TD3).

The framework automatically handles model persistence, allowing for checkpoint loading and continuation of training sessions. This capability enables incremental learning across difficulty levels and extended training periods.

B. Progressive Training Methodology

The training follows a Curriculum Learning approach with five distinct difficulty levels, each requiring specific training durations optimized through empirical testing:

1) **Start Level** (25,000 steps): Single target position with robot initially oriented toward the goal, establishing basic navigation capabilities
2) **Easy Level** (50,000 steps): Multiple predefined positions with target-oriented initialization, introducing environmental variety
3) **Medium Level** (125,000 steps): Easy positions with random robot orientations, requiring directional learning without initial guidance
4) **Hard Level** (200,000 steps): Complex obstacle configurations with target-oriented initialization, demanding sophisticated obstacle avoidance
5) **All Level** (100,000 steps): Comprehensive training across all map configurations with random orientations, achieving robust generalization

C. Metrics Collection and Monitoring

The training script incorporates comprehensive metrics tracking through a custom `MetricsCallback` class that monitors training progress without creating intermediate checkpoints.

TensorBoard integration provides real-time visualization of training progress, with logs organized by algorithm, difficulty level, and training duration. The metrics are automatically saved to CSV files for post-training analysis and comparison across different experimental configurations.

VII. Results

A. Performance Summary

Table I presents the key performance metrics for each algorithm after 600,000 total training steps.

TD3 emerged as the clear winner with a remarkable 90.2% success rate and the lowest collision rate of 9.8%. The algorithm demonstrated exceptional efficiency with an average of only 31.3 steps per episode while achieving the highest average reward of 541.

TABLE I
Algorithm Performance Comparison

Alg.	Success (%)	Steps (avg)	Reward (avg)	Locations (avg)
TD3	90.2	31.3	541	28.3
SAC	67.0	38.5	377	33.6
PPO	61.0	32.8	342	26.1
DQN	19.1	232.1	-1,617	50.3

B. Algorithm-Specific Analysis

TD3's superior performance can be attributed to several key algorithmic features:

- **Continuous Action Space**: TD3's native support for continuous actions allows for smooth, precise robot movements, particularly beneficial for navigation tasks requiring fine motor control
- **Target Policy Smoothing**: The addition of noise to target actions during training prevents overfitting to narrow peaks in the Q-function, leading to more robust policies
- **Delayed Policy Updates**: By updating the actor network less frequently than the critic, TD3 reduces the variance in policy updates, resulting in more stable learning
- **Twin Critic Networks**: The use of two Q-networks with the minimum value taken for updates helps mitigate overestimation bias, a critical factor in achieving high success rates

SAC achieved a solid 67.0% success rate. While not matching TD3's performance, SAC demonstrated strong exploration capabilities, visiting an average of 33.6 unique locations per episode—the highest among all algorithms. SAC's performance characteristics reflect its algorithmic design. SAC's entropy maximization encourages exploration, leading to the highest unique location count but potentially suboptimal convergence to deterministic policies.

PPO achieved a 61.0% success rate, demonstrating moderate but consistent performance. The algorithm completed 3,062 episodes with relatively efficient 32.8-step episodes on average. PPO's action distribution revealed a preference for forward movement (66%) over turning actions (16% each), indicating learned directional bias:

TABLE II
PPO Action Distribution

Action	Frequency (%)
Forward (0)	66
Turn Left (1)	16
Turn Right (2)	16

PPO's clipped objective function prevents large policy updates, ensuring stable but potentially slower convergence. The discrete action space (forward, turn left, turn right) may have constrained the robot's ability to make optimal fine-grained movements.

DQN significantly underperformed with only a 19.1% success rate. Despite completing only 430 episodes, DQN

agents explored extensively (50.3 unique locations on average) but failed to learn effective navigation policies. The poor performance can be attributed to several factors:

- **Overestimation Bias**: Standard DQN suffers from Q-value overestimation, which may have led to overly aggressive policies resulting in frequent collisions
- **Experience Replay Inefficiency**: While experience replay generally improves sample efficiency, the long episodes (232.1 steps on average) suggest the agent struggled to learn from its experiences effectively
- **Action Distribution Imbalance**: The balanced turning preference (42% each direction) versus low forward movement (15%) (image III) indicates the agent learned to avoid obstacles by spinning rather than navigating purposefully, which was observed during training.

TABLE III
DQN ACTION DISTRIBUTION

Action	Frequency (%)
Forward (0)	15
Turn Left (1)	42
Turn Right (2)	42

C. Key Performance Insights

The results reveal several critical insights about algorithm suitability for robotic navigation tasks:

*a) **Continuous vs. Discrete Actions**:* The superior performance of continuous action algorithms (TD3, SAC) over discrete action algorithms (PPO, DQN) highlights the importance of action space design. Navigation tasks benefit significantly from the ability to make fine-grained adjustments to movement and rotation speeds.

*b) **Exploration vs. Exploitation Balance**:* While SAC's high exploration (33.6 unique locations) might seem advantageous, TD3's more focused exploration (28.3 unique locations) combined with higher success rates suggests that directed exploration toward goal-oriented behavior is more effective than undirected exploration.

VIII. BENCHMARKING

This section benchmarks the four reinforcement learning algorithms used — PPO, DQN, SAC, and TD3 — across the different curriculum learning stages. We evaluate each algorithm's learning efficiency, stability, and final performance through episode reward and length metrics. The analysis identifies the strengths and weaknesses of on-policy (PPO), off-policy value-based (DQN), and off-policy actor-critic (SAC, TD3) methods in robotic navigation tasks, providing insights for algorithm selection in similar applications.

(NOTE: graphs in images 8, 9, and 10, have the following label: PPO-black; DQN-light blue; SAC-purple; TD3-yellow.)

A. Algorithm Performance by Level

EASY Training Step Results: In the EASY training phase, TD3 shows superior performance with the fastest convergence to the highest episode rewards (500) and maintains consistent episode lengths throughout training. SAC achieves the second-best performance, reaching rewards of 400-450 with generally stable learning, though it exhibits a notable spike in episode length around 10k steps, likely due to exploration.

PPO shows gradual but steady improvement, reaching rewards of 300-400, while DQN displays the most volatile behavior with a high variance in rewards despite eventually reaching competitive performance levels. DQN also shows distinctively high initial episode lengths (85 steps) before rapidly decreasing, suggesting an exploration-heavy initial phase followed by efficient learning.

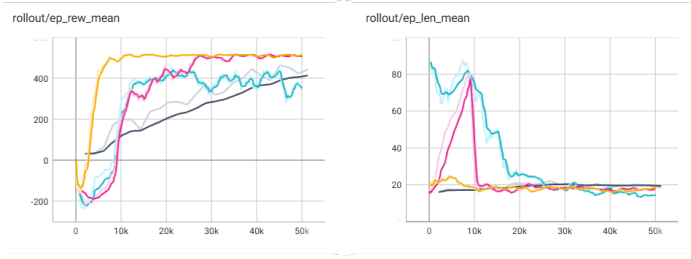

Fig. 8. EASY level benchmarking

Medium Training Step Results: The medium curriculum stage reveals remarkably similar performance across all algorithms. TD3, SAC, and PPO achieve comparable final rewards (500-550) with similar convergence rates and stability. All three algorithms maintain consistent episode lengths around 20k steps after initial learning phases. DQN shows slightly more volatility in rewards but ultimately reaches similar performance levels, demonstrating that the increased complexity of random target orientations does not significantly differentiate between algorithm capabilities at this difficulty level.

Hard Training Step Results: The hard curriculum stage exposes significant performance differences, particularly highlighting DQN's limitations:

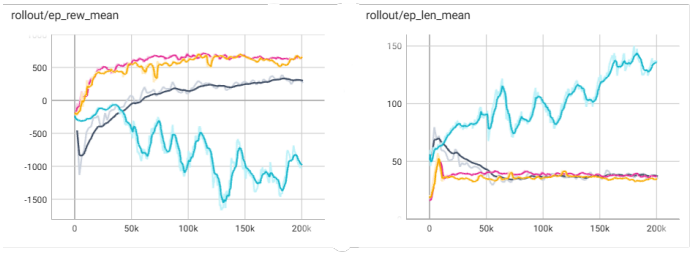

Fig. 9. HARD level benchmarking

- **Reward Performance:** TD3, SAC, and PPO maintain strong performance with rewards reaching 500, while DQN exhibits catastrophic failure with highly volatile

rewards oscillating between -1500 and positive values, never achieving stable learning.

- **Episode Length:** DQN's episode lengths become increasingly erratic and prolonged (up to 150 steps), indicating the agent fails to develop efficient navigation strategies. In contrast, other algorithms maintain reasonable episode lengths (30-40 steps).

DQN's poor performance in hard scenarios stems from its value-based approach struggling with the increased state-action space complexity and sparse rewards. The challenging starting positions and complex navigation requirements exceed DQN's sample efficiency, leading to unstable Q-value estimates and poor policy extraction. Additionally, DQN's discrete action space may be insufficient for the precise control required in difficult navigation scenarios.

RANDOM Training Step Results: The random curriculum stage, featuring a completely different map environment without middle walls and 1-3 target scenarios with random positioning, reveals distinct algorithm behaviors:

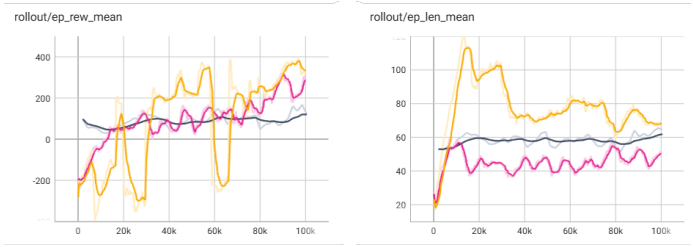

Fig. 10. RANDOM level benchmarking

- **Reward Performance:** SAC demonstrates superior stability and performance, achieving consistent rewards around 200-300 with minimal volatility. TD3 shows highly erratic behavior with extreme oscillations between -300 and +400, indicating unstable learning in this complex environment. PPO maintains moderate but steady performance around 100-150 rewards.
- **Episode Length:** SAC maintains the most efficient navigation with episode lengths stabilizing around 45-50 steps. TD3 exhibits significant instability with episode lengths that vary dramatically (60-110 steps), reflecting its inconsistent policy learning. PPO shows super stable episode lengths around 55-65 steps.

The random environment's increased complexity and multi-target scenarios favor SAC's soft actor-critic approach, which handles exploration-exploitation trade-offs more effectively through its entropy regularization.

TD3's deterministic policy and twin critic networks appear less suited for the high variability and uncertainty inherent in random positioning and multiple target scenarios. SAC's stochastic policy enables better adaptation to the diverse and unpredictable nature of this environment, while TD3's brittleness becomes apparent when faced with such complexity.

IX. Conclusions and Future Work

This study successfully demonstrated the effectiveness of curriculum learning in reinforcement learning robotics through comprehensive experimentation with four distinct algorithms—PPO, DQN, SAC, and TD3—in a simulated E-Puck robot navigation environment. The progressive training approach, advancing from simple single-target scenarios to complex multi-target configurations, proved superior to direct training on challenging tasks.

The results clearly establish TD3 as the most effective algorithm for robotic navigation tasks, achieving a remarkable 90.2% success rate with only 31.3 steps per episode on average. This superior performance stems from TD3's continuous action space, target policy smoothing, delayed policy updates, and twin critic networks, which collectively enable precise motor control and stable learning convergence. SAC demonstrated strong exploration capabilities, while PPO showed moderate but consistent performance. DQN's poor performance highlighted the limitations of discrete action spaces and value-based methods in navigation tasks requiring fine-grained control.

The curriculum learning approach delivered substantial efficiency gains across all algorithms, reducing training time by approximately 5x compared to direct training on complex scenarios. The progressive skill development enabled robots to build foundational navigation behaviors before tackling advanced multi-target scenarios, resulting in more stable and efficient policies. Particularly noteworthy was the emergence of autonomous strategic behaviors, such as target-oriented rotation in medium difficulty levels, without explicit programming.

The comprehensive benchmarking revealed that continuous action algorithms significantly outperform discrete action methods in navigation tasks, emphasizing the importance of action space design. Additionally, the exploration-exploitation balance proved critical, with directed exploration toward goal-oriented behavior being more effective than undirected exploration.

A. Future Work

The most immediate enhancement to this work involves implementing adaptive early stopping mechanisms within the curriculum learning framework. By incorporating convergence detection at each curriculum stage, the training process could automatically advance to the next difficulty level upon achieving stable performance metrics. This would minimize unnecessary training time on mastered skills while ensuring solid foundational development, potentially reducing total training time even further while maintaining performance quality.

Additional future research include extending the curriculum learning investigating the transferability of learned policies across different robot platforms. The development of more sophisticated reward shaping techniques and the exploration of hierarchical reinforcement learning approaches could further enhance the efficiency and robustness of the navigation policies.

X. References

References

[1] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *arXiv preprint arXiv:1606.01540*. Available: https://gymnasium.farama.org/

[2] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268), 1-8. Available: https://stable-baselines3.readthedocs.io/

[3] Michel, O. (2004). Cyberbotics Ltd. Webots™: professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1), 39-42. Available: https://cyberbotics.com/

[4] Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. *Proceedings of the 26th annual international conference on machine learning*, 41-48.

[5] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[6] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529-533.

[7] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., ... & Levine, S. (2018). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

[8] Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. *International conference on machine learning*, 1587-1596.

[9] Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., ... & Retornaz, P. (2009). The e-puck, a robot designed for education in engineering. *Proceedings of the 9th conference on autonomous robot systems and competitions*, 1(1), 59-65.

[10] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Available: https://www.tensorflow.org/tensorboard