Evaluation of Spherical Robots Navigation Using Deep Reinforcement Learning

James Zhang [*] [1] Haotian Zhang [*] [2]

Abstract

This study investigates the integration of model-based control techniques with deep reinforcement learning (DRL) for autonomous navigation of spherical robots in complex maze environments. A hybrid control framework is developed by combining traditional PID and spherical controllers with state-of-the-art DRL algorithms, including VPG, PPO, DDPG, TD3, and DQN. Simulations are conducted using ROS, Gazebo, and Python, featuring a fixed maze with dynamic obstacles and shifting goal positions. Experimental results show that DQN achieves the highest success rate, while TD3 delivers the best overall return in pure DRL setups between on-policy and off-policy algorithms. Hybrid approaches, particularly DDPG combined with PID, improve stability and learning consistency. These findings highlight the trade-off between adaptability and robustness in combining feedback control with learning-based navigation.

1. Introduction

Spherical robots appeared for multiple years, such as famous character BB-8 in Star Wars, but there haven't been much researches about their autonomous manipulations. They have unique advantages in terms of maneuverability, stability, and adaptability, making them attractive for navigation tasks in complex terrestrials(Escorza et al., 2025; Keymasi-Khalaji et al., 2025). However, navigating intricate environments with such a unique shape reveals significant challenges, including precise motion control, obstacle avoidance, and perfect path planning(Keymasi-Khalaji et al., 2025). Traditional methods like PID controllers and specialized spherical controllers offer stability but are easier to fail to adapt to dynamic and complicated maze scenarios(Li et al., 2023; Schröder et al., 2023).

Reinforcement Learning has showed promising results in solving puzzles, like AlphaGo(Li et al., 2023) and maze. Recent advancements in deep reinforcement learning (DRL) algorithms, including on-policy methods like Proximal Policy Optimization (PPO) and off-policy methods including Twin Delayed Deep Deterministic Policy Gradient (TD3), Deep Deterministic Policy Gradient (DDPG), Soft Actor-Critic (SAC), and Deep Q-Network (DQN), show promising results thanks to their strong adaptability and rapid learning capacity in a maze solving scenario. However, standalone DRL approaches can sometimes exhibit instability and slow convergence in real-world robotic maze-solving challenges.

To make a better and more efficient spherical robot simulation, this project explores a hybrid approach, which integrates a robust spherical controller, based on previous research(Schröder et al., 2023), combining with different state-of-the-art DRL algorithms. By combining the deterministic stability of traditional spherical controllers with the adaptive DRL algorithms, this study aims to evaluate the viability and performance of such hybrid approach, specifically providing valuable insights into the effectiveness and practicality of blending classical control methods with modern reinforcement learning techniques for spherical robotic navigation.

2. Methodology and Algorithms

This section provides a detailed explanation of the methodologies employed in this paper, with focus on the integration of classical spherical control techniques with state-of-the-art DRL algorithms to enable efficient and effective path planning for spherical robots navigating complex maze environments. To start with, traditional control algorithms, such as PID and a specialized spherical controller, are introduced, and their stability roles are highlighted here. Next, various deep reinforcement learning techniques implemented are discussed, both on-policy and off-policy strategies, each offering advantages and disadvantages in planning with this maze condition specifically. Finally, the simulation environment used to evaluate these methods is briefly described and

[*]Equal contribution [1]UNI: tz2642 [2]UNI: hz2994. Correspondence to: James Zhang <tz2642@columbia.edu>, Haotian Zhang <hz2994@columbia.edu>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

we will reveal setup considerations.

2.1. Controller Algorithms

To evaluation the viability of involvement of control, as well as ensure robust control of spherical robot movements, two traditional control algorithms were selected for analysis:

- **Proportional-Integral-Derivative controller (PID)** is a widely used feedback control mechanism because of its simplicity, effectiveness, and ease of tuning(Li et al., 2023), and PID controllers have solved 90% of real world control problem(Zhao et al., 2025). The goal of PID controller is to minimize the error between expected and actual outputs by numerical calculation. Since it is simple to operate, PID controllers always face challenges in dynamic and nonlinear environments, which always require multiple inputs and outputs. Specifically with the manipulation of spherical robots, PID can't satisfy its control requirements for precision and effectiveness.

- Derived from recent literature(Schröder et al., 2023), the **spherical controller** is specifically designed for spherical robot dynamics, which specifically aims for those unique challenges such as rolling stability and dynamic responsiveness. Unlike standard control methods, this spherical controller incorporates specialized feedback loops designed explicitly for spherical locomotion, improving navigation accuracy and stability within paths and dynamical environments. Based on the paper(Schröder et al., 2023), we have these two control equations

$$\mathrm{w}(t) = -k_{a_e}\alpha_{a_e}(t) - k_{eacc}\alpha_{eacc}(t) \tag{1}$$

$$V(t) = \begin{cases} V_{\max} & \text{if } |d(t)| > k_r \\ \frac{d(t)V_{\max}}{k_r} & \text{if } |d(t)| \leq k_r \end{cases} \tag{2}$$

  Here, Equation (1) is the angular speed control, where w is the robot's angular speed, α_e is angular error and α_{eacc} is accumulation angular error over time.k_{α_e} and $k_{\alpha_{eacc}}$ are corresponding gains. Equation (2) is the linear speed control for forward motion, where k_r is the gain term and we set it to be 0.1 for a stable but robust approach(Schröder et al., 2023).

  Therefore, its integration with DRL aims to combine deterministic stability with adaptive learning capabilities, potentially enhancing overall navigational performance.

2.2. DRL Algorithms

To evaluate learning-based control strategies for spherical robot navigation, we examine both on-policy and off-policy deep-reinforcement-learning (DRL) methods:

2.2.1. On-Policy Algorithms

- **Vanilla Policy Gradient (VPG)** is the canonical gradient-ascent method that updates policy parameters via $\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\big[\nabla_\theta \log \pi_\theta(a_t|s_t)\, G_t\big]$(Williams, 1992; Sutton et al., 2000). While conceptually simple, its high-variance estimates slow convergence when spherical robots face frequent slip or slope changes.

- **Proximal Policy Optimization (PPO)** improves stability by constraining each update with a clipped surrogate loss(Schulman et al., 2017). Recent work further tailors PPO for safe mobile and spherical robot navigation under limited on-board compute(Zhang et al., 2024).

2.2.2. Off-Policy Algorithms

- **Deep Deterministic Policy Gradient (DDPG)** employs an actor–critic architecture with deterministic continuous actions and experience replay(Lillicrap et al., 2015). In spherical-robot motion-control studies, DDPG achieves centimetre-level path-tracking accuracy in MuJoCo simulations(Nor & Smith, 2024).

- **Twin Delayed DDPG (TD3)** mitigates the value over-estimation issue in DDPG by learning twin critics, adding target-policy smoothing and delaying actor updates(Fujimoto et al., 2018). These refinements yield smoother velocity commands and more reliable policies for precise spherical locomotion.

- **Deep Q-Network (DQN)** extends Q-learning with experience replay and a target network to learn value functions from high-dimensional inputs(Mnih et al., 2015). By discretizing torque or velocity commands into finite actions, DQN provides a baseline; recent visual-explanation studies show its decision saliency in robot navigation tasks(Kawai et al., 2022).

2.3. Simulation Framework

The simulation framework utilized in this study is built on an open-source plarform available on GitHub(Tomasvr), specifically designed for robotic navigations using deep reinforcement learning. The simulation environment integrates Robot Operating System 2 (ROS2), Gazebo, and Python within a Linux-based system. ROS facilitates communication and management, Gazebo provides a realistic simulation environment with physics-based interactions, and Python is used for implementing and executing the algorithms.

The maze environment used in simulations is fixed, presenting a structured but challenging navigation scenario. Additionally, dynamic complexity is introduced through the moving obstacles, which require the robot to continuously adapt its decision. The goal within the maze is represented as a dynamic coordinate, changing its position whenever the robot successfully reaches it or encounters a collision with obstacles or walls. This dynamic goal setting ensures that the algorithms and controllers are tested rigorously for adaptability and robustness under realistic conditions.

2.4. Implementation Summary

To achieve the objectives of this study, we extended the existing DRL framework by integrating a custom spherical controller with reinforcement learning policies written in Python. We adapted and fine-tuned standard DRL algorithms, including DDPG, TD3, and DQN to explore their effectiveness in the environment. To comprehensively assess the impact of off-policy learning strategies, we also developed and implemented on-policy algorithms, specifically PPO and VPG, allowing for a comparative analysis between on-policy and off-policy methods in the context of spherical robotic navigation.

3. Results

3.1. On-policy Model Performance

3.1.1. Reward-curve analysis, on-policy traits, and model suitability

The two panels in Fig. 3 report the *episode-average return* (smoothed over a 100-episode window) for the **PPO** and **VPG** agents[1]. The vertical axis is negative because the task uses a penalty-dominant reward shaping scheme; therefore *larger* (i.e. less negative) numbers indicate better performance.

Early learning (< 200 **episodes).** PPO emerges from an extremely poor initial policy, about-1.4×10^4, and crosses the -9×10^3 line within the first 150 episodes—a gain of roughly 5×10^3 reward units. The VPG curve instead *descends* from -2.5×10^3 to its nadir at about -3.1×10^3 during the same period. This contrast confirms the superior sample efficiency of PPO's surrogate-loss updates: trust-region clipping allows comparatively large yet stable policy steps, whereas the REINFORCE-style gradient in VPG must be kept tiny to avoid divergence, hence slow progress.

Mid-training dynamics ($200-2\,000$ **episodes).** Between episodes 200 and 2000, PPO's rolling mean drifts upward toward -8×10^3, while its $\pm1\sigma$ band remains wide, approximately $\pm1\,000$. This volatility stems from the alternating optimisation of policy and value networks: temporary critic over-estimation inflates the advantage and produces occasional large positive updates.

[1]Everything else—the environment, reward definition, network capacity and optimiser— is identical; only the update rule differs.

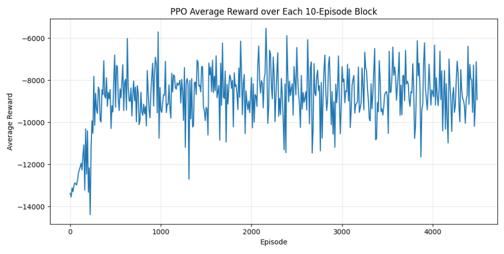

Figure 1. PPO

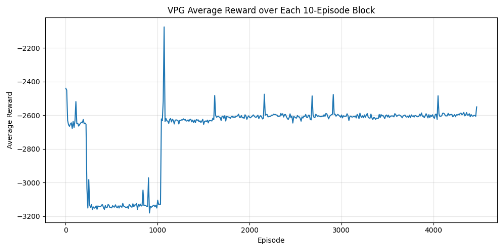

Figure 2. VPG

Figure 3. Episode-average return ($\pm1\sigma$ shaded). Axes are kept on their original scales to highlight each algorithm's own dynamics; absolute values are not directly comparable because the reward ranges differ between the two runs.

updates (the visible spikes). In contrast, VPG settles quickly: after episode 400 its mean return stabilises near -2.7×10^3 with a tightly bounded variance (±100). Regular thin spikes ($\Delta R \approx +200$) repeat every $\sim$ 500 episodes, suggesting the agent periodically encounters a rare bonus event but otherwise executes nearly identical trajectories.

Late-training behaviour ($> 2\,000$ **episodes).** PPO never completely suppresses variance: the envelope narrows only slightly, and sporadic outliers persist. Nevertheless the long-term trend is flat—indicating that learning saturates around -7.5×10^3. VPG's curve, by contrast, is practically a straight line after episode 700, signalling that the policy has fully converged (albeit to a mediocre operating point).

3.2. Raw Off-policy Model Performance

Figure 4 plots the block-averaged episode return (window $=$ 10)for three off-policy agents:DQN, TD3, and DDPG. A shaped reward of $+2\,500$ is granted on every successful navigation trial; therefore a block mean higher than $-2\,500$ implies that at least a fraction of the ten episodes ended in success, while values below that threshold correspond to either pure failures or exceedingly costly trajectories. Moreover, the success rates for the three off-policy DRL algorithms evaluated are summarized in Figure 5:

These results demonstrate that DQN achieved the highest success rate among the off-policy algorithms, followed by

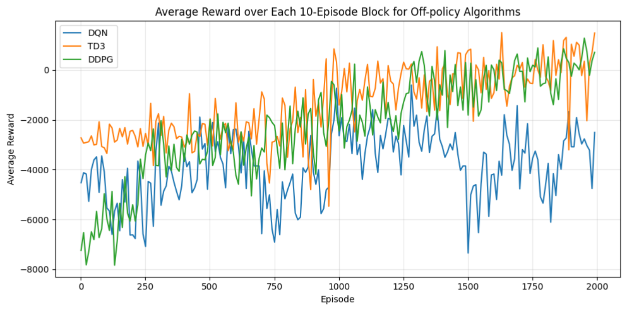

Figure 4. Performance of off-policy models

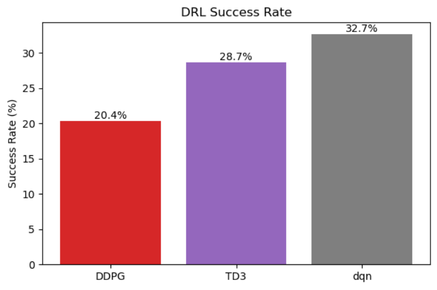

Figure 5. Success Rate of off-policy models

TD3 and DDPG. This suggests that value-based methods such as DQN may be more effective in discrete, structured environments like mazes, while actor-critic methods like DDPG may struggle due to their sensitivity to hyperparameters and exploration strategies.

Early phase (< 400 **episodes).** All three curves start in the range $[-7\,500, -3\,500]$, indicating collision-dominated exploration. DDPG (green) climbs fastest, benefiting from deterministic policy gradients that can exploit the continuous velocity space of the spherical robot. TD3 (orange) rises more carefully because twin critics and target-policy smoothing curb overly optimistic Q-values. DQN (blue) lags: its discrete action set cannot express subtle torque modulations required for balanced rolling, leading to slower reward accumulation.

Mid-training ($400-1200$ **episodes).** Both actor–critic methods cross the $-2\,500$ line around episode $1\,000$, meaning roughly one in ten roll-outs now reaches the goal without tumbling or timing out. Variance narrows as replay memory fills with near-successful trajectories, yet occasional outliers persist whenever an exploratory action triggers the success bonus. DQN remains largely below the threshold, reflecting the difficulty of learning fine-grained steering from sparse binary feedback in a high-dimensional state space.

Late phase ($> 1\,200$ **episodes).** TD3 and DDPG continue an overall upward drift and oscillate about the zero line, signalling that successful episodes outnumber failures despite the per-step penalties. The envelopes remain wide because a single success ($+2\,500$) superposes on a background of small negative rewards. DQN shows a modest positive slope but seldom exceeds $-2\,000$, implying success is sporadic and inconsistent

3.3. DRL with Controller Performance

Next, we try to integrate these three DRL off-policy algorithms with traditional model-based controller algorithms from Section 2.1. These two algorithms are blended during training process. By multiple days of training, the optimal percentage is 85% DRL with 15% controller. Sadly, due to their time and computing consuming characters, we can only report a preliminary results from the first 1000 episodes.

3.3.1. DRL with Spherical Controller

To evaluate the performance of various DRL algorithms integrated with a spherical controller, we combine three DRL algorithms with spherical controllers as mentioned in Section 2.1. In 8, DDPGS means the DDPG algorithm with spherical controller, DQNS means DQN algorithm with Spherical controller and TD3S means TD3 algorithm with Spherical controller.

Actor Loss (Figure 6): DDPG and TD3 show smoother and generally lower actor loss values during training compared to DQN, which exhibits high variance and elevated loss values.

Critic Loss (Figure 7): TD3 exhibits a significantly higher and more gradual loss of critics over time, while DDPG shows a moderate, more stable loss. DQN has no baseline setting, so it remains at zero. It is worth to note that the dramatic drop for DDPG with sphere controller at around 800 episodes is due to the unexpected disconnection from computer, which causes a loss of latest buffer.

Success Rate Histogram (Figure 8): Among all models, DDPG-P (DDPG with PID) achieved the highest success rate (9.0%), followed by DQNS (DQN with Sphere) (6.4%) and TD3S (TD3 with Sphere) (3.5%). All other variants were performed under 2%.

Cumulative Success Rate (Figure 9): The cumulative curve doubles ensures DQNS's early and consistent gains, while DDPG and TD3 show slower and less stable improvements over episodes.

Reward Distribution (Figure 10): TD3 and DQN exhibit more right-skewed reward distributions, indicating more frequent higher returns and success possibilities, while DDPG has a wider spread with heavier negative tails, which con-

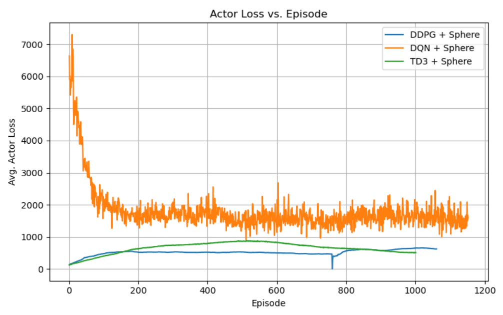

Figure 6. Actor Loss vs. Episode

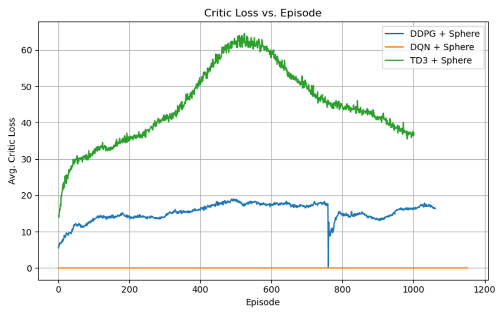

Figure 7. Critic Loss vs. Episode

firms its low success rate during training.

3.3.2. Blended DDPG Algorithm

This section evaluates the impact of controller choice: PID vs. Sphere, as well as training difficulty (Stage 4 vs. Stage 9) on the performance of DDPG-based algorithms. In Figure 11 and Figure 8, DDPGP stands for DDPG with PID controller in stage 4, DDPGS stands for DDPG with Spherical controller in stage 4, and DDPG9 stands for DDPG with Spherical controller in stage 9.

Learning Curves (Figure 11): The moving average episode returns for DDPG with different configurations are plotted. DDPGP shows a clear upward trend and better stability over time, outperforming DDPGS and the DDPG9, which displays highly unstable and flat returns.

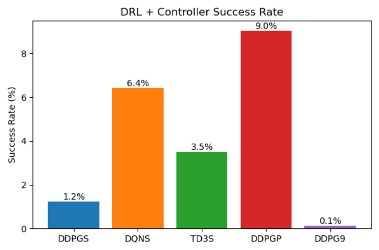

Figure 8. DRL + Controller Success Rate

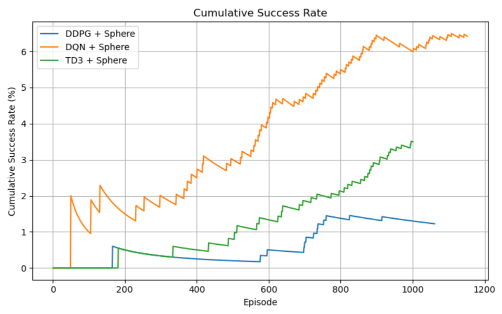

Figure 9. Cumulative Success Rate

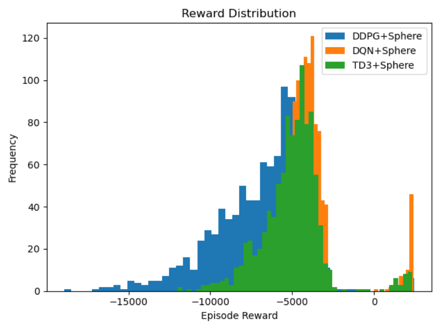

Figure 10. Reward Distribution Across Training Episodes

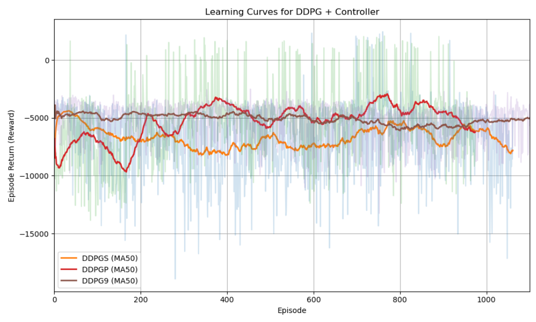

Figure 11. Learning Curves for DDPG with Different Controllers and Stages

Success Rate Comparison (Figure 8): Quantitatively, DDPGP achieves the highest success rate (9.0%), followed by DDPGS (1.2%). DDPG9 has the lowest success rate at only 0.1%, indicating severe learning failure.

4. Discussion

4.1. Failure of on-policy models

On-policy limitations and their effect. Both PPO and VPG are *on-policy*: each update relies solely on trajectories drawn from the *current* policy. In the present navigation environment, episodes last up to 10^3 time-steps and rewards are sparse; once an update is applied, those samples become obsolete and cannot be replayed. Consequently:

- **Data inefficiency.** The agent discards the bulk of the past experience, forcing it to regenerate thousands of expensive roll-outs. This throttles VPG's progress and leaves PPO sensitive to exploration noise.
- **High return variance.** Monte Carlo advantage estimates carry a large variance, manifested as the wide $\pm 1\sigma$ band and reward spikes in PPO; VPG mitigates variance only by shrinking its step size, trading learning speed for stability.

Task suitability. The spherical-robot navigation task demands two qualities:

1. **Learning efficiency**. Robots operate under strict battery- and experiment-time budgets. PPO meets this requirement, reaching a steady-state return after only $\sim$150–200 episodes, whereas VPG needs an order of magnitude more roll-outs before it stops degrading and still shows no clear upward trend.
2. **Behavioural consistency**. In real hardware, high-variance returns typically manifest as jerky motion or unsafe accelerations. VPG is better in this regard: once converged, its variance band is narrow and the curve free of large jumps. PPO's persistent $\pm 1\,000$ swings could translate to unpredictable accelerations unless mitigated by additional safeguards (e.g., online monitoring).

Because of their on-policy nature, both algorithms struggle to reuse data, and reward variance remains high in PPO or learning is slow in VPG. For simulation-phase training—where speed of improvement is paramount and safety is virtual—*PPO is preferable*. For direct deployment on the physical platform, the low-variance yet low-return *VPG* policy is safer but unlikely to achieve the desired navigation success rate. In short, on-policy methods alone appear *insufficient for optimal spherical-robot navigation*, motivating the exploration of more data-efficient off-policy or hybrid approaches.

4.2. Success on raw off-policy models

Why these trends emerge in spherical-robot navigation.

- **Continuous actions matter.** Rolling locomotion requires precise torque adjustments; actor–critic algorithms (DDPG, TD3) output real-valued commands directly, while DQN must approximate them with a coarse discrete set.
- **Sample reuse counters sparsity.** Off-policy replay allows TD3 and DDPG to learn from rare success transitions repeatedly, accelerating value propagation across the state space. DQN shares this advantage, but is further hampered by its discrete policy representation.
- **Robustness to Q-value bias.** TD3's conservative twin-critic update curbs over-estimation, which is crucial when the large terminal bonus ($+2\,500$) can otherwise distort value targets and destabilize learning. This explains its smoother ascenreachesred to the noisier trajectory of DDPG.

Crossing the $-2\,500$ mark denotes the onset of a reliable goal achievement. TD3 reaches and maintains that level with the lowest variance, making it the most promising candidate for real-world deployment. DDPG is competitive but occasionally unstable, while DQN's discrete action bottleneck leaves it ill-suited for the continuous control demands of spherical robot navigation.

4.3. Integration of off-policy model with controllers

Based on Figures 6 to 11 , it is evident that different DRL algorithms show distinct learning characteristics when integrated with a spherical controller:

- The high variance in DQN's actor loss suggests instability during policy updates, possibly due to value overestimation or uncoordinated exploration. However, it still has a higher success rates, indicating effective exploitation in some scenarios.
- TD3's high critic loss and moderate performance might come from its sensitivity to hyperparameter tuning or insufficient exploration in the setup. Despite its promise, it performs better DQN in both reward and success.
- DDPG shows relatively stable loss behavior but a broad and negative reward distribution. However, DDPG with PID controller still facilitated better convergence

and performance compared to the corresponding spherical controller (DDPGS). This suggests that the simpler and more stable control dynamics of PID may be more compatible with the DDPG learning process, particularly in early training stages.

- The extremely low performance of DDPG9, which was trained on Stage 9, indicates that the increased task complexity, more moving obstacles in this case, likely overwhelmed the exploration capacity of the algorithm. This also suggests that curriculum learning or stage-wise progression may be necessary for scaling up training difficulty effectively.
- Interestingly, even though DDPGS was trained on the same stage as DDPGP, its success rate and return remain lower, which underscores the idea that controller design significantly affects policy learning.
- Overall, DQNS appears to offer the best trade-off between success rate consistency and learning efficiency, though none of the algorithms achieved high success rates across all episodes. This suggests potential benefits in hybrid strategies, further tuning, or exploration enhancement.

5. Conclusion

In all trials, the off-policy methods were consistently stronger than the on-policy ones. TD3 and DDPG cleared the success mark after about 1000 episodes and maintained higher, less-negative returns; PPO leveled off well below that, and VPG barely moved from its starting point. Two factors explain the gap. Replay buffers allow off-policy agents to recycle infrequent successful rollouts, letting the rare +2 500 terminal bonus permeate value estimates instead of vanishing with the next policy update. In addition, their continuous-action actors can express the subtle torque modulations a spherical robot needs to stay balanced, something discrete-action DQN cannot match and on-policy gradients only glimpse before data age out. The result is faster skill acquisition and better long-run reward for off-policy learners. Among the hybrid approaches, DDPG combined with PID showed the most consistent improvement, suggesting that low-level feedback control can enhance stability during learning. Separately, TD3 as a pure off-policy method achieved the highest overall return, while DQN paired with a spherical controller also demonstrated strong performance. These results suggest that while hybrid methods can offer added robustness, well-tuned off-policy algorithms like TD3 may be more effective when extensive training is feasible.

References

Escorza, O., Garcia, G., Fabregas, E., Velastin, S. A., Eskandarian, A., and Farias, G. Deep reinforcement learning applied to a spherical robot for target tracking. *IEEE Transactions on Industrial Electronics*, pp. 1–10, 2025. doi: 10.1109/TIE.2025.3557993.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 1587–1596, 2018.

Kawai, Y., Tanaka, S., and Yamashita, A. Visualizing deep q networks for mobile robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8345–8352, 2022. doi: 10.1109/IROS47612.2022.9981683.

Keymasi-Khalaji, A., Mokhtari, P., and Bathaei, F. Predictive control for the navigation of spherical robots in obstacle-rich environments. *Scientific Reports*, 15(1): 13859, Apr 2025. ISSN 2045-2322. doi: 10.1038/s41598-025-96521-6. URL https://doi.org/10.1038/s41598-025-96521-6.

Li, X., Lu, H., Liang, X., and Wang, L. Attitude control of spherical robot based on reinforcement learning. In Yan, L., Duan, H., and Deng, Y. (eds.), *Advances in Guidance, Navigation and Control*, pp. 1248–1257, Singapore, 2023. Springer Nature Singapore. ISBN 978-981-19-6613-2.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., et al. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, 2015. doi: 10.1038/nature14236.

Nor, A. B. and Smith, J. High-precision path tracking of spherical robots using ddpg. *Journal of Intelligent & Robotic Systems*, 110(2):201–219, 2024. doi: 10.1007/s10846-024-01987-7.

Schröder, K., Garcia, G., Chacón, R., Montenegro, G., Marroquín, A., Farias, G., Dormido-Canto, S., and Fabregas, E. Development and control of a real spherical robot. *Sensors*, 23(8), 2023. ISSN 1424-8220. doi: 10.3390/s23083895. URL https://www.mdpi.com/1424-8220/23/8/3895.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pp. 1057–1063, 2000.

Tomasvr. A ROS2-based Framework for TurtleBot3 DRL Autonomous Navigation. URL `https://github.com/tomasvr/turtlebot3_drlnav`.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. doi: 10.1007/BF00992696.

Zhang, H., Chen, L., and Gupta, R. Safe and sample-efficient ppo for spherical robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1234–1240, 2024.

Zhao, C., Wang, D., and Xue, W. Beyond linear limits: Design of robust nonlinear pid control. *Automatica*, 173:112075, 2025. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2024.112075. URL `https://www.sciencedirect.com/science/article/pii/S0005109824005685`.

A. Code Availability

The simulation platform is built based on ”turtlebot3_drlnav” github page. The tailored script for this study can be found at `https://github.com/JameZ233/RL_Sphere`

B. Work Division

James Zhang: Code Development and Simulation (DRL + Controller), Report Writing
Haotian Zhang: Code Development and Simulation (general on-policy/off-policy models), Report Writing