Journal of Mechatronics and Robotics

Research Article

A Lightweight PyBullet-Based Framework for Fast Reinforcement Learning Prototyping on 6-DOF Robotic Arms

Ngoc Kim Khanh Nguyen[1], Anh Thu Mang[2] and Quang Nguyen[3,4]

[1]*Faculty of Basic Sciences, Van Lang University, Ho Chi Minh City, Vietnam*
[2]*eWalk Co. Ltd., Ho Chi Minh City, Vietnam*
[3]*Department of Physics, International University, VNU-HCM, Ho Chi Minh City, Vietnam*
[4]*Viet Nam National University, Ho Chi Minh City, Vietnam*

Article history
Received: 28-08-2025
Revised: 16-12-2025
Accepted: 20-01-2026

Corresponding Author:
Ngoc Kim Khanh Nguyen
Faculty of Basic Sciences, Van Lang University, Ho Chi Minh City, Vietnam
Email: khanh.nnk@vlu.edu.vn

Abstract: Controlling 6-Degree-of-Freedom (6-DOF) robotic arms for precise manipulation tasks is challenging due to kinematic redundancy and the complexity of existing simulation environments like MuJoCo or ROS-Gazebo. This paper presents ArmReach6DOFEnv, a lightweight, open-source simulation framework built on PyBullet for rapid Reinforcement Learning (RL) prototyping on 6-DOF robotic arms. Using a Universal Robot Description Format (URDF) model, the environment supports a continuous state-action space for a 3D reaching task, with a reward function balancing accuracy and control effort. We evaluate two state-of-the-art RL algorithms, Proximal Policy Optimization (PPO) and Deep Deterministic Policy Gradient (DDPG), implemented via Stable-Baselines3, comparing their convergence, success rate, and motion smoothness. Experimental results demonstrate DDPG's superior performance (69% success rate vs. PPO's 34%) and smoother trajectories, despite PPO's faster convergence. This framework enables accessible RL experimentation on resource-constrained systems, with potential for future sim-to-real transfer.

Keywords: Reinforcement Learning, 6-DOF Robotic Arm, PyBullet, PPO, DDPG, Rapid Prototyping

Introduction

Six-Degree-of-Freedom (6-DOF) robotic arms are essential in applications such as industrial automation, medical robotics, and service robotics due to their ability to perform complex manipulation tasks in 3D space (Chiaverini *et al*., 2008). However, tasks like reaching a specific 3D point introduce kinematic redundancy, as the six revolute joints exceed the three degrees of freedom required for positioning (Craig, 2017). This redundancy complicates traditional Inverse Kinematics (IK) solvers, which struggle with selecting optimal configurations, handling singularities, and incorporating dynamic constraints like energy efficiency or joint limits (Aristidou and Lasenby, 2009). Existing simulation environments, such as MuJoCo (Todorov *et al*., 2012) or ROS-Gazebo (Koenig and Howard, 2004), often require significant computational resources and complex setups, limiting their accessibility for rapid RL prototyping (Zhu *et al*., 2020).

Reinforcement Learning (RL) offers a promising alternative by learning control policies through trial-and-error, optimizing reward functions that balance multiple objectives without explicit IK computation (Levine *et al*., 2016). Despite its potential, RL adoption for 6-DOF arm control is hindered by the lack of lightweight, user-friendly simulation platforms. To address this gap, we propose ArmReach6DOFEnv, a PyBullet-based (Coumans and Bai, 2016), Gym-compatible environment tailored for rapid RL prototyping on a 6-DOF reaching task. Integrated with Stable-Baselines3 (Raffin *et al*., 2021), our framework uses a URDF-modeled arm and evaluates two RL algorithms: PPO (Schulman *et al*., 2017) and DDPG (Fujimoto *et al*., 2018). Our contributions include:

(1) An open-source, lightweight environment for 6-DOF arm reaching tasks
(2) A comparative analysis of PPO and DDPG in handling kinematic redundancy
(3) A platform for accessible RL experimentation with potential sim-to-real applications

This paper is organized as follows: The second reviews related work on IK, RL, and simulation environments. The third section details the ArmReach6DOFEnv methodology, including the kinematic model. The forth section presents experimental results, comparing PPO and DDPG. The fifth section discusses findings, limitations, and future directions. The final section concludes with objectives achieved, limitations, and implications.

Related Work

Kinematic redundancy in 6-DOF robotic arms, where multiple joint configurations achieve the same end-effector position, enables optimization of secondary objectives like energy efficiency or obstacle avoidance (Chiaverini *et al*., 2008). Kinematic redundancy refers to the excess degrees of freedom in a manipulator relative to the task requirements (Craig, 2017). Traditional IK methods include analytical solutions, which require manual configuration selection (Craig, 2017), and numerical methods, like Jacobian-based solvers, which are computationally expensive and sensitive to singularities (Aristidou and Lasenby, 2009). Advanced techniques, such as quadratic programming (Kanoun *et al*., 2011), demand problem-specific tuning, limiting rapid prototyping.

Reinforcement Learning (RL) is a learning paradigm where an agent optimizes a reward function through trial-and-error (Sutton and Barto, 2018). RL has shown promise in robotic manipulation, with PPO (Schulman *et al*., 2017) offering stability in stochastic environments and DDPG (Fujimoto *et al*., 2018) excelling in continuous control (Zeng *et al*., 2018). Studies like (Kalakrishnan *et al*., 2011) leverage redundancy for tasks like obstacle avoidance but often use complex setups. Our framework simplifies this by focusing on a lightweight reaching task.

Simulation environments are critical for RL. MuJoCo (Todorov *et al*., 2012) offers accurate physics but high computational demands. ROS-Gazebo (Koenig and Howard, 2004) integrates with robotic software but requires complex configurations (Quigley *et al*., 2009). OpenAI Gym (Brockman *et al*., 2016) lacks lightweight 6-DOF arm options. Robosuite (Zhu *et al*., 2020) and PyRobot (Murali *et al*., 2019) focus on specific robots, reducing generalizability. In contrast, PyBullet (Coumans and Bai, 2016) provides a lightweight, open-source platform, and Stable-Baselines3 (Raffin *et al*., 2021) streamlines RL implementation. Our ArmReach6DOFEnv combines these for rapid, accessible RL prototyping.

Materials and Methods

ArmReach6DOFEnv is a custom Gym-compatible environment built using PyBullet (Coumans and Bai, 2016) for RL prototyping on a 6-DOF robotic arm reaching task. This section describes the robot's kinematic model, environment design, and RL agents.

Robot Kinematic Model

The 6-DOF arm is modeled using a URDF file, encapsulating kinematic and dynamic properties. The arm consists of six revolute joints, each with limits derived from realistic designs (Craig, 2017). The forward kinematics map joint angles $\theta = [\theta_1, \theta_2, \ldots, \theta_6] \in \mathbb{R}^6$ to the end-effector position $p_e = [x, y, z] \in \mathbb{R}^3$ via:

$$p_e = f(\theta) = T_1(\theta_1)T_2(\theta_2)\ldots T_6(\theta_6) \cdot p_0, \quad (1)$$

Where $T_i(\theta_i)$ are homogeneous transformation matrices, and p_0 is the base frame origin. The state dynamics are governed by:

$$\dot{\theta} = u, p_e = f(\theta), \ \dot{p}_e = J(\theta)\dot{\theta}, \quad (2)$$

Where $u \in \mathbb{R}^6$ is the control input (joint velocities), and $J(\theta)$ is the Jacobian matrix. Physical parameters (e.g., link masses) are approximated for lightweight arms, with gravity and contact dynamics enabled in PyBullet (Coumans and Bai, 2016).

Environment Design: ArmReach6DOFEnv

The environment simulates a reaching task where the arm's end-effector moves to a 3D target position.

The RL workflow, illustrated in Fig. 1, shows the interaction between the RL agent, the PyBullet environment, and the reward function, providing a clear overview of the learning process.

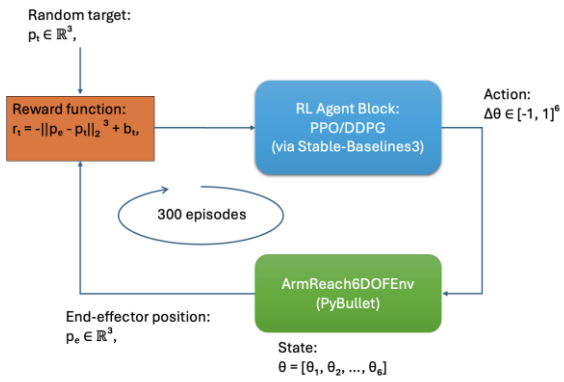

Fig. 1: Reinforcement Learning Workflow in ArmReach6DOFEnv. The RL agent (PPO or DDPG) interacts with the PyBullet-based environment, receiving states (joint angles θ, end-effector position p_e, target position p_t) and outputting actions ($\Delta\theta$). The environment updates the 6-DOF arm's state, and the reward function computes $r_t = -d_t^3 + b_t$, where $d_t = \|p_e - p_t\|_2$ is the Euclidean distance to the target and b_t is a success bonus if $d_t < 0.05$ m

DOI: 10.3844/jmrsp.2025.35.39

The environment's key components are.

State Space: A continuous vector comprising:

- Joint angles: $\theta \in \mathbb{R}^6$, constrained within limits
- End-effector position: $p_e \in \mathbb{R}^3$, computed via forward kinematics
- Target position: $p_t \in \mathbb{R}^3$, randomized in a spherical workspace (radius 0.5m)
- Joint velocities: $\dot{\theta} \in \mathbb{R}^6$, capturing dynamics

Action Space: A continuous 6D vector $\Delta\theta \in [-1, 1]^6$, representing relative changes in joint angles, scaled by a step size and applied via PyBullet's position control (Coumans and Bai, 2016).

Further description of the State and Action space can be found in Table 1.

Reward Function: The reward at timestep t is:

$$r_t = -d_t^3 + b_t \quad (3)$$

Where $d_t = \|p_e - p_t\|_2$ is the Euclidean distance between the end-effector (p_e) and target (p_t), and $b_t = 10.0$ is a sparse success bonus if $d_t < 0.05$m (5cm), terminating the episode. Episodes also terminate after 2048 timesteps or if joint limits are violated (Levine *et al*., 2018).

Dynamics: The environment resets with a randomized target in a 0.5 m-radius spherical workspace, a fixed initial joint configuration, and zero velocities. PyBullet enforces collision detection and joint limits (Coumans and Bai, 2016).

RL Agents

PPO and DDPG are implemented using Stable-Baselines3 (Raffin *et al*., 2021). PPO uses a Multilayer Perceptron (MLP) with two 128-unit hidden layers (ReLU activation), with hyperparameters: Learning rate 3×10^{-4}, discount factor $\gamma = 0.99$, clipping 0.2, batch size 300, and updates every 2048 steps (Schulman *et al*., 2017). DDPG uses MLPs with two 256-unit hidden layers, with learning rate 10^{-3}, $\gamma = 0.99$, soft target update $\tau = 0.005$, batch size 300, and Ornstein-Uhlenbeck noise ($\sigma = 0.1$) (Fujimoto *et al*., 2018). Training runs for 614,400 timesteps (300 episodes, 2048 timesteps each).

Table 1: State and action space components

Component	Description
Joint Angles	6D vector, $\theta \in \mathbb{R}^6$, within joint limits
End-Effector Position	3D Cartesian coordinates, $p_e \in \mathbb{R}^3$
Target Position	3D coordinates, $p_t \in \mathbb{R}^3$, randomized
Joint Velocities	6D vector, $\dot{\theta} \in \mathbb{R}^6$
Action	6D vector, $\Delta\theta \in [-1, 1]^6$, relative joint angle changes

Experiments

Environment Setup

The 6-DOF arm is simulated in PyBullet (Coumans and Bai, 2016). The task involves reaching a 3D target within a 5 cm radius, a tolerance selected to balance accuracy and convergence speed under our computational constraints. The state includes joint angles, velocities, target position, and end-effector-to-target vector. Actions are relative joint angle changes. Training runs for 300 episodes (justified for sufficient RL exploration (Schulman *et al*., 2017)), each capped at 2048 timesteps (aligned with Stable-Baselines3 defaults (Raffin *et al*., 2021)), on an NVIDIA 3060 RTX Mobile GPU using PyTorch 2.1.

Learning Performance and Convergence

As we can see in Fig. 2, PPO converges faster (episode 177) but achieves a lower final reward (3.0) compared to DDPG (episode 294, reward 4.95). Convergence is defined as the moving average reward staying within ±5% of its final value for 100 episodes.

Table 2 shows DDPG's higher reward but slower convergence, with PPO exhibiting greater variability.

Success Rate and Goal Reaching

Post-training, policies were tested on 100 episodes with random targets. Success is defined as reaching within 5 cm.

DDPG achieves a 69% success rate, significantly outperforming PPO's 34%, with comparable steps to goal (Table 3). Figure 3 illustrates a success state of the arm.

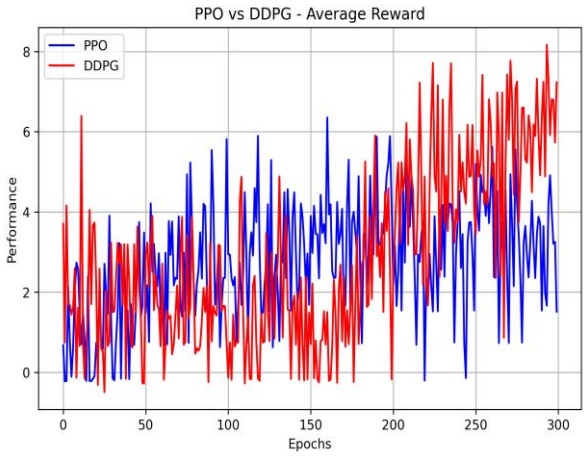

Fig. 2: Learning curves: Average episode reward vs. training episodes for PPO (blue) and DDPG (orange)

Table 2: Reward and convergence statistics

Algorithm	Convergence Episode	Final Average Reward
PPO	177±25	3.0±0.4
DDPG	294±18	4.95±0.3

DOI: 10.3844/jmrsp.2025.35.39

Joint Trajectory Smoothness

Joint trajectories in successful episodes were analyzed for smoothness via jerk (time derivative of acceleration).

DDPG maintains lower jerk (0.02 rad/s³) compared to PPO (0.11 rad/s³), indicating smoother, more physically plausible trajectories (Fig. 4).

Table 3: Goal-Reaching Performance

Algorithm	Success Rate (%)	Average Steps to Goal
PPO	34	1500±200
DDPG	69	1450±180

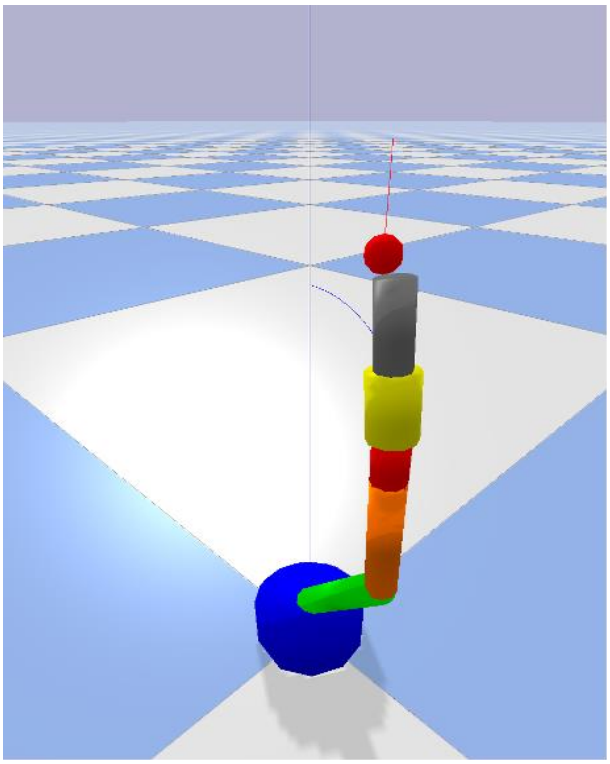

Fig. 3: Simulated 6-DOF arm reaching task in PyBullet, showing the arm's trajectory toward a red spherical target in 3D space

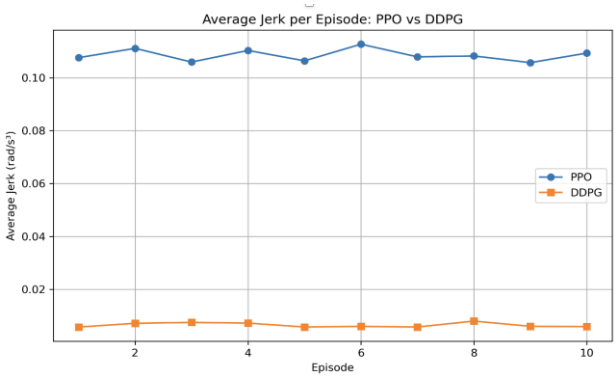

Fig. 4: Average joint jerk over time (rad/s³) for PPO (blue) and DDPG (orange). Lower values indicate smoother motion

Summary and Insights

The experiments reveal a trade-off between PPO and DDPG. DDPG outperforms PPO in success rate (69 vs. 34%), final reward (4.95 vs. 3.0), and motion smoothness (jerk 0.02 vs. 0.11 rad/s³), as summarized in Table 4. PPO converges faster (episode 177 vs. 294), suitable for rapid prototyping, but yields suboptimal policies. DDPG's off-policy nature and replay buffer enable better exploration of the sparse reward function, leading to more reliable and smoother policies (Fujimoto *et al*., 2018).

Table 4: Goal-Reaching Performance

Algorithm	Success Rate (%)	Final Reward	Average Jerk (rad/s³)
PPO	34	3.0	0.11
DDPG	69	4.95	0.02

Discussion

The results align with prior RL studies, where off-policy algorithms like DDPG excel in continuous control due to experience replay (Fujimoto *et al*., 2018; Zeng *et al*., 2018). Compared to complex simulators like MuJoCo (Todorov *et al*., 2012) or IsaacGym, PyBullet's lightweight design enables accessible prototyping (Coumans and Bai, 2016). Limitations include the focus on a single reaching task, lack of robustness analysis (e.g., noise sensitivity), and absence of sim-to-real validation (Zhu *et al*., 2020). Future work will extend the framework to complex tasks (e.g., grasping, obstacle avoidance), test robustness, and validate on physical arms (Tassa *et al*., 2018).

Conclusion

This study achieved its objectives of developing ArmReach6DOFEnv and comparing PPO and DDPG, with DDPG demonstrating superior success rate (69 vs. 34%), reward (4.95 vs. 3.0), and smoothness (jerk 0.02 vs. 0.11 rad/s³). The lightweight framework enables rapid RL prototyping on resource-constrained systems, with practical implications for accessible research and managerial benefits for cost-effective robotic development. Limitations include the single-task focus and lack of sim-to-real tests. Future work will explore multi-task scenarios, robustness, and hardware validation.

Acknowledgment

The authors gratefully acknowledge Van Lang University (Ho Chi Minh City, Vietnam) for proving their necessary support for this study.

Funding Information

This research was funded by eWalk.vn.

Author's Contributions

Ngoc Kim Khanh Nguyen: Analysis, methodology, visualization, writing.

Anh Thu Mang: Data collection, analysis, visualization, writing.

Quang Nguyen: Supervision, project administration, reviewing, methodology.

Ethics

All of the other authors have read and approved the manuscript, and no ethical issues are involved.

References

Aristidou, A., & Lasenby, J. (2009). *Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *ArXiv Preprint*, *1*, 1–6. https://doi.org/https://doi.org/10.48550/arXiv.1606.01540

Chiaverini, S., Oriolo, G., & Walker, I. D. (2008). Kinematically Redundant Manipulators. *Springer Handbook of Robotics*, 245–268. https://doi.org/10.1007/978-3-540-30301-5_12

Coumans, E., & Bai, Y. (2016). PyBullet, a Python module for physics simulation for games, robotics and machine learning. *OpenAI / Community Contributors*. https://docs.google.com/document/d/10sXEhzFRSnvFcl3XxNGhnD4N2SedqwdAvK3dsihxVUA.

Craig, J. J. (2017). *Introduction to Robotics: Mechanics and Control*.

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. *Proceedings of the 35th International Conference on Machine Learning (ICML*, 1587–1596. https://doi.org/10.48550/arXiv.1802.09477

Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., & Schaal, S. (2011). Learning, planning, and control for quadruped locomotion over challenging terrain. *The International Journal of Robotics Research*, *30*(2), 236–258. https://doi.org/10.1177/0278364910388677

Kanoun, O., Lamiraux, F., & Wieber, P.-B. (2011). Kinematic Control of Redundant Manipulators: Generalizing the Task-Priority Framework to Inequality Task. *IEEE Transactions on Robotics*, *27*(4), 785–792. https://doi.org/10.1109/tro.2011.2142450

Koenig, N., & Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2149–2154. https://doi.org/10.1109/iros.2004.1389727

Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, *17*(1), 1334–1373. https://doi.org/10.5555/2946645.2946684

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., & Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, *37*(4–5), 421–436. https://doi.org/10.1177/0278364917710318

Murali, A., Chen, T., Alwala, K. V., Gandhi, D., Pinto, L., Gupta, S., & Gupta, A. (2019). PyRobot: An open-source robotics framework for research and benchmarking. *ArXiv:1906.08236*, *1*, 1–10. https://doi.org/https://doi.org/10.48550/arXiv.1906.08236

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Andrew, N. (2009). ROS: An open-source Robot Operating System. *Proceedings of the IEEE International Conference on Robotics and Automation*, 229–234.

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*(1), 12348–12355. https://doi.org/10.5555/3546258.3546526

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *ArXiv Preprint*, *1*, 1–13. https://doi.org/10.48550/arXiv.1707.06347

Todorov, E., Erez, T., & Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. https://doi.org/10.1109/iros.2012.6386109

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Las Casas, D. de, Guy, D., & Jaderberg, M. (2018). DeepMind control suite. *ArXiv Preprint ArXiv:1801.00690*, *1*, 1–20. https://doi.org/10.48550/arXiv.1801.00690

Zeng, A., Song, S., Welker, S., Lee, J., Rodriguez, A., & Funkhouser, T. (2018). Learning Synergies Between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning. *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4238–4245. https://doi.org/10.1109/iros.2018.8593986

Zhu, Y., Wong, J., Mandlekar, Ajay, Martin, R. M., Joshi, A., Lin, K., Maddukuri, A., Nasiriany, S., & Zhu, Y. (2020). Robosuite: A modular simulation framework and benchmark for robot learning. *ArXiv Preprint ArXiv:2009.12293*, *1*, 1–15. https://doi.org/10.48550/arXiv.2009.12293