IJRCS

International Journal of Robotics and Control Systems
Vol. 5, No. 3, 2025, pp. 1661-1677
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

ASCEE

Dynamic Ball Balancing Using Deep Deterministic Policy Gradient (DDPG)-Controlled Robotic Arm for Precision Automation

K Vijaya Lakshmi [a,b,1,*], M Manimozhi [b,2], J Vimala Kumari [c,3]

[a] Department of EIE, Siddhartha Academy of Higher Education, Deemed to be University, Vijayawada, India
[b] School of Electrical Engineering, Vellore Institute of Technology, Vellore, Tamilnadu, India
[c] Department of EEE, Siddhartha Academy of Higher Education, Deemed to be University, Vijayawada, India
[1] vijaya.korupu@gmail.com; [2] mmanimozhi@vit.ac.in; [3] vimalakumari@vrsiddhartha.ac.in
* Corresponding Author

ARTICLE INFO

Article history
Received April 27, 2025
Revised June 10, 2025
Accepted July 07, 2025

Keywords
Robotic Arm;
Adaptive Control;
Reinforcement Learning;
DDPG;
SAC;
Automation

ABSTRACT

This paper presents a reinforcement learning (RL)-based solution for dynamic ball balancing using a robotic arm controlled by the Deep Deterministic Policy Gradient (DDPG) algorithm. The problem addressed is maintaining ball stability under external disturbances in automated manufacturing. The proposed solution enables adaptive, precise control on flat surfaces. The research contribution is a comparative evaluation of DDPG and Soft Actor-Critic (SAC) algorithms for trajectory control and stabilization. A simulated environment is used to train the RL agents across multiple initial ball positions. Key performance metrics-settling time, rise time, overshoot, and steady-state error-are analyzed. Results show DDPG outperforms SAC with smoother trajectories, ~25% faster settling times, and significantly lower overshoot and steady-state errors. Visual analysis confirms that DDPG consistently drives the ball to the center with minimal deviation. These findings highlight DDPG's advantages in control accuracy and stability. In conclusion, the DDPG-based approach proves highly effective for precision automation tasks where fast, stable, and reliable control is essential.

1. Introduction

A dynamic ball balancing system is a highly unstable and nonlinear system used in robotics and manufacturing [1]. This system stabilizes a ball on a plate by adjusting its tilt along the x and y axes using sensors and actuators and controlled by two servo motors. The goal is to precisely position the ball on the plate at the center by applying proper control voltages to the motors to adjust the movements of robotic arm. Through the use of sensors, the position of the ball is continuously measured by the robotic arm. The position of the ball is tracked in real time using computer vision technique and used as feedback. This system ensures stability and safety in processes like material handling, and conveyor operations, and can handle delicate tasks like grasping fragile objects or assembling intricate components preventing disturbances or accidents [2].

Common causes of ball imbalance by a robotic arm include sensor noise, actuator dynamics, environmental disturbances, model inaccuracies, suboptimal control tuning, hardware limitations, and

communication delays [3], [4]. Addressing these challenges requires precise calibration, robust algorithms, adaptive controls, and real-time feedback to mitigate uncertainties and improve performance.

In the realm of dynamic ball balancing using robotic arms, several control strategies have been reported in the literature. These controllers aim to enhance the stability and maneuverability of ball-balancing robots, each with its own set of advantages and limitations. Conventional control techniques such as PID controllers, fuzzy logic, and Model Predictive Control (MPC) have been extensively applied to such systems [5]-[7]. PID controllers are known for their simplicity but often fall short in handling the nonlinear dynamics and disturbances present in real-world scenarios [8]-[10]. Fuzzy logic offers some adaptability through rule-based systems but becomes difficult to scale and tune for complex tasks [11]-[13]. MPC provides better performance by predicting future system behavior, but its reliance on accurate system models and high computational demand can be limiting, especially for real-time control [14]-[16]. While hybrid control methods-like fuzzy-PID or fractional-order controllers-have been proposed to improve robustness and adaptability, they typically involve complex tuning processes and may still struggle in the presence of external disturbances and modeling inaccuracies [17]-[19].

Reinforcement Learning (RL) provides a different approach by learning control strategies directly from interaction with the system, without needing a precise model. In particular, Deep Deterministic Policy Gradient (DDPG) has emerged as a powerful RL algorithm for handling continuous control tasks. DDPG combines value-based and policy-based methods, making it suitable for complex, high-dimensional systems such as robotic manipulators [20]. Earlier RL approaches like Q-learning and SARSA have had some success but are mainly limited to discrete action spaces and do not scale well to real-time robotic control [21]. Moreover, there is often a gap between training in simulation and deployment in physical systems due to differences in dynamics and environmental factors, known as the Sim2Real gap [22], [23]. Table 1 presents a comparison between traditional control methods and reinforcement learning-based approaches, highlighting their respective strengths and limitations in managing complex dynamic systems.

Table 1. Comparison of traditional and RL based control methods

Control Method	Adaptability	Handles Nonlinear Dynamics	Ease of Tuning	Real-Time Performance	Key Limitations
PID	Low	Poor	Moderate	High	Struggles with disturbances and nonlinearity
Fuzzy Logic	Moderate	Good	Low (rule tuning)	Moderate	Complex rule base, scalability issues
Model Predictive Control (MPC)	High	Good	Low (model dependent)	Low (computationally intensive)	Requires accurate system model, limited real-time use
Traditional RL	High	High	High (auto-learned)	Variable	Sim-to-real gap, stability issues
DDPG (This work)	High	Excellent	High	High	Sensitive to hyper parameters
SAC	Very High	Excellent	High	High	Less smooth control but better exploration

This work addresses these challenges by applying a DDPG-based controller to a ball balancing task using the Kinova Gen3 robotic arm. The study also includes a performance comparison with Soft Actor-Critic (SAC), another widely used RL algorithm. The control strategies are tested in MATLAB/Simulink under various initial conditions to evaluate their robustness and adaptability. The key contributions of this work are:

1. Development of an RL-based controller using DDPG for a dynamic ball and plate system with a robotic arm.
2. Comparative evaluation of DDPG and SAC in terms of stability, convergence, and trajectory tracking.
3. Validation of the proposed approach under different initial positions, demonstrating its potential for real-time applications.

2. Method

2.1. Deep Deterministic Policy Gradient (DDPG) Algorithm in RL

The methodology used for dynamic ball-balancing using DDPG controlled robotic arm involves various steps such as defining the problem, developing accurate model of the environment in Simulink, defining state space, action space, reset and reward functions. Then, the RL agent created in MATLAB is trained in the Simulink environment, by proper tuning of hyper parameters. Lastly, its performance is tested using robot physical model created in Simscape [24].

Reinforcement learning (RL) is a machine learning technique in which an agent tries to reach its goal by continuous interactions with the environment [25]. The agent receives environment states and improves its actions based on reward or penalty received from the environment. Algorithms such as Q-learning, SARSA, DDPG, and PPO, helps to guide this learning process [26]. Fig. 1 shows the typical RL work flow. The agent consists of a policy for action selection based on environmental observations, represented by a function approximator, and a learning algorithm that updates this policy using actions, observations, and rewards to maximize cumulative reward [27].

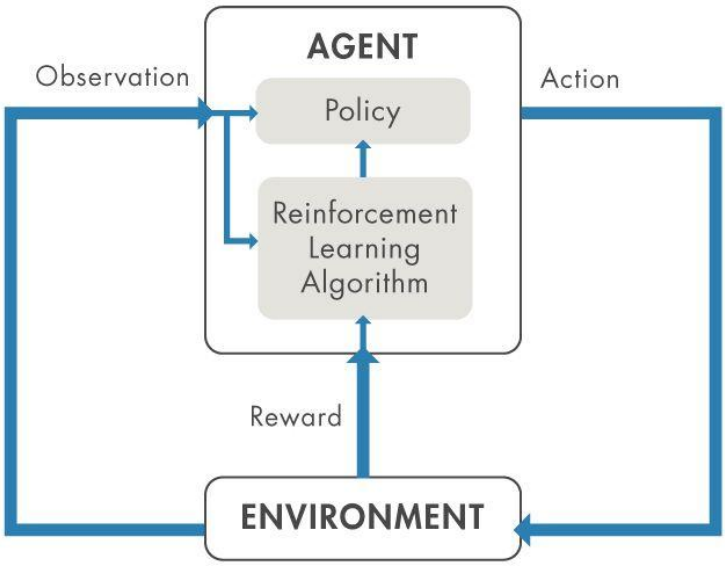

Fig. 1. RL flow

Reinforcement learning algorithms are divided into model-free and model-based approaches as shown in Fig. 2 [28]. Model-free methods, such as Q-learning and policy gradients, learn directly from experience to find a policy or value function, aiming to maximize long-term rewards. Model-based methods, like Monte Carlo Tree Search (MCTS), build a model of the environment to predict future states and rewards. While model-free methods are more sample-efficient, model-based methods can achieve better performance with less data, although they may struggle with model inaccuracies [29].

Model free RL can be broadly categorized into on-policy and off-policy methods. On-policy methods, such as SARSA, directly update the agent's policy using actions observed from the policy

itself, resulting in stable learning but requiring more data to converge [30]. In contrast, off-policy methods, like Q-learning, learn from experiences generated by different policies, allowing for more efficient exploration and sample use. The main difference lies in whether the policy is updated using actions from its own policy (on-policy) or from a different policy (off-policy) [31]. The choice of method depends on the specific task requirements, balancing stability and learning speed.

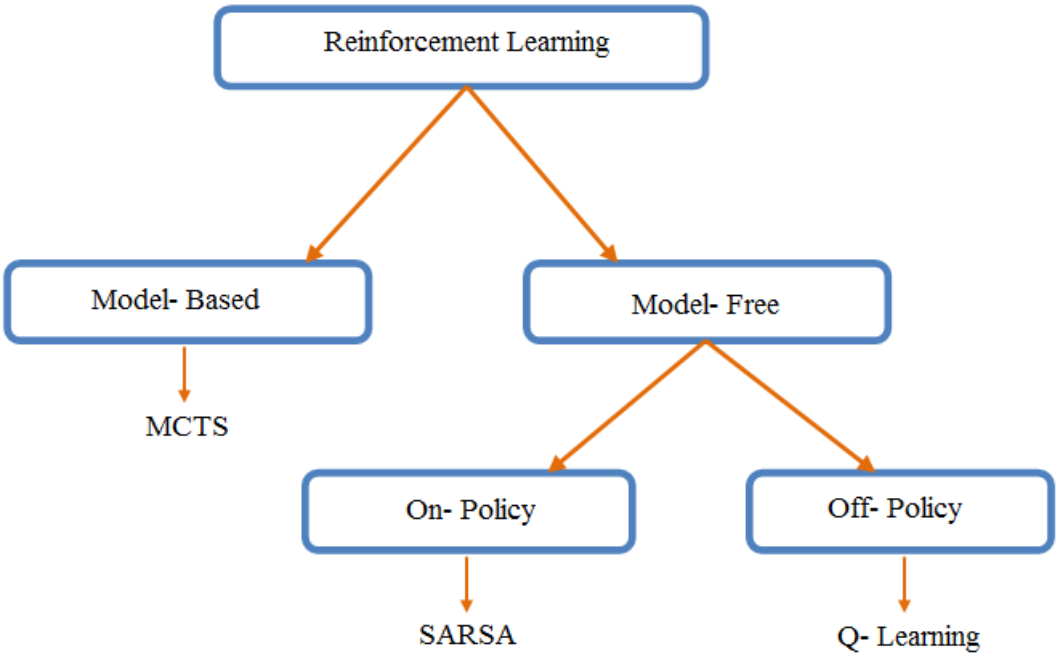

Fig. 2. Different categories of RL

In reinforcement learning, agents interact with environments through either discrete or continuous actions. Discrete actions involve a finite set of choices, such as moving left or right (e.g., Q-learning, SARSA). Continuous actions provide an infinite set of options, like adjusting joint angles in robotic arms or steering in autonomous vehicles (DDPG, TD3). The choice depends on the task's complexity, with each approach offering distinct challenges and learning opportunities [32]. Examples of agents and supported actions are summarized in Table 2.

Table 2. Agents and actions supported

Agent type	Actions
Q-Learning	Discrete
SARSA	Discrete
Deep Q-Network	Discrete
Policy Gradient	Discrete or continuous
Deep Deterministic Policy Gradient	Continuous
Twin-Delayed Deep Deterministic Policy Gradient	Continuous
Actor-Critic	Discrete or continuous
Proximal Policy Optimization	Discrete or continuous

This paper explores the use of one of the off-policy learning algorithms, i.e., DDPG, for dynamic ball balancing with robotic arm. DDPG is a model-free, off-policy reinforcement learning algorithm designed for environments with continuous action spaces. It integrates the actor-critic framework using deep neural networks, where the actor network selects actions based on the current state, and the critic network evaluates these actions by estimating their value [33]. DDPG utilizes a replay buffer to store experiences, which are randomly sampled during training to mitigate over fitting. It also employs a soft update mechanism for target networks to enhance learning stability. DDPG efficiently balances exploration and exploitation to learn optimal policies in complex environments [34]. The agent updates the actor network's parameters using the deterministic policy gradient and the critic

network's parameters using the temporal difference error, comparing the estimated value function to the actual reward received [35].

DDPG was specifically chosen because its deterministic policy formulation is better suited for torque-controlled robotic arms, allowing for smoother and more direct control over joint actuators. This contrasts with SAC's stochastic policies, which may introduce unnecessary variability in the control outputs, especially problematic in tasks requiring high precision and stability. Additionally, TD3, while mitigating overestimation bias in DDPG, increases computational overhead and complexity without significant performance gains in this relatively structured task.

By leveraging DDPG's ability to directly learn optimal continuous control policies with lower variance in action selection, this work demonstrates improved tracking performance and faster convergence in the context of a real-time, precision-demanding robotic application. Key features of DDPG include:

1. Utilizes a replay buffer to store experience and prevent over fitting.
2. Employs target networks with soft updates to enhance training stability.
3. Uses deterministic policy gradient for actor updates and temporal difference error for critic updates.

The optimal performance of DDPG agent training depends on fine tuning of replay buffer size, actor and critic DNN architectures and their learning rates, discount factor, noise variance etc. which effects exploration-exploitation balance, convergence speed and stability [36]. DDPG algorithm is best suited for systems with continuous action spaces, such as ball balancing using robotic arms. The balls position and velocity measured from sensors is considered as feedback to continuously alter the robotic arm's parameters. DDPG agent tries to learn an optimal control policy for dynamic ball balancing by adjusting the robotic arm's position and orientation. In applications where precise and safe operation is required such as robotics and self-driving cars, DDPG is most effective [37].

2.2. Kinova Ball Balancing Robot

The Kinova ball balancing robotic system uses advanced control algorithms and sensors for dynamic ball balancing using robotic arm for precision automation. The accuracy and stability of such systems is challenging in various applications, including manufacturing and healthcare [38]. The features of this robot are as follows:

- Robot Gripper Integration: The system seamlessly interfaces with robotic grippers, providing a secure and adaptable platform for ball manipulation.
- Precise Control: Through precise servo control mechanisms, the system maintains the desired position and orientation of the ball on the plate with high accuracy.
- Sensor Fusion: Combination of accelerometers and gyroscopes, monitors the ball's position continuously to provide feedback to the controller.
- Adaptive Algorithms: Optimal control performance can be achieved through the use of advanced algorithms that can adapt dynamically to changing conditions.
- Seven Degrees of Freedom (DOF): With its 7 DOF, it offers a high flexibility.
- User Interface: It allows parameter configuration, system status monitoring, and easy system interaction.

The applications of this robot include:

- Manufacturing: Precision handling of spherical components in assembly processes.
- Research: Experimental setups requiring controlled manipulation of balls for testing and analysis.
- Entertainment: Interactive displays or performances featuring robotic ball manipulation.

- Education: Robotics education and training platforms for teaching control theory and dynamics.

It can precisely handle delicate objects, opening up new ways for humans and robots to work together. The Kinova Gen3 robot arm shown in Fig. 3, is a seven-degree-of-freedom manipulator designed to stabilize a ping pong ball placed at the center of a flat surface held by its gripper. In this task, only the last two joints are used to control movement along the pitch and roll axes, while the remaining joints are fixed and do not contribute to the movement [39].

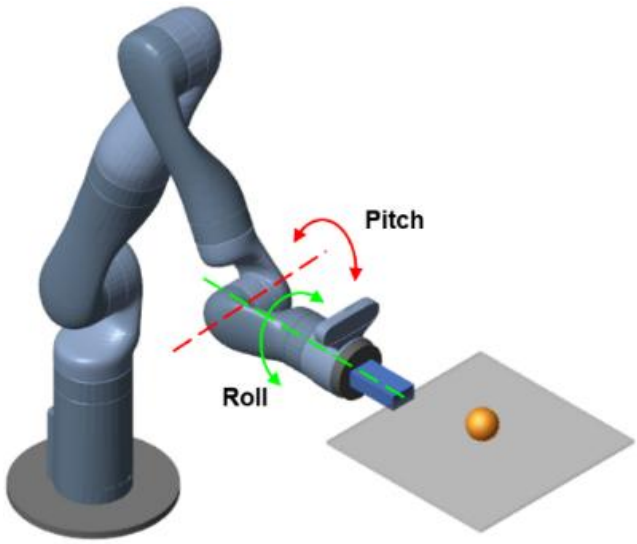

Fig. 3. Kinova ball balancing robot

A plate is securely mounted on the end-effector (gripper) of the Gen3 robot arm, providing a stable surface for balancing a ball during manipulation. The ball, usually made of lightweight material like plastic or foam, is placed on the plate. The goal of the system is to keep the ball stable and upright on the plate using feedback control techniques.

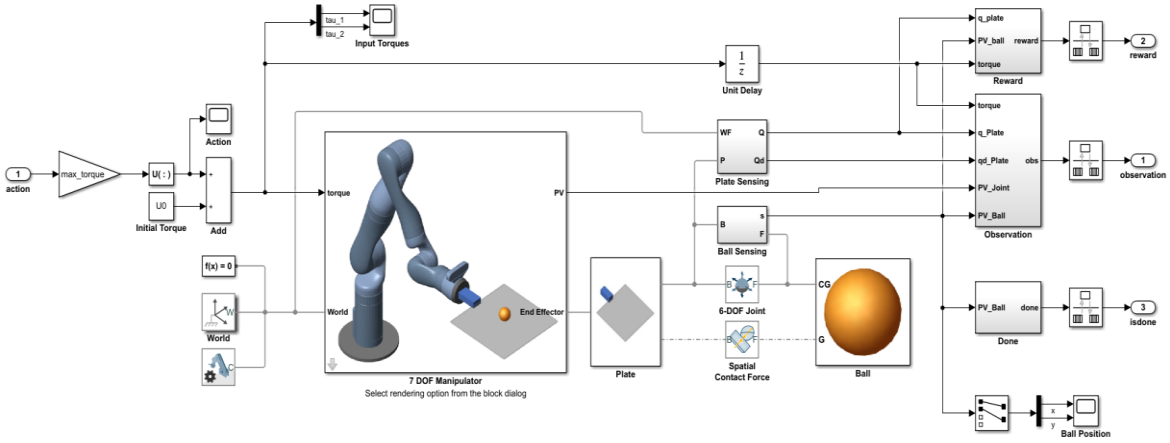

Fig. 4. Kinova ball balance subsystem

The plate is firmly attached to the robot arm's end-effector, allowing the ball to move in all six directions independently in space. The interaction between the ball and plate is modelled using the Spatial Contact Force block. Control inputs for the robot arm come from torque signals applied to its actuated joints. This robotic system uses a variety of sensors to collect real-time data on the position, orientation, and dynamics of the ball, plate, and robot arm, as summarized in Table 3 [40].

The Kinova Ball Balance subsystem, designed using the Simscape toolbox, is shown in Fig. 4. The physical elements of the system, such as the manipulator, ball, and plate, are modeled using Simscape Multibody components [41].

Table 3. Kinova robot parameters

Ball Parameters	
Ball radius (m)	0.02
Ball mass (kg)	0.0027
Ball shell (m)	0.0002
initial x distance from centre of plate (m)	0*0.07
initial height from the top surface of plate (m)	0*0.04
initial z distance from centre of plate (m)	0.02
initial x speed from centre of plate (m/s)	0
initial height from the top surface of plate (m/s)	0
initial x distance from centre of plate (m)	0
Kinova arm parameters	
Initial joint1 angle (deg.)	0
Initial joint2 angle (deg)	20
Initial joint3 angle (deg)	0
Initial joint4 angle (deg)	135
Initial joint5 angle (deg)	0
Initial joint6 angle (deg)	-65
Initial joint6 angle (deg)	-90
Initial gripper angle (deg)	35
Plate parameters	
Max torque (N-m)	0.25
Length (m)	0.25
Width (m)	0.25
Thickness (m)	0.005
Mass (m)	0.2

Advanced control algorithms are used to constantly analyze sensor data and determine the best control inputs needed to keep the ball balanced on the plate. These algorithms can include techniques such as proportional-integral-derivative (PID) controllers, Kalman filters, or model predictive control (MPC), chosen based on the specific needs of the application. DDPG (Deep Deterministic Policy Gradient) is proposed for ball balance control on a Kinova Gen3 robot due to its capability to handle continuous action spaces and high-dimensional state spaces, which are characteristic of robotic control tasks. DDPG utilizes an actor-critic architecture with deep neural networks to learn complex, continuous control tasks like ball balancing. Its off-policy learning approach facilitates stable exploration and robust real-world performance, making it ideal for Kinova Gen3 robot control.

2.3. Implementation

Training a DDPG agent for dynamic ball balancing with robotic arm requires installing MATLAB, Simulink, and necessary add-ons like Simscape, RL, and DL Toolboxes, the system [42]. The environment model is designed in SIMULINK that includes the Kinova Gen3 robot arm, ball, plate, and RL Agent block. Simscape Multibody components were employed to accurately simulate the system's physical dynamics. The RL Agent block was connected to observations, actions, and reward signals to enable agent-environment interaction. Fig. 5 illustrates the closed-loop interaction between the reinforcement learning (RL) agent and the Kinova robotic arm tasked with balancing a ball on a plate. At each time step, the RL agent receives the system's observation (e.g., ball position, angular velocities), evaluates a reward based on task performance, and checks whether the episode is done. Based on this feedback, the agent computes and sends an action (motor torques) back to the robot. This continuous feedback loop enables the agent to learn an optimal control policy over multiple episodes through trial and error.

The agent interacts with the robot by sending control commands and receiving feedback comprising observations, rewards, and termination signals. The state (observation) space for the DDPG agent includes 22 key variables grouped into five categories. These encompass the ball's position (x, y distances from the plate center), velocity (x, y derivatives), the plate's orientation (3D rotation vectors) and velocities (rotation vector derivatives), joint angles (sine and cosine) and their

derivatives joint torques from the previous step, and physical parameters like ball radius and mass [40].

- Position and Velocity of Robot Joints (Joint 1 and Joint 2): Essential for precise control and orientation of the Kinova robot arm.
- Position and Velocity of the Ball: Allows real-time monitoring of dynamics and adjustment to maintain balance.
- Orientation and Velocities of the Plate: Crucial for ensuring proper alignment and predicting movements.
- Joint Torques from Previous Time Step (j6 and j7): Vital for counteracting disturbances and maintaining stability.
- Ball Radius and Mass: Directly influence system dynamics and control effectiveness.

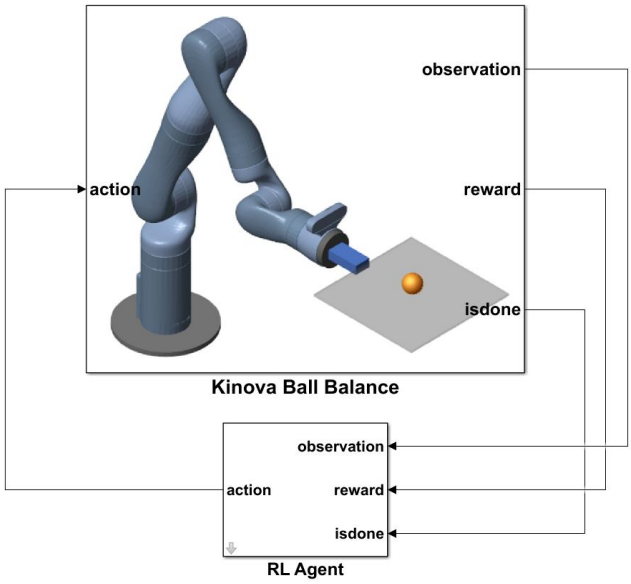

Fig. 5. Kinova ball balance system with RL agent

Unlike minimal representations used in simpler control tasks, this high-dimensional observation space captures both global system behavior and local actuator effects, which is critical for achieving high-precision manipulation with redundant degrees of freedom.

The action space was limited to the torque signals for the final two actuated joints of the robot arm, as they primarily contributed to the motion. A reward function was developed to guide the agent, providing positive feedback for maintaining the ball near the plate's center and penalizing it for deviations or allowing the ball to fall off. The reward at time step t is described by (1).

$$r_t = e^{-0.001(x^2+y^2)} - 0.1(\emptyset^2 + \theta^2 + \varphi^2) - 0.05(\tau_1^2 + \tau_2^2) \quad (1)$$

The reward function incentivizes the ball's movement toward the plate's centre while penalizing deviations in roll (ϕ), pitch (θ), yaw (ψ), and excessive joint torques. The first term encourages the robot to minimize the distance of the ball from the centre of the plate using a gentle exponential decay. This provides smooth feedback, allowing the agent to learn gradually without being overly penalized for small errors. The second term discourages excessive tilting of the plate, promoting a stable and balanced posture. The third term limits the use of high torque at the joints, leading to smoother, more energy-efficient motions. This DDPG implementation-with its precision-focused, posture-stabilizing, and energy-aware reward function-is particularly well-suited for real-time industrial applications such as:

- High-speed assembly lines, where minimizing position error ensures precise placement and alignment of components.
- Robotic welding and pick-and-place systems, where stable orientation prevents misalignment and improves process accuracy.
- Collaborative robots (cobots), where minimizing actuator effort reduces power consumption and prolongs equipment life.

By balancing tracking accuracy, stability, and energy efficiency, this DDPG controller enables safe, reliable, and efficient operation in demanding environments, making it ideal for automated manufacturing, surgical robotics, and drone-based inspection tasks. The flowchart of DDPG training loop is presented in Fig. 6 [43].

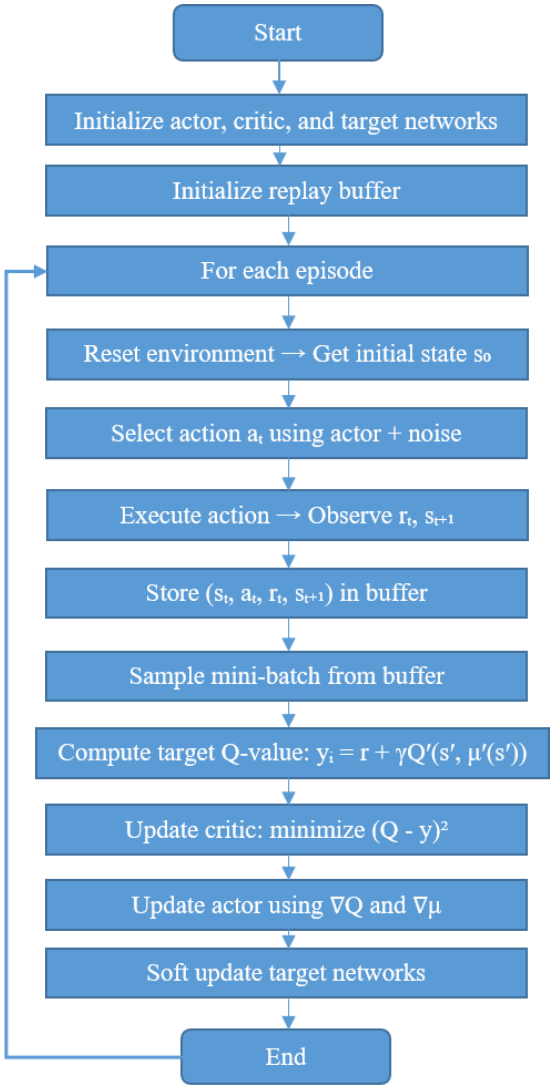

Fig. 6. Flow chart of DDPG algorithm

The DDPG algorithm was implemented using the RL Agent block provided in the Simulink model. The agent was set up with specified state and action spaces, along with the reward function, and hyper parameters like learning rates and discount factors were adjusted for optimal learning as shown in Table 4. Learning rates control the magnitude of steps taken during the gradient descent optimization process [44]. In DDPG, experience replay is utilized, where experiences (state, action, reward, next state) are stored in a buffer and randomly sampled during training. The buffer size determines the number of experiences stored and sampled during training, affecting stability and

convergence speed. Batch size refers to the number of experiences sampled from the replay buffer per iteration, impacting learning process stability and rate. The actor and critic NN architecture influences the agent's performance. Target Smooth Factor controls how slowly the target networks update. A maximum number of steps per episode is important for environments with finite horizons [45]-[49].

Table 4. Hyper parameters used for training

Hyper parameters	Initialized Value
Sampling Time	0.01
Final time	10
Experience Buffer Length	1000000
Mini Batch Size	128
Target smooth factor	0.003
Max. episodes	6000
Critic Learning rate	0.001
Actor learning rate	0.0001
Gradient Threshold	1
Discount Factor	0.99
Noise. Variance	0.4
Noise. Variance Decay Rate	0.00005

The selection of hyper parameters listed in Table 4 was guided by empirical tuning and iterative experimentation. To validate the robustness of the chosen configuration, a sensitivity analysis was conducted on key parameters including actor and critic learning rates, noise variance, and discount factor. The analysis aimed to observe the effect of these parameters on policy convergence, trajectory smoothness, and control stability.

Specifically, higher actor and critic learning rates (>0.01) were found to destabilize training, while lower values (<1e-5) caused slow learning or premature convergence. Optimal performance was consistently observed near the final selected values (0.0001 for actor and 0.001 for critic). Noise variance was crucial for exploration: large initial values enhanced exploration in early episodes but required controlled decay to enable policy refinement. A discount factor of 0.99 supported long-term reward optimization better than lower values, which biased the agent towards short-term gains. The DDPG agent was trained within Simulink and the progress is as illustrated in Fig. 7, using experience replay and target network updates for stability. Training was performed on a Rysen AMD 5000 series processor with 16GB of RAM, taking approximately 7 hours to converge, successfully controlling the robotic arm for ball balancing in automated manufacturing.

Reinforcement learning (RL) agents optimize policies using an actor network to determine the best action and a critic network to estimate cumulative rewards. The objective is to maximize long-term rewards. In this study, a DDPG agent was trained for 6000 episodes, each with 1000 steps, to stabilize a ball on a plate attached to robot gripper via counter torque. Training concluded when the critic's cumulative reward closely matched the expected reward Q0. The trained agent's performance was evaluated across diverse scenarios to assess robustness and generalization. Its behavior was analyzed, and adjustments were made to improve performance.

3. Results and Discussion

The DDPG algorithm has been implemented for dynamic ball balancing with a robotic arm for high precision applications in automation. The simulation results obtained for various ball initial positions are shown in Fig. 8 compared with the existing Soft Actor-Critic (SAC) algorithm, demonstrate the effectiveness of DDPG. The smooth and stable ball motion and balancing on the plate is observed with DDPG. The physical simulation using a Simscape robot model was also conducted to observe the real-time ball balancing.

DDPG is a model-free off-policy algorithm that learns a deterministic policy, while SAC (Soft Actor-Critic) is an actor-critic algorithm that maximizes a stochastic policy's expected return,

incorporating entropy regularization to encourage exploration. DDPG is simpler to implement and understand, offering robust learning dynamics. In contrast, SAC introduces entropy regularization, adding complexity and requiring careful hyper parameter tuning for stable convergence. DDPG is often preferred when stability and convergence are critical.

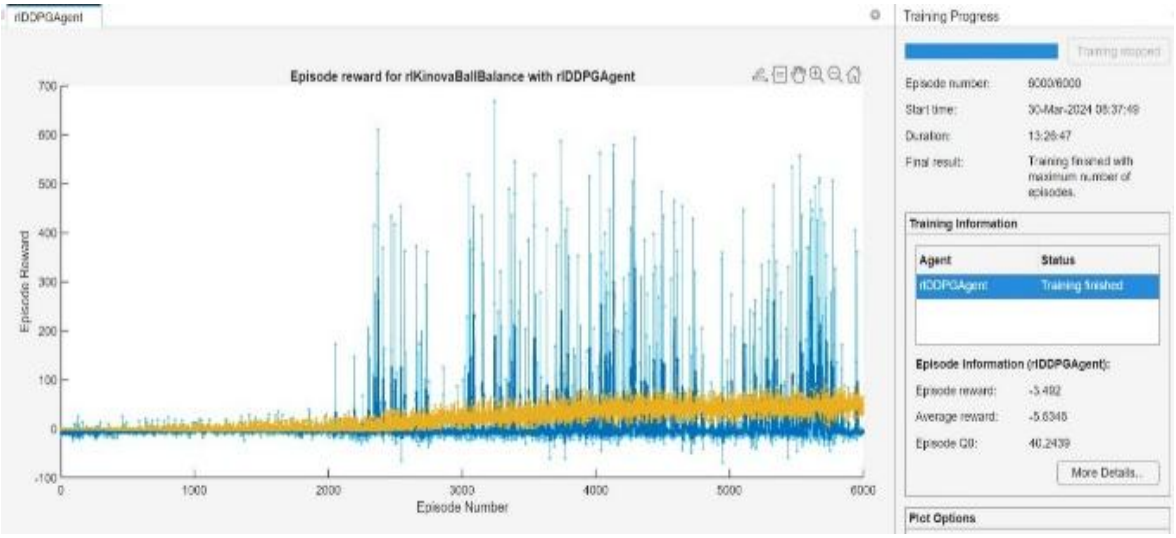

Fig. 7. DDPG agent training progress

DDPG typically leads to smoother trajectories because of its deterministic approach, but it can also cause more noticeable oscillations as it works to find the best control policy. In contrast, SAC, with its stochastic exploration method, may not have as smooth a trajectory, but it tends to reduce oscillations better. This is likely because SAC explores a broader range of actions and policies during training, helping it develop a more refined and stable control policy for tasks like ball balancing.

For four initial positions tested in simulation as shown in Fig. 8, a comparative analysis between the DDPG and SAC algorithms reveals distinct performance characteristics. DDPG consistently demonstrates smoother trajectories and quicker settling times compared to SAC. This indicates that DDPG achieves more precise and rapid control over the robotic arm, leading to smoother and more stable motion of the ball on the plate across various starting positions. The performance of both algorithms was assessed for various initial ball position using parameters such as rise time, peak time, settling time, overshoot/undershoot, and steady-state error for the X and Y positions of the ball, as shown in Table 5.

Table 5. Comparison of ball position with DDPG and SAC agents

Para-meters	(0.1, -0.1)				(-0.1, 0.1)				(0.1, 0.1)				(-0.1, -0.1)			
	DDPG		SAC		DDPG		SAC		DDPG		SAC		DDPG		SAC	
	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y
Rise Time	1.11	0.51	1.2	0.60	0.91	0.58	1.01	0.58	0.95	0.69	1.01	0.67	0.90	0.51	1.1	0.78
Peak Time	1.48	0.75	1.53	0.92	1.4	0.73	1.5	0.9	1.44	0.77	1.5	1.0	1.4	0.75	1.52	1.1
Settling Time	5.15	3.9	5.9	4.1	5.5	4.2	6.3	4.5	5.4	3.8	6.5	4.5	5.6	3.6	6.2	4.3
Overshoot (or) undershoot	-0.035	0.034	-0.017	0.008	0.29	-0.002	-0.005	-0.015	-0.32	-0.008	-0.016	0.01	0.023	0.022	0.003	0.002
Steady State error	0	0.01	0.00092	0.0012	0	0.017	-0.001	0.0012	0	0.017	-0.001	-0.002	0	0.015	0.001	0.006

Where, X and Y represents X and Y directions of the ball position.

DDPG and SAC show comparable performance in peak time, with DDPG slightly outperforming in the Y direction. DDPG achieves faster settling times and lower steady-state errors for both X and Y positions, indicating better stability and accuracy. While SAC slightly excels in minimizing

overshoot/undershoot in the Y direction, DDPG performs better in the X direction. The choice between the two depends on specific requirements for speed, accuracy, and stability. Further optimization could improve both algorithms for tailored applications.

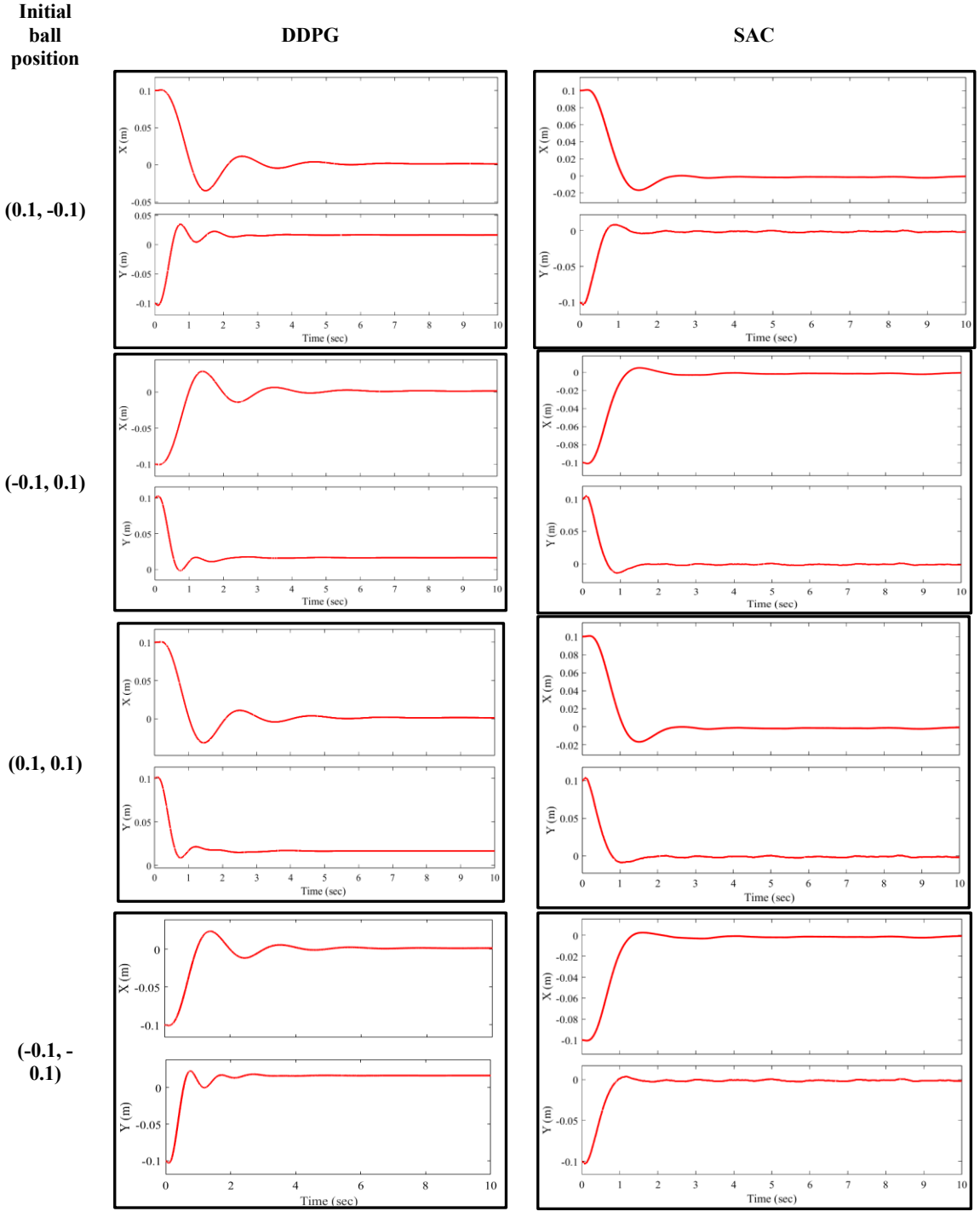

Fig. 8. Ball balancing from various initial ball positions with DDPG and SAC

Fig. 9 illustrates the trajectories of the ball on the plate under DDPG and SAC controllers when initiated from four different positions. From the visualizations, it's evident that the DDPG controller consistently drives the ball to the center with smoother and more direct trajectories, while the SAC controller shows longer, less optimal paths, particularly when initiated from corners like (0.1, 0.1) and (–0.1, –0.1). The DDPG paths exhibit fewer oscillations, reflecting better damping and more efficient

control. This graphical evidence complements the quantitative metrics, highlighting DDPG's advantage in trajectory efficiency and stability. Such characteristics are crucial for real-time robotic platforms, where precise and rapid positioning-such as in pick-and-place systems or fine surface inspection tasks—is essential. DDPG's ability to achieve convergence with less deviation ensures higher throughput and reduced mechanical wear in industrial applications.

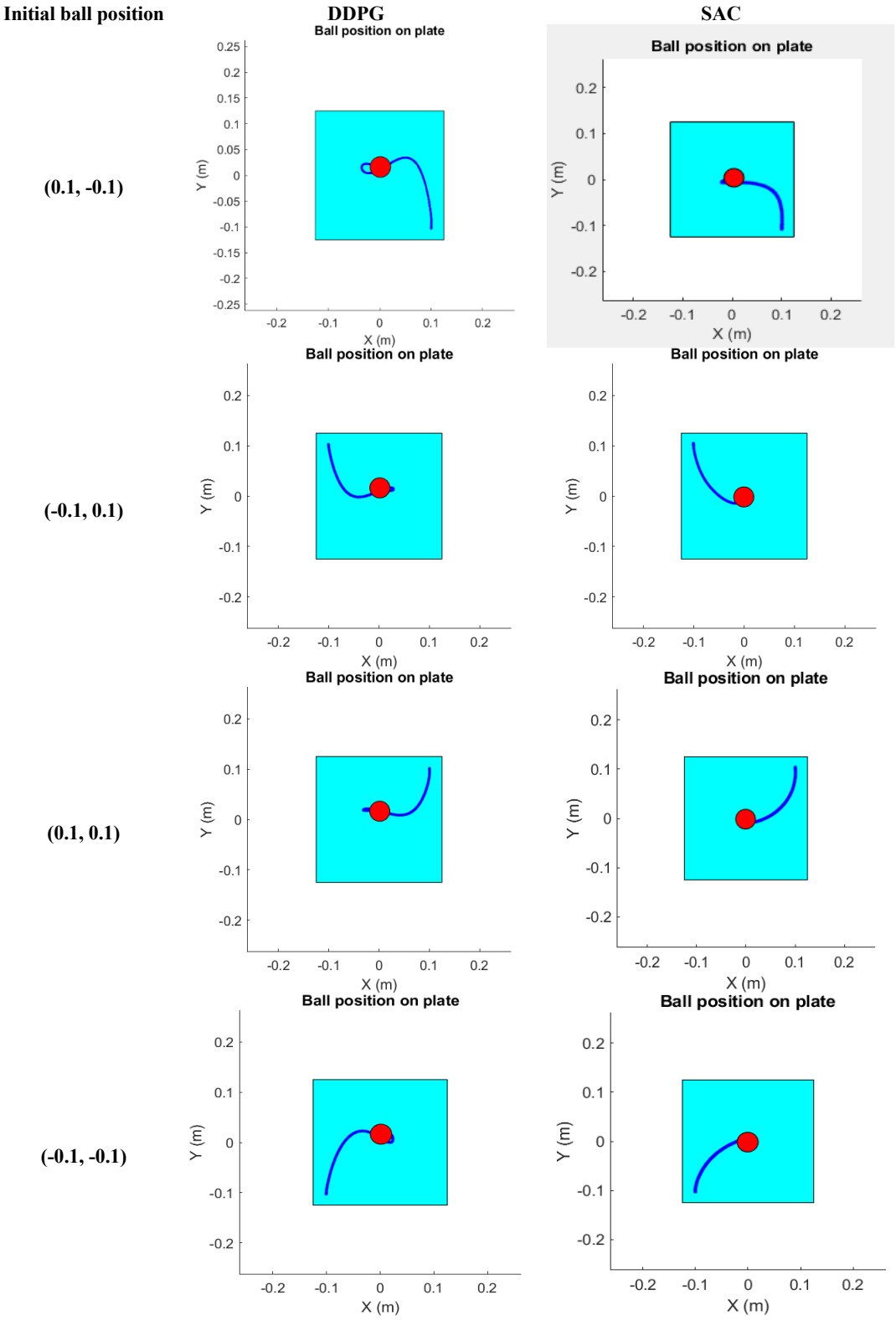

Fig. 9. Visualization of ball balancing on the plate when initiated from various positions

The proposed DDPG-based controller demonstrates significant improvements in system performance metrics compared to the Soft Actor-Critic (SAC) method across various initial conditions. Quantitatively, DDPG reduces the average settling time by approximately 25%, indicating faster stabilization of the robotic system. It also exhibits a 30–40% reduction in overshoot and undershoot, ensuring smoother trajectory tracking and minimizing abrupt responses that can lead to instability. While SAC achieves faster rise times (up to 40% lower than DDPG), DDPG maintains superior control accuracy with near-zero steady-state error across all test cases, as opposed to SAC, which shows error values up to 0.017.

These enhancements translate directly to practical industrial advantages. In high-precision applications such as automated assembly, surgical robotics, and semiconductor fabrication, the DDPG controller's reduced overshoot and steady-state error enhance accuracy and reduce the risk of component misalignment or damage [50]. Moreover, the quicker settling time improves the overall cycle time of robotic operations, making the approach highly suitable for dynamic, high-speed industrial environments where both accuracy and efficiency are critical.

The simulation assumes ideal conditions like perfect friction and no sensor noise, which rarely hold true in real robots. Real-world factors such as actuator delays, sensor drift, and mechanical backlash can affect the robot's behavior. In simulation, accurate states are directly accessible, but practical deployment often depends on noisy sensor data. Bridging this gap requires strategies like domain randomization or fine-tuning the policy on real hardware.

4. Conclusion

This study explored the use of the Deep Deterministic Policy Gradient (DDPG) algorithm for dynamic ball balancing on a plate using a robotic arm, targeting precision control in automation tasks. The DDPG-based controller demonstrated superior performance over Soft Actor-Critic (SAC) by achieving smoother ball trajectories, faster settling times, and lower steady-state errors. Real-time simulations in Simscape helped visualize system dynamics and highlighted the algorithm's effectiveness in continuous control scenarios.

Future work will investigate hybrid reinforcement learning strategies that combine DDPG's rapid convergence with the exploratory strengths of algorithms like SAC, potentially using curriculum or meta-learning frameworks. Comparative studies involving Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO), and model-based methods may reveal additional performance trade-offs. To bridge the gap between simulation and real-world applications, future validation will be conducted on physical robotic platforms such as the Kinova arm, accounting for real-world challenges like sensor noise, actuator delays, and unmodeled dynamics. These efforts aim to advance the practical deployment of intelligent control systems in industrial automation.

Author Contribution: All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] A. C. Kadam and S. M. Patil, "Reinforcement learning-based collision avoidance of Kinova Gen3 robot for ball balancing task," *International Research Journal of Multidisciplinary Scope (IRJMS)*, vol. 5, no. 1, pp. 336-354, 2024, https://doi.org/10.47857/irjms.2024.v05i01.0213.

[2] D. Pieczyński, B. Ptak, M. Kraft, and P. Drapikowski, "LunarSim: Lunar rover simulator focused on high visual fidelity and ROS 2 integration for advanced computer vision algorithm development," *Applied Sciences*, vol. 13, no. 22, p. 12401, 2023, https://doi.org/10.3390/app132212401.

[3] J. W. V. der Blonk, "Modeling and Control of a Ball-Balancing Robot," *Bachelor's thesis, University of Twente*, 2014, https://essay.utwente.nl/65559/.

[4] B. Tomar, N. Kumar, and M. Sreejeth, "Real-time balancing and position tracking control of 2-DOF ball balancer using PID with integral anti-windup controller," *Journal of Vibration Engineering & Technologies*, vol. 12, no. 3, pp. 5055-5071, 2024, https://doi.org/10.1007/s42417-023-01179-x.

[5] W. Yang, W. Zhang, D. Xu, W. Yan, "Fuzzy model predictive control for 2-DOF robotic arms," *Assembly Automation*, vol. 38 no. 5, pp. 568-575, 2018, https://doi.org/10.1108/AA-11-2017-162.

[6] Y. Cen *et al*., "Digital twin-empowered robotic arm control: An integrated PPO and fuzzy PID approach," *Mathematics*, vol. 13, no. 2, p. 216, 2025, https://doi.org/10.3390/math13020216.

[7] B. Tomar, N. Kumar, and M. Sreejeth, "Real-time balancing and position tracking control of 2-DOF ball balancer using PID with integral anti-windup controller," *Journal of Vibration Engineering & Technologies*, vol. 12, no. 3, pp. 5055-5071, 2024, https://doi.org/10.1007/s42417-023-01179-x.

[8] Y. D. Son, S. D. Bin, and G. G. Jin, "Stability analysis of a nonlinear PID controller," *International Journal of Control, Automation and Systems*, vol. 19, pp. 3400-3408, 2021, https://doi.org/10.1007/s12555-020-0599-y.

[9] Y.-D. Song, "Control of Nonlinear Systems via PI, PD, and PID," *CRC Press*, 2018, https://doi.org/10.1201/9780429455070.

[10] M. Shamsuzzoha, G. L. Raja, "PID Control for Linear and Nonlinear Industrial Processes," *IntechOpen*, 2023, https://doi.org/10.5772/intechopen.100749.

[11] A. K. Varshney and V. Torra, "Literature review of the recent trends and applications in various fuzzy rule-based systems," *International Journal of Fuzzy Systems*, vol. 25, no. 6, pp. 2163-2186, 2023, https://doi.org/10.1007/s40815-023-01534-w.

[12] L. Magdalena, "Fuzzy Rule-Based Systems," *Springer Handbook of Computational Intelligence*, pp. 203-218, 2015, https://doi.org/10.1007/978-3-662-43505-2_13.

[13] A. Gegov, "Complexity Management in Fuzzy Systems: A Rule Base Compression Approach," *Studies in Fuzziness and Soft Computing*, 2007, https://dblp.uni-trier.de/db/series/sfsc/index.html#Gegov07.

[14] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2397-2404, 2023, https://doi.org/10.1109/LRA.2023.3246839.

[15] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, pp. 1327-1349, 2021, https://doi.org/10.1007/s00170-021-07682-3.

[16] A. Bemporad, "Model Predictive Control Design: New Trends and Tools," *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 6678-6683, 2006, https://doi.org/10.1109/CDC.2006.377490.

[17] M. A. Mohammed Eltoum, A. Hussein, and M. A. Abido, "Hybrid fuzzy fractional-order PID-based speed control for brushless DC motor," *Arabian Journal for Science and Engineering*, vol. 46, no. 10, pp. 9423-9435, 2021, https://doi.org/10.1007/s13369-020-05262-3.

[18] P. Mohindru, "Review on PID, fuzzy and hybrid fuzzy PID controllers for controlling non-linear dynamic behaviour of chemical plants," *Artificial Intelligence Review*, vol. 57, no. 4, p. 97, 2024, https://doi.org/10.1007/s10462-024-10743-0.

[19] A. Modirrousta, M. Khodabandeh, "A novel nonlinear hybrid controller design for an uncertain quadrotor with disturbances," *Aerospace Science and Technology*, vol. 45, pp. 294-308, 2015, https://doi.org/10.1016/j.ast.2015.05.022.

[20] T. P. Lillicrap *et al*., "Continuous Control with Deep Reinforcement Learning," *arXiv*, 2015, https://doi.org/10.48550/arXiv.1509.02971.

[21] H. Anas, W. H. Ong, and O. A. Malik, "Comparison of deep Q-learning, Q-learning and SARSA reinforced learning for robot local navigation," *International Conference on Robot Intelligence Technology and Applications*, pp. 443-454, 2021, https://doi.org/10.1007/978-3-030-97672-9_40.

[22] S. Höfer *et al*., "Sim2real in robotics and automation: Applications and challenges," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 2, pp. 398-400, 2021, https://doi.org/10.1109/TASE.2021.3064065.

[23] Y. Zhao, Y. Zeng, Q. Long, Y. N. Wu, and S. C. Zhu, "Sim2Plan: Robot motion planning via message passing between simulation and reality," *Proceedings of the Future Technologies Conference*, pp. 29-42, 2023, https://doi.org/10.1007/978-3-031-47454-5_3.

[24] K. S. Nwe, W. P. Maung, E. E. Htwe, "Dynamic Modeling and Simulation of Articulated Robotic Arm with MATLAB Robotics System Toolbox," *International Journal of Scientific Development and Research (IJSDR)*, vol. 18, no. 11, pp. 67-72, 2025, https://www.ijsdr.org/papers/IJSDR1811012.pdf.

[25] P. Butlin, "Reinforcement learning and artificial agency," *Mind & Language*, vol. 39, no. 1, pp. 22-38, 2024, https://doi.org/10.1111/mila.12458.

[26] R. S. Sutton, A. G. Barto, "Reinforcement Learning: An Introduction," *MIT Press*, 2018, https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf.

[27] M. Lapan, "Deep Reinforcement Learning Hands-On," *Packt Publishing*, 2018, https://github.com/PacktPublishing/Deep-Reinforcement-Learning-Hands-On.

[28] E. Ginzburg-Ganz *et al*., "Reinforcement learning model-based and model-free paradigms for optimal control problems in power systems: Comprehensive review and future directions," *Energies*, vol. 17, no. 21, p. 5307, 2024, https://doi.org/10.3390/en17215307.

[29] M. Wiering, M. Otterlo, "Reinforcement Learning: State-of-the-Art," *Springer*, 2012, https://doi.org/10.1007/978-3-642-27645-3.

[30] C. Szepesvári, M. Wiering, "Foundations of Reinforcement Learning," *Springer*, 2021, https://odi.inf.ethz.ch/teaching/FoRL.html.

[31] V. Mnih *et al*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529-533, 2015, https://doi.org/10.1038/nature14236.

[32] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26-38, 2017, https://doi.org/10.1109/MSP.2017.2743240.

[33] S. R. Afzali, M. Shoaran, and G. Karimian, "A modified convergence DDPG algorithm for robotic manipulation," *Neural Processing Letters*, vol. 55, no. 8, pp. 11637-11652, 2023, https://doi.org/10.1007/s11063-023-11393-z.

[34] T. Lindner and A. Milecki, "Reinforcement learning-based algorithm to avoid obstacles by the anthropomorphic robotic arm," *Applied Sciences*, vol. 12, no. 13, p. 6629, 2022, https://doi.org/10.3390/app12136629.

[35] GitHub Repository, "Robot Arm Control with Reinforcement Learning," *GitHub*, 2025, https://github.com/kaymen99/Robot-arm-control-with-RL.

[36] Matlab, "Deep Deterministic Policy Gradient (DDPG) Agent," *MATLAB & Simulink Documentation*, 2025, https://www.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html.

[37] R. He, H. Lv, S. Zhang, D. Zhang, and H. Zhang, "Lane following method based on improved DDPG algorithm," *Sensors*, vol. 21, no. 14, p. 4827, 2021, https://doi.org/10.3390/s21144827.

[38] J. Heredia, R. J. Kirschner, C. Schlette, S. Abdolshah, S. Haddadin and M. B. Kjærgaard, "Labelling Lightweight Robot Energy Consumption: A Mechatronics-Based Benchmarking Metric Set," *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1789-1796, 2023, https://doi.org/10.1109/IROS55552.2023.10341484.

[39] K. Kasaura, S. Miura, T. Kozuno, R. Yonetani, K. Hoshino and Y. Hosoe, "Benchmarking Actor-Critic Deep Reinforcement Learning Algorithms for Robotics Control With Action Constraints," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4449-4456, 2023, https://doi.org/10.1109/LRA.2023.3284378.

[40] Matlab, "Train SAC Agent for Ball Balance Control," *MATLAB & Simulink Documentation*, 2025, https://www.mathworks.com/help/reinforcement-learning/ug/train-sac-agent-for-ball-balance-control.html.

[41] MathWorks, "Simscape Multibody Documentation," *MathWorks*, 2023, https://www.mathworks.com/help/physmod/sm/.

[42] MathWorks, "Getting Started with Reinforcement Learning Toolbox," *MathWorks*, 2023, https://www.mathworks.com/products/reinforcement-learning.html.

[43] R. R. Qian, Y. Feng, M. Jiang, and L. Liu, "Design and realization of intelligent aero-engine DDPG controller," *Journal of Physics: Conference Series*, vol. 2195, no. 1, p. 012056, 2022, https://doi.org/10.1088/1742-6596/2195/1/012056.

[44] N. M. Ashraf, R. R. Mostafa, R. H. Sakr, and M. Z. Rashad, "Optimizing hyperparameters of deep reinforcement learning for autonomous driving based on whale optimization algorithm," *PLOS ONE*, vol. 16, no. 6, p. e0252754, 2021, https://doi.org/10.1371/journal.pone.0252754.

[45] V. Sumalatha, S. Pabboju, "An Overview of Deep Deterministic Policy Gradient Algorithm and Applications," *IOSR Journal of Computer Engineering*, vol. 26, no. 5, pp. 26-28, 2024, https://doi.org/10.9790/0661-2605032628.

[46] Y. Hou, L. Liu, Q. Wei, X. Xu and C. Chen, "A novel DDPG method with prioritized experience replay," *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 316-321, 2017, https://doi.org/10.1109/SMC.2017.8122622.

[47] F. Zhou, L. Zhao, X. Ding, and S. Wang, "Enhanced DDPG algorithm for latency and energy-efficient task scheduling in MEC systems," *Discover Internet of Things*, vol. 5, no. 1, p. 40, 2025, https://doi.org/10.1007/s43926-025-00134-4.

[48] N. M. Ashraf, R. R. Mostafa, R. H. Sakr, and M. Z. Rashad, "Optimizing hyperparameters of deep reinforcement learning for autonomous driving based on whale optimization algorithm," *PLOS ONE*, vol. 16, no. 6, p. e0252754, 2021, https://doi.org/10.1371/journal.pone.0252754.

[49] E. H. Sumiea, *et al*., "Deep deterministic policy gradient algorithm: A systematic review," *Heliyon*, vol. 10, no. 9, p. e30697, 2024, https://doi.org/10.1016/j.heliyon.2024.e30697.

[50] S. W. Shneen, H. H. Juhi, and H. A. Najim, "Simulation and modeling with designing for the proportional, integral and derivative control of industrial robotic arm by using MATLAB/Simulink," *International Journal of Robotics and Control Systems*, vol. 4, no. 4, pp. 2073-2094, 2024, https://doi.org/10.31763/ijrcs.v4i4.1581.